NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

PhD THESIS

The DEMOS family of e-voting systems: End-to-end verifiable elections in the standard model

Thomas M. Zacharias

ATHENS

JULY 2016

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Η οικογένεια συστημάτων ηλεκτρονικής ψηφοφορίας DEMOS: Άμεση επαληθευσιμότητα στο standard μοντέλο

Θωμάς Μ. Ζαχαρίας

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2016

PhD THESIS

The DEMOS family of e-voting systems: End-to-end verifiable elections in the standard model

Thomas M. Zacharias

SUPERVISOR: Aggelos Kiayias, Associate Professor UoA

THREE-MEMBER ADVISORY COMMITTEE:

Aggelos Kiayias, Associate Professor UoA

Ioannis Emiris, Professor UoA

Stavros Kolliopoulos, Associate Professor UoA

SEVEN-MEMBER EXAMINATION COMMITTEE

Aggelos Kiayias,
Associate Professor UoA

Ioannis Emiris,
Professor UoA

Stavros Kolliopoulos,
Associate Professor UoA

Alex Delis,
Professor UoA

Panagiotis Rondogiannis,
Associate Professor UoA

Mema Roussopoulos,
Associate Professor UoA

Vassilis Zikas,
Assistant Professor Rensselaer PI

Examination Date: July 14, 2016

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Η οικογένεια συστημάτων ηλεκτρονικής ψηφοφορίας DEMOS: Άμεση επαληθευσιμότητα στο standard μοντέλο

Θωμάς Μ. Ζαχαρίας

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Άγγελος Κιαγιάς, Αναπληρωτής Καθηγητής ΕΚΠΑ

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:

Άγγελος Κιαγιάς, Αναπληρωτής Καθηγητής ΕΚΠΑ

Ιωάννης Εμίρης, Καθηγητής ΕΚΠΑ

Σταύρος Κολλιόπουλος, Αναπληρωτής Καθηγητής ΕΚΠΑ

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Άγγελος Κιαγιάς,
Αναπληρωτής Καθηγητής ΕΚΠΑ

Ιωάννης Εμίρης,
Καθηγητής ΕΚΠΑ

Σταύρος Κολλιόπουλος,
Αναπληρωτής Καθηγητής ΕΚΠΑ

Αλέξης Δελής,
Καθηγητής ΕΚΠΑ

Παναγιώτης Ροντογιάννης,
Αναπληρωτής Καθηγητής ΕΚΠΑ

Μέμα Ρουσσοπούλου,
Αναπληρώτρια Καθηγήτρια ΕΚΠΑ

Βασίλειος Ζήκας,
Επίκουρος Καθηγητής Rensselaer PI

Ημερομηνία Εξέτασης: 14 Ιουλίου 2016

ABSTRACT

Political activity in every modern democratic state is built upon composite democratic procedures, which in turn rely on functional and secure voting systems, operating as the medium that expresses peoples' will. During the last decades, *electronic voting (e-voting) systems* have emerged as promising technologies for carrying out democratic procedures in the Digital Age. Their major advantages comprise the facilitation of participation of social groups with considerable physical barriers, reduction of the election cost and acceleration of the set up, voting and tally phases during an election execution. By today, e-voting systems have been used in several countries either in pilot executions (Australia, England, Ireland, Italy, Norway) or binding elections at a municipality or national level (Belgium, Brazil, Canada, Estonia, India, Switzerland, USA). Nonetheless, they have been subject to occasionally trenchant criticism, mainly due to the disquiet about potential security threats caused by the amount of power now transferred to the machines.

This PhD thesis addresses the prospect of e-voting from the viewpoint of contemporary cryptography. It is widely believed that cryptography is the most conducive scientific area to the advancement of the e-voting concept, providing us with tools and formal frameworks for the construction of efficient systems which security is *mathematically provable*. An indisputable mathematical proof that an e-voting system preserves the integrity and secrecy of an election procedure can be an asset of great value to a democratic society.

The main contribution of this PhD thesis is the introduction of e-voting systems that achieve *end-to-end verifiability in the standard model* for the first time. End-to-end verifiability is a strong security property suggesting that the voters can verify the integrity of the election procedure without putting trust in any administration authority. End-to-end verifiability in the standard model denotes that verification is executed without assuming any trusted setup. Prior to this thesis, all top-tier e-voting systems (e.g. SureVote, JCJ, Prêt à Voter, Helios, Scantegrity, etc.) assumed honesty of the voting clients, the random oracle model, or the existence a randomness beacon to achieve end-to-end verifiability. Therefore, until recently, the feasibility of end-to-end verifiable elections in the standard model remained an open question.

The above question is answered affirmatively in this thesis with the introduction of the *DEMOS-A* and *DEMOS-2* e-voting systems. The two systems follow different approaches with respect to their design. In particular, DEMOS-A follows the *code-voting approach*, where the voters obtain ballots that contain independent and random encodings of the election options (typically vote-codes in one-to-one correspondence with the election options). At the voting phase, the voters cast the encodings that correspond to their intended selections in their ballots. Consequently, vote submission becomes a simple procedure which can be run by devices of minimum computational power. However, this flexibility comes with a price of high complexity at the election preparation phase from the election servers side, resulting in important scalability restrictions for DEMOS-A. To resolve this

issue, this thesis introduces the DEMOS-2 e-voting system, in the spirit of the *client-side encryption*. Namely, in DEMOS-2, the overhead is distributed to the voting clients, which now must be computationally able to locally encrypt the voters' ballots, hence to perform cryptographic operations. As a result, DEMOS-A and DEMOS-2 have complementary benefits and weaknesses regarding their functionality and security, hence the choice of the most preferable system depends on the given election setting.

The design of DEMOS-A and DEMOS-2 consists of an elaborate composition of well-known and ad-hoc cryptographic tools. In the core of both systems, is a novel mechanism that extracts the randomness required for verification from the *entropy generated by the voters*, when they engage in the voting phase. This entropy is *internal* with respect to the election environment, therefore the need for trusting an outer source of randomness is removed. As a result, DEMOS-A and DEMOS-2 achieve end-to-end verifiability assuming only a consistent publicly accessible bulletin board, a requirement that can be seen as a tight condition for this security property.

For a concrete security analysis of DEMOS-A and DEMOS-2, this thesis brings in a strong cryptographic framework that encompasses formal definitions of end-to-end verifiability and voter privacy, where the latter integrates the property of coercion resistance against passive adversaries. The sain framework adds to the literature a state-of-the-art methodology for the study of e-voting systems and can be seen as an independent contribution. The provable security of DEMOS-A and DEMOS-2, especially the statements of the respective end-to-end verifiability theorems, reveal that the security level is in high correlation with the auditing behaviour of the electorate. Motivated by this finding, this thesis extends the framework by modelling e-voting systems as *ceremonies*, inspired by the work of Ellison in 2007. In an e-voting ceremony, the human entities are analysed separately from their associated devices and operate as finite state machines (transducers) with limited power. As a case study of an e-voting ceremony, this thesis investigates the security of the well-known Helios e-voting system by providing formal proofs as well as evaluations based on real-world and simulation data.

The introduction of DEMOS-A and DEMOS-2 gave birth to the DEMOS family of e-voting systems. Concurrent with the writing of this PhD thesis, the DEMOS family is enlarged by the addition of systems that address technical challenges not resolved by the two original systems, while enjoying end-to-end verifiability in the standard model and voter privacy. The thesis is concluded with an overview of its findings, a brief description of the said follow up systems and a list of intriguing directions for future research.

SUBJECT AREA: Electronic voting

KEYWORDS: cryptography, end-to-end verifiability, security modelling, standard model

ΠΕΡΙΛΗΨΗ

Η πολιτική δραστηριότητα σε κάθε σύγχρονη δημοκρατική πολιτεία συντελείται από σύνθετες δημοκρατικές διαδικασίες, οι οποίες με τη σειρά τους βασίζονται σε λειτουργικά και ασφαλή συστήματα ψηφοφορίας ως μέσα έκφρασης της λαϊκής βούλησης. Τις τελευταίες δεκαετίες, τα *συστήματα ηλεκτρονικής ψηφοφορίας* (e-voting systems) έχουν προταθεί ως μία υποσχόμενη τεχνολογία για τη διεξαγωγή δημοκρατικών διαδικασιών στα πλαίσια της Ψηφιακής Εποχής. Τα βασικά τους πλεονεκτήματα συνίστανται στη διευκόλυνση της συμμετοχής κοινωνικών ομάδων που αντιμετωπίζουν φυσικά εμπόδια και κινησιακές δυσκολίες, καθώς και στη μείωση του κόστους και του απαιτούμενου χρόνου της εκλογικής διαδικασίας σε σημαντικό βαθμό. Μέχρι σήμερα, τα συστήματα ηλεκτρονικής ψηφοφορίας έχουν εφαρμοστεί τόσο σε πιλοτικά πειράματα όσο και σε εκλογές σε περιφερειακό και εθνικό επίπεδο. Εντούτοις, έχουν υπάρξει αντικείμενα ενίοτε έντονης κριτικής κυρίως λόγω ζητημάτων ασφάλειας που εγείρονται από τη χρήση πιθανώς διαβλητών υπολογιστικών συσκευών.

Η παρούσα διατριβή μελετά το αντικείμενο της ηλεκτρονικής ψηφοφορίας από τη σκοπιά της σύγχρονης κρυπτογραφίας. Είναι γενικώς αποδεκτό ότι η κρυπτογραφία αποτελεί την πλέον πρόσφορη επιστημονική περιοχή για την προώθηση της ηλεκτρονικής ψηφοφορίας, παρέχοντας εργαλεία και τυπικά μοντέλα για την κατασκευή συστημάτων ηλεκτρονικής ψηφοφορίας των οποίων η ασφάλεια είναι *αποδείξιμη μαθηματικά*. Μία αδιαμφισβήτητη μαθηματική απόδειξη ότι ένα σύστημα ηλεκτρονικής ψηφοφορίας διατηρεί την ακεραιότητα και τη μυστικότητα της εκλογικής διαδικασίας συνιστά πολύτιμη επιστημονική προσφορά σε μία δημοκρατική κοινωνία.

Η σημαντικότερη συνεισφορά της παρούσας διατριβής είναι η εισαγωγή συστημάτων που επιτυγχάνουν για πρώτη φορά *άμεση επαληθευσιμότητα* (end-to-end verifiability) στο *standard* μοντέλο. Η άμεση επαληθευσιμότητα είναι μια ισχυρή ιδιότητα ασφάλειας, σύμφωνα με την οποία ο ψηφοφόρος μπορεί να επαληθεύσει την ακεραιότητα της εκλογικής διαδικασίας χωρίς να εναποθέτει εμπιστοσύνη σε καμία εκλογική αρχή. Η άμεση επαληθευσιμότητα στο standard μοντέλο υποδηλώνει ότι για την επαλήθευση δεν απαιτείται η υπόθεση κάποιας έμπιστης παραμέτρου στο εκλογικό περιβάλλον. Προγενέστερα της παρούσας διατριβής, όλα τα κορυφαία συστήματα ηλεκτρονικής ψηφοφορίας (π.χ. SureVote, JCJ, Prêt à Voter, Helios, Scantegrity κ.ά.) προϋπέθεταν αδιάβλητες συσκευές ψηψοφορίας, το μοντέλο τυχαίου μαντείου (random oracle model), ή την ύπαρξη μίας έμπιστης πηγής τυχαιότητας (randomness beacon) για την εξασφάλιση της άμεσης επαληθευσιμότητας. Κατά συνέπεια, η ύπαρξη ενός συστήματος με άμεση επαληθευσιμότητα στο standard μοντέλο παρέμενε ανοιχτό ερώτημα.

Το παραπάνω ερώτημα απαντάται καταφατικά στην παρούσα διατριβή με την παρουσίαση των συστημάτων ηλεκτρονικής ψηφοφορίας DEMOS-A και DEMOS-2. Τα δύο συστήματα ακολουθούν διαφορετικές κατευθύνσεις ως προς το στο σχεδιασμό τους. Ειδικότερα, το DEMOS-A ακολουθεί την *code-voting* κατεύθυνση, όπου οι ψηφοφόροι λαμβάνουν

ψηφοδέλτια τα οποία περιέχουν ανεξάρτητα τυχαίες κωδικοποιήσεις των εκλογικών επιλογών (τυπικά κωδικούς σε 1-1 αντιστοιχία με τις εκλογικές επιλογές). Κατά τη διάρκεια της ψηφοφορίας, οι ψηφοφόροι υποβάλλουν τις κωδικοποιήσεις που αντιστοιχούν στις επιλογές τις επιθυμίας τους στα ψηφοδέλτιά τους. Με αυτό τον τρόπο, η υποβολή ψήφου γίνεται μια απλή διαδικασία που μπορεί να πραγματοποιηθεί μέσω συσκευών ελάχιστης υπολογιστικής ισχύος. Ωστόσο, αυτή η ευελιξία έχει ως αντίτιμο την υψηλή πολυπλοκότητα κατά το στάδιο εκλογικής προετοιμασίας από πλευράς των διαχειρστικών αρχών, το οποίο οδηγεί σε σημαντικούς περιορισμούς του DEMOS-A όσον αφορά την επεκτασιμότητα (scalability) του συστήματος. Για την επίλυση αυτού του ζητήματος, η παρούσα διατριβή εισάγει το DEMOS-2, το οποίο είναι σχεδιασμένο στο πνεύμα της *client-side encryption* ψηφοφορίας. Στο DEMOS-2, το υπολογιστικό κόστος επιμερίζεται στις συσκευές ψηφοφορίας, οι οποίες τώρα απαιτούνται να είναι επαρκώς υπολογιστικά ισχυρές ώστε να κρυπτογραφούν τοπικά τα ψηφοδέλτια, επομένως και να εκτελούν κρυπτογραφικές λειτουργίες. Ως εκ τούτου, τα DEMOS-A και DEMOS-2 εμφανίζουν διαφορετικά πλεονεκτήματα και αδυναμίες όσον αφορά τη λειτουργικότητα και την ασφάλεια αποτελώντας δύο συμπληρωματικές προτάσεις διεξαγωγής ηλεκτρονικής ψηφοφορίας, όπου η επιλογή της καταλληλότερης πρότασης εξαρτάται από τις εκάστοτε εκλογικές παραμέτρους.

Ο σχεδιασμός των DEMOS-A και DEMOS-2 περιλαμβάνει καθιερωμένα και εκ νέου κατασκευασμένα κρυπτογραφικά εργαλεία. Στην καρδιά και των δύο συστημάτων εντοπίζεται ένας μηχανισμός εξαγωγής τυχαιότητας, απαιτούμενης για την άμεση επαληθευσιμότητα, από την εντροπία που παράγεται από τη διάδραση των ψηφοφόρων με το σύστημα. Η προκείμενη εντροπία είναι *εσωτερική* ως προς το περιβάλλον ψηφοφορίας, εξαλείφοντας έτσι την ανάγκη ύπαρξης μιας εξωτερικής πηγής τυχαιότοτητας. Ως αποτέλεσμα, τα συστήματα DEMOS-A και DEMOS-2 επιτυγχάνουν άμεση επαληθευσιμότητα υποθέτοντας μόνο την ύπαρξη ενός αδιάβλητου δημόσια προσβάσιμου πίνακα επαλήθευσης (bulletin board), μία προϋπόθεση η οποία μπορεί να δειχθεί ότι είναι ανελαστική για την εν λόγω ιδιότητα.

Για την αυστηρή ανάλυση ασφάλειας των DEMOS-A και DEMOS-2, η παρούσα διατριβή προτείνει ένα πλήρες κρυπτογραφικού πλαισίου το οποίο περιλαμβάνει τυπικούς ορισμός της άμεσης επαληθευσιμότητας καθώς και της *ιδιωτικότητας* (privacy), όπου η τελευταία ιδιότητα συμπεριλαμβάνει την *αντίσταση στον καταναγκασμό* (coercion resistance) απέναντι σε *παθητικούς αντιπάλους* (passive adversaries). Το παραπάνω πλαίσιο προσθέτει στη βιβλιογραφία μία υψηλών προδιαγραφών μεθοδολογία για τη μελέτη των συστημάτων ηλεκτρονικής ψηφοφορίας και μπορεί να θεωρηθεί ως επιπρόσθετη συνεισφορά της παρούσας διατριβής. Τα θεωρήματα άμεσης επαληθευσιμότητας που προκύπτουν από την ανάλυση των DEMOS-A και DEMOS-2, αποκαλύπτουν μία στενή εξάρτηση της ασφάλειας των δύο συστημάτων από τη συμπεριφορά εκλογικού σώματος ως προς τη διαδικασία επαλήθευσης. Το συμπέρασμα αυτό αποτέλεσε κίνητρο για την επέκταση του προαναφερθέντος κρυπτογραφικού πλαισίου, όπου ένα σύστημα ηλεκτρονικής ψηφοφορίας μοντελοποιείται ως *ceremony*, σύμφωνα με την προσέγγιση στο έργο του Ellison το 2007. Σε ένα ceremony ηλεκτρονικής ψηφοφορίας, οι ανθρώπινες οντότητες διαχωρίζονται από τις συσκευές και λειτουργούν ως μηχανές πεπερασμένων καταστάσεων (transducers) περιορισμένης ισχύος. Ως υπόδειγμα μελέτης ενός ceremony ηλεκτρονικής ψηφοφορίας, η παρούσα διατριβή μελετά την ασφάλεια του ευρέως εφαρμοσμένου συστήματος ηλεκτρο-

νικής ψηφοφορίας Helios, παρέχοντας τυπικές αποδείξεις αλλά και πειραματικές εκτιμήσεις βασισμένες σε πραγματικά δεδομένα και προσομοιώσεις.

Η εισαγωγή των DEMOS-A και DEMOS-2 σηματοδότησε τη δημιουργία της οικογένειας των συστημάτων ηλεκτρονικής ψηφοφορίας DEMOS. Παράλληλα με τη συγγραφή της παρούσας διατριβής, η οικογένεια DEMOS διευρύνεται με την προσθήκη συστημάτων που αντιμετωπίζουν τις τεχνικές προκλήσεις που παραμένουν ανεπίλυτες στα δύο αρχικά συστήματα εξακολουθώντας να απολαμβάνουν άμεση επαληθευσιμότητα στο standard μοντέλο και ιδιωτικότητα. Η διατριβή ολοκληρώνεται με μία σύνοψη των αποτελεσμάτων της, σύντομη περιγραφή των νέων ακόλουθων συστημάτων και μία αναφορά σε ενδιαφέρουσες κατευθύνσεις για μελλοντική έρευνα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ηλεκτρονική ψηφοφορία

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: κρυπτογραφία, άμεση επαληθευσιμότητα, μοντελοποίηση ασφάλειας, standard μοντέλο

ΣΥΝΟΠΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

1. Εισαγωγή

Τα *συστήματα ηλεκτρονικής ψηφοφορίας* (e-voting systems) έχουν προταθεί ως μία υποσχόμενη τεχνολογία για τη διεξαγωγή δημοκρατικών διαδικασιών στα πλαίσια της Ψηφιακής Εποχής. Τα βασικά τους πλεονεκτήματα συνίστανται στη διευκόλυνση της συμμετοχής στην εκλογική διαδικασία κοινωνικών ομάδων που αντιμετωπίζουν φυσικά εμπόδια και κινησιακές δυσκολίες, καθώς και στη μείωση του κόστους και του απαιτούμενου χρόνου της εκλογικής διαδικασίας σε σημαντικό βαθμό. Μέχρι σήμερα, τα συστήματα ηλεκτρονικής ψηφοφορίας έχουν εφαρμοστεί τόσο σε πιλοτικά πειράματα (Αγγλία, Αυστραλία, Ιρλανδία, Ιταλία, Νορβηγία) όσο και σε εκλογές σε περιφερειακό και εθνικό επίπεδο (Βέλγιο, Βραζιλία, Ελβετία, Εσθονία, ΗΠΑ, Ινδία, Καναδάς, Ολλανδία). Εντούτοις, έχουν υπάρξει αντικείμενα έντονης κριτικής κυρίως λόγω ζητημάτων ασφάλειας που εγείρονται από τη χρήση πιθανώς διαβλητών υπολογιστικών συσκευών, το οποίο έχει οδηγήσει στον τερματισμό προγραμμάτων ηλεκτρονικής ψηφοφορίας, είτε λόγω διαπιστωμένων κενών ασφάλειας (Ολλανδία, Ιρλανδία) είτε λόγω ανησυχιών των πολιτών για την αξιοπιστία των ηλεκτρονικών εκλογών (Νορβηγία).

Τα σύγχρονα συστήματα ηλεκτρονικής ψηφοφορίας [25, 32, 84, 29, 2, 12, 99, 96] ικανοποιούν την ιδιότητα της *άμεσης επαληθευσιμότητας* (end-to-end verifiability), σύμφωνα με την οποία ο ψηφοφόρος έχει τη δυνατότητα να επαληθεύσει την ορθή καταγραφή της ψήφου του καθώς και ολόκληρης εκλογικής διαδικασίας χωρίς να χρειάζεται να εναποθέτει εμπιστοσύνη σε καμία εκλογική αρχή. Σημειώνεται ότι η άμεση επαληθευσιμότητα είναι πολύ δύσκολο να επιτευχθεί στα πλαίσια των παραδοσιακών εκλογών, όπου η ακεραιότητα του αποτελέσματος υποθέτει ότι η εγγενής τιμιότητα ή η συγκρουση συμφερόντων εξασφαλίζει την αδιάβλητη εκτέλεση καθηκόντων από πλευράς εκλογικών αρχών.

Η παρούσα διατριβή μελετά το αντικείμενο της ηλεκτρονικής ψηφοφορίας από τη σκοπιά της σύγχρονης κρυπτογραφίας. Ειδικότερα, αποσκοπεί κυρίως στην απάντηση ενός ανοιχτού ερωτήματος που αφορά τη βέλτιστη εφικτή άμεση επαληθευσιμότητα. Προγενέστερα της παρούσας διατριβής, όλα τα κορυφαία συστήματα ηλεκτρονικής ψηφοφορίας [25, 64, 32, 84, 29, 2, 12] προϋπέθεταν εμπιστοσύνη στις συσκευές ψηψοφορίας, το μοντέλο τυχαίου μαντείου (random oracle model), ή την ύπαρξη μίας έμπιστης πηγής τυχαιότητας (randomness beacon) για την εξασφάλιση της άμεσης επαληθευσιμότητας. Ωστόσο, απέναντι σε αντιπάλους που ελέγχουν πλήρως όλες τις αρχές και συσκευές ψηφοφορίας και στα πλαίσια του *standard* μοντέλου, όπου δεν υποτίθεται καμία αδιάβλητη εξωτερική πηγή τυχαιότητας ή οποιασδήποτε άλλης έμπιστης παραμέτρου στο εκλογικό περιβάλλον, τα παραπάνω συστήματα αποτυγχάνουν να προστατέψουν την ψήφο ενός έντιμου ψηφοφόρου.

Η παρούσα διατριβή απαντά καταφατικά στο ανοιχτό ερώτημα ύπαρξης συστήματος ηλεκτρονικής ψηφοφορίας με *άμεση επαληθευσιμότητα στο standard μοντέλο* με την εισαγωγή

της οικογένειας συστημάτων ηλεκτρονικής ψηφοφορίας DEMOS, η οποία αποτελεί την κύρια συνεισφορά της διατριβής και περιλαμβάνει τις εξής τρεις πτυχές:

I. Την κατασκευή των συστημάτων *DEMOS-A* και *DEMOS-2* τα οποία αποτελούν τα αρχικά μέλη της οικογένειας DEMOS. Τα δύο συστήματα ηλεκτρονικής ψηφοφορίας ακολουθούν διαφορετικές κατευθύνσεις, μοιράζονται όμως το ίδιο κοινό γνώρισμα: έναν νέο μηχανισμό εξαγωγής τυχαιότητας που είναι αναγκαία για την επαλήθευση από την εντροπία που παράγεται από τη διάδραση των ψηφοφόρων με το σύστημα. Η προκείμενη εντροπία είναι εσωτερική ως προς το περιβάλλον ψηφοφορίας, εξαλείφοντας έτσι την ανάγκη ύπαρξης μιας εξωτερικής πηγής τυχαιότοτητας. Ως αποτέλεσμα, τα συστήματα DEMOS-A και DEMOS-2 επιτυγχάνουν άμεση επαληθευσιμότητα στο standard μοντέλο υποθέτοντας μόνο την ύπαρξη ενός αδιάβλητου δημόσια προσβάσιμου *πίνακα επαλήθευσης* (bulletin board), μία προϋπόθεση η οποία μπορεί να δειχθεί ότι είναι ανελαστική για την εν λόγω ιδιότητα.

II. Τη διαμόρφωση ενός αυστηρού κρυπτογραφικού πλαισίου για την ανάλυση ασφάλειας συστημάτων ηλεκτρονικής ψηφοφορίας, το οποίο περιλαμβάνει ορισμούς της άμεσης επαληθευσιμότητας καθώς και της *ιδιωτικότητας* (privacy), όπου η τελευταία ιδιότητα συμπεριλαμβάνει την *αντίσταση στον καταναγκασμό* (coercion resistance) απέναντι σε *παθητικούς αντιπάλους* (passive adversaries). Τα συστήματα DEMOS-A και DEMOS-2 αποδεικνύονται ασφαλή στο παραπάνω πλαίσιο.

III. Την επέκταση του προαναφερθέντος πλαισίου ασφάλειας για την αυστηρή μοντελοποίηση της ανθρώπινης συμπεριφοράς σε μία εκλογική διαδικασία, έχοντας ως αρχικό ερέθισμα την ισχυρή εξάρτηση της ασφάλειας των DEMOS-A και DEMOS-2 από την ενεργή συμμετοχή των έντιμων ψηφοφόρων στη διαδικασία επαλήθευσης. Η επέκταση του πλαίσιου βασίζεται στην ανάλυση δομής ενός δικτυακού πρωτοκόλλου ως *ceremony*, όπως διατυπώθηκε από τον Ellison το 2007 [44]. Ως υπόδειγμα μελέτης του επεκτεταμένου μοντέλου, η παρούσα διατριβή αναλύει σε βάθος την ασφάλεια του ευρέως εφαρμοσμένου συστήματος ηλεκτρονικής ψηφοφορίας Helios, μέσα από θεωρητική αλλά και πειραματική προσέγγιση.

2. Σχετική βιβλιογραφία

2.1 Συστήματα ηλεκτρονικής ψηφοφορίας

Έως σήμερα, μία μακριά λίστα από συστήματα ηλεκτρονικής ψηφοφορίας (ενδεικτικά [23, 37, 24, 47, 41, 25, 62, 65, 32, 67, 94, 29, 2, 36, 84, 50, 12, 96, 99]) με ποικίλα χαρακτηριστικά, τα οποία μπορούν να κατηγοριοποιηθούν σύμφωνα με δύο θεμελιώδεις τρόπους:

I. Σε συστήματα *επί τόπου ψηφοφορίας* (on-site voting) [12, 32, 94, 29, 84] ή *απομακρυσμένης ψηφοφορίας* [25, 99, 23, 37, 24, 47, 41, 62, 65, 67, 2, 36, 50, 96] (remote e-voting/i-voting) ανάλογα με τον τρόπο πρόσβασης των ψηφοφόρων στο σύστημα.

II. Σε συστήματα *client-side κρυπτογράφησης* (client-side encryption) [12, 23, 37, 24, 47, 41, 62, 65, 67, 2, 36, 50, 96] ή *ψηφοφορίας βάσει κωδικών* (code-voting) [32, 94, 29, 84, 25, 99] ανάλογα με τη μέθοδο υποβολής ψήφου.

2.2 Μοντελοποίηση ασφάλειας ηλεκτρονικής ψηφοφορίας

Προγένεστεροι ορισμοί της επαληθευσιμότητας έχουν δοθεί στα [23, 89, 64, 69, 33]. Η άμεση επαληθευσιμότητα με τη μορφή που είναι αντιληπτή σήμερα, είναι αποτέλεσμα των εργασιών των Chaum [26] και Neff [80]. Αυστηροί κρυπτογραφικοί ορισμοί της άμεσης επαληθευσιμότητας πέραν της πάρουσας διατριβής έχουν προταθεί από τους Küsters, Truderung και Vogt [70] καθώς και τους Smyth, Frink και Clarkson [95].

Ορισμοί της ιδιωτικότητας και αντίστασης στον καταναγκασμό έχουν δοθεί στα [37, 33, 57, 42, 72, 14, 15] και [13, 78, 42, 57]. Σημειώνεται ότι η παρούσα διατριβή περιορίζεται στη μελέτη της αντίστασης στον καταναγκασμό απέναντι σε παθητικούς αντιπάλους, επομένως σκόπιμα παρακάμπτει τον ορισμό της πληρούς αντίστασης στον καταναγκασμό (full coercion resistance) για τον οποίο ο αναγνώστης παραπέμπεται στα [64, 42, 97, 71, 4].

3. Ορισμοί και Εργαλεία

Σε αυτήν την ενότητα, αναφέρουμε τις μαθηματικές και κρυπτογραφικές έννοιες που μορφοποιούν το υπόβαθρο της διατριβής χωρίς να εμμένουμε σε αυστηρά πλαίσια διατύπωσης λόγω της συνοπτικής φύσης του κειμένου.

Ορισμοί.

Μία συνάρτηση είναι *αμελητέα* (negligible) εάν είναι ασυμπτωτικά μικρότερη από τον αντίστροφο οποιουδήποτε πολυωνύμου. Δύο τυχαίες μεταβλητές X, Y είναι *μη διακρίσιμες* (indistinguishable) εάν κάθε πιθανοτικός αλγόριθμος πολυωνυμικού χρόνου (PPT) αποφασίζει (επιστρέφει 1 ή 0) με αμελητέα διαφορά όταν δέχεται εισόδους οι οποίες ακολουθούν είτε την κατανομή της X είτε της Y. Χρησιμοποιούμε το λ για να συμβολίσουμε την παράμετρο ασφάλειας.

Εργαλεία.

- Ένα *σχήμα δέσμευσης* (commitment scheme) είναι μία τριάδα αλγορίθμων που αποτελείται από (i) έναν αλγόριθμο Gen που με είσοδο 1^λ επιστρέφει ένα κλειδί ck, (ii) έναν αλγόριμο Com ο οποίος με είσοδο ck, ένα μήνυμα M και string r επιστρέφει ένα commitment c στο M, και (iii) έναν αλγόριθμο $Verify$ ο οποίος με είσοδο ck, c και ένα opening (M, r) αποφασίζει accept ή reject. Ένα σχήμα δέσμευσης υλοποιεί την έννοια ενός *ηλεκτρονικού φακέλου*, δηλαδή ένα commitment c στο M πρέπει (a) να μην μπορεί να ανοιχτεί για ένα άλλο μήνυμα M' (binding ιδιότητα) και επίσης (b) να μην αποκαλύπτει κάποια πληροφορία για το M χωρίς τη χρήση του opening (M, r) (hiding ιδιότητα). Για τις ανάγκες της καταμέτρησης ψήφων με μυστικότητα, χρησιμοποιούμε το κρυπτοσύστημα ElGamal [49] ως σχήμα δέσμευσης με την επιπλέον *ομομορφική προσθετική* (homomorphic additivity) ιδιότητα

$$Com(\mathsf{ck}, M_1) \cdot Com(\mathsf{ck}, M_2) = Com(\mathsf{ck}, M_1 + M_2) \ .$$

- Μία *απόδειξη μηδενικής γνώσης* (zero-knowledge (ZK) proof) είναι ζεύγος δύο διαλογικών μηχανών Turing που αποτελείται από έναν *αποδείκτη* (prover) $\mathcal{P}$ ο οποίος

επιχειρεί να πείσει έναν PPT επαληθευτή (verifier) $\mathcal{V}$ για την αλήθεια ενός ισχυρισμού $x \in \mathcal{L}$, όπου $\mathcal{L}$ είναι μία γλώσσα στο **NP**. Μία ZK απόδειξη πληροί τις εξής ιδιότητες:

I. *Πληρότητα* (Completeness): ο $\mathcal{V}$ πάντα αποδέχεται την απόδειξη του $\mathcal{P}$ για κάθε ισχυρισμό $x \in \mathcal{L}$.

II. *Ορθότητα* (Soundness): ο $\mathcal{V}$ δεν αποδέχεται την απόδειξη ενός πιθανώς κακόβουλου αποδείκτη $\mathcal{P}^*$ για οποιόδηποτε $x \notin \mathcal{L}$, εκτός από αμελητέα πιθανότητα.

III. *Μηδενική γνώση* (Zero-Knowledge): ο $\mathcal{V}$ δεν μαθαίνει τίποτα πέραν της αλήθειας του ισχυρισμού $x \in \mathcal{L}$, μέσω της διάδρασής του με τον $\mathcal{P}$.

Στην παρούσα διατριβή, χρησιμοποιούνται δύο ειδικές κατηγορίες ZK αποδείξεων: (i) Τα *Σ-πρωτόκολλα*, όπου ή διάδραση περιλαμβάνει τρεις γύρους και τα νομίσματα του $\mathcal{V}$ είναι δημόσια και (ii) τις *μη διαλογικές* ZK αποδείξεις [NIZK proofs], όπου η διάδραση αποτελείται μόνο από έναν γύρο, εν προκειμένω τη δημιουργία και την αποστολή της απόδειξης από τον $\mathcal{P}$ στον $\mathcal{V}$.

- Τα *κατανεμημένα κρυπτοσυστήματα δημόσιου κλειδιού* [threshold public key encryption (TPKE) cryptosystems], όπου πολλαπλοί servers συνεργατικά παράγουν το δημόσιο κλειδί κρυπτογράφησης, ενώ για την αποκρυπτόγραφηση ενός κρυπτοκειμένου (ciphertext) κάθε server συνεισφέρει ένα απαραίτητο μερίδιο αποκρυπτόγραφησης. Στη βιβλιογραφία της ηλεκτρονικής ψηφοφορίας, τυπικά χρησιμοποιείται το TPKE σύστημα ElGamal [83].

4. Το κρυπτογραφικό πλαίσιο

4.1. Οντότητες και σύνταξη

Οι οντότητες που εμπλέκονται σε μια διαδικασία ηλεκτρονικής ψηφοφορίας είναι (i) η Εκλογική Αρχή [Election Authority (EA)] η οποία είναι υπεύθυνη για την εκλογική προετοιμασία, (ii) η Ψηφιακή Κάλπη [Vote Collector (VC)] όπου υποβάλλονται οι ψήφοι, (iii) οι n το πλήθος ψηφοφόροι $V_1, \ldots V_n$, (iv) οι συσκευές ψηφοφορίας, (v) οι k το πλήθος Έφοροι [Trustees] $T_1, \ldots, T_k$ υπεύθυνοι για την καταμέτρηση των ψήφων και την ανακοίνωση των εκλογικών αποτελεσμάτων και (vi) ένας δημόσια προσβάσιμος πίνακας επαλήθευσης [Bulletin Board (BB)]. Οι m το πλήθος εκλογικές επιλογές συμβολίζονται με $\text{opt}_1, \ldots, \text{opt}_m$

Ένα σύστημα ηλεκτρονικής ψηφοφορίας απαρτίζεται από τα πρωτόκολλα (i) προετοιμασίας **Setup**, (ii) ψηφοφορίας **Cast**, (iii) καταμέτρησης **Tally** και τους αλγορίθμους (iv) υπολογισμού αποτελεσμάτων **Result** και (v) επαλήθευσης **Verify**.

4.2. Ορισμός άμεσης επαληθευσιμότητας

Για τον ορισμό της άμεσης επαληθευσιμότητας χρειάζεται πρωτίστως να εξηγήσουμε ρητά το στόχο του αντιπάλου και το μέγεθος που αυτός επιτυγχάνεται, δηλαδή την απόκλιση που ο αντίπαλος προκαλεί από το προτιθέμενο αποτέλεσμα. Για το λόγο αυτό, κωδικούμε τα αποτελέσματα ως διανύσματα m ακεραίων και μετράμε τη διαφορά τους μέσω της ℓ_1

νόρμας στον $\mathbb{R}^m$ (πολλαπλασιασμένη κατά το ήμισυ). Ως σημείο αναφοράς των κακόβουλων ψήφων μέσα στο προτιθέμενο αποτέλεσμα, υπόθετουμε ένα *εξαγωγέα ψήφων* (vote extractor) $\mathcal{E}$, όχι αναγκαστικά πολυωνυμικού χρόνου, ο οποιός ερμηνεύει το μέρος του εκλογικού αποτελέσματος από τις ψήφους που διαχειρίζεται ο αντίπαλος.

Η άμεση επαληθευσιμότητα ορίζεται μέσω ενός παιγνίου $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$ ανάμεσα σε ένα challenger και τον αντίπαλο $\mathcal{A}$, παραμετροποιημένου από δύο μεγέθη δ, θ και τον $\mathcal{E}$. Ο αντίπαλος ελέγχει όλες τις εκλογικές αρχές, όλες τις συσκευές ψηφοφορίας και ένα μέρος των ψηφοφόρων. Από την άλλη, ο challenger παίζει το ρόλο των έντιμων ψηφοφόρων ενώ ο BB θεωρείται αδιάβλητος. Ο αντίπαλος κερδίζει το $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$ εάν παρόλο που επιτρέπει σε τουλάχιστον θ έντιμους ψηφοφόρους να ψηφίσουν και να επαληθεύσουν τη διαδικασία, ένα από τα ακόλουθα ισχύει: (a) ο $\mathcal{A}$ επιτυγχάνει απόκλιση εκλογικού αποτελέσματος τουλάχιστον δ ή (b) ο $\mathcal{E}$ αποτυγχάνει να ερμηνεύσει το αποτέλεσμα των κακόβουλων ψήφων.

Βάσει των παραπάνω, η άμεση επαληθευσιμότητα ορίζεται ως εξής:

Ορισμός. *Ένα σύστημα ηλεκτρονικής ψηφοφορίας με m εκλογικές επιλογές, n ψηφοφόρους και k Εφόρους επιτυγχάνει άμεση επαληθευσιμότητα για θ έντιμους ψηφοφόρους και απόκλιση εκλογικού αποτελέσματος δ με σφάλμα ϵ, εάν υπάρχει εξαγωγέας ψήφων $\mathcal{E}$ έτσι ώστε κάθε αντίπαλος $\mathcal{A}$ να μπορεί να κερδίσει το παίγνιο $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$ με το πολύ ϵ πιθανότητα.*

4.3. Το μοντέλο ιδιωτικότητας/ αντίστασης στον καταναγκασμό

Ο αντίπαλος απέναντι στην ιδιωτικότητα/ αντίσταση στον καταναγκασμό ενός συστήματος της ηλεκτρονικής ψηφοφορίας, μπορεί να ελέγχει την VC όλους πλην ενός εφόρους να παρακολουθεί την κίνηση στο δίκτυο και να ζητήσει όλα τα δεδομένα που αποκτούν οι ψηφοφόροι για την επαλήθευση της ψήφου τους. Το τελευταίο συνεπάγεται ότι οι ψηφοφόροι θα πρέπει να έχουν τη δυνατότητα να ξεγελάσουν τον αντίπαλο παρουσιάζοντας ψευδή αλλά συγχρόνως αληθοφανή δεδομένα επαλήθευσης, το οποίο τυπικά μοντελοποιείται μέσω της ύπαρξης ενός *αλγορίθμου προσομοίωσης* της διάδρασης των ψηφοφόρων με το σύστημα.

5. Το σύστημα DEMOS-A

5.1. Συνοπτική περιγραφή του DEMOS-A

Το σύστημα DEMOS-A απαρτίζεται από τα εξής πρωτόκολλα και αλγορίθμους.

Κατά το **Setup** πρωτόκολλο, η ΕΑ ετοιμάζει και διανέμει σε κάθε ψηφόφορο ένα ψηφοδέλτιο με την ακόλουθη δομή: το ψηφοδέλτιο αποτελείται από δύο ισοδύναμες όψεις, όπου σε κάθε όψη τυχαίοι κωδικοί αντιστοιχίζονται στις εκλογικές επιλογές. Στη συνέχεια, η ΕΑ δεσμεύεται στην πληροφορία που περιέχεται σε κάθε ψηφοδέλτιο, αναρτώντας τη κρυμμένη σε ηλεκτρονικούς φακέλους (commitments) σε σωστή αντιστοιχία στον BB. Επίσης, εκτελεί τον πρώτο γύρο Σ-πρωτοκόλλων για την επαλήθευση της ορθής κωδικοποίησης και ανάρτησης των ψηφοδελτίων. Τέλος, η ΕΑ διαμοιράζει το state της στους k Εφόρους μέσω ενός *γραμμικού σχήματος διαμοίρασης μυστικών* [linear secret sharing

scheme (LSSS)] ώστε να μπορεί να διαγραφεί/καταστραφεί για λόγους ιδιωτικότητας.

Κατά το **Cast** πρωτόκολλο, οι ψηφοφόροι ρίχνουν ένα νομισμα και διαλέγουν τυχαία μία όψη του ψηφοδελτίου τους για να ψηφίσουν και υποβάλλουν στη VC των κωδικό που αντιστοιχεί στην επιλογή τους. Ως δεδομένα επαλήθευσης, κρατούν τον υποβεβλημένο κωδικό και την άλλη όψη του φηφοδελτίου τους που δε χρησιμοποίησαν για να ψηφίσουν.

Κατά το **Tally** πρωτόκολλο, οι Έφοροι συλλογικά ανοίγουν τα commitments των κωδικών ώστε η VC να σημειώσει τους ηλεκτρονικούς φακέλους των κωδικοποήμενοων επιλογών που αντιστοιχούν σε υποβεβλημένους κωδικούς. Οι Έφοροι υπολογίζουν ομομορφικά το εκλογικό αποτέλεσμα πολλαπλασιάζοντας τους παραπάνω φακέλους και ανακοινώνουν το εκλογικό αποτέλεσμα (σε κωδικοποιημένη μορφή) ως το άνοιγμα στο άθροισμα των ψήφων των φακέλων. Στη συνέχεια, ανοίγουν όλους τους φακέλους που αντιστοιχούν σε πληροφορία όψεων ψηφοδελτίων που δε χρησιμοποίηθηκαν προς επαλήθευση. Τέλος, αναρτούν τον τρίτο γύρο των Σ-πρωτοκόλλων ορθότητας ψηφοδελτίων, όπου ο δεύτερος γύρος (νομίσματα του επαληθευτή) προκύπτει από τα νομίσματα που συλλογικά συνεισφέρουν οι ψηφόφοροι κατά το **Cast** πρωτόκολλο.

Ο αλγόριθμος **Result** είναι η αποκωδικοποίηση του εκλογικού αποτελέσματατος και μπορεί να εκτελεστεί από οποιαδήποτε οντότητα.

Κάθε ψηφοφόρος με τα προσωπικά του δεδομένα επαλήθευσης εκτελεί τον αλγόριθμο **Verify** ελέγχοντας (1) την συνέπεια των κρυπτογραφικών δεδομένων στον BB, (2) ότι ο κωδικός που υπέβαλαν είναι σημειωμένος προς καταμέτρηση και (3) ότι η αντιστοιχία κωδικών και επιλογών στην όψη που δε χρησιμοποίησαν να ψηφίσουν είναι όμοια με αυτή που έχει ανοιχθεί στο BB.

5.2. Ασφάλεια του DEMOS-A

1. Το DEMOS-A επιτυγχάνει άμεση επαληθευσιμότητα για θ έντιμους ψηφοφόρους και απόκλιση εκλογικού αποτελέσματος δ με σφάλμα $2^{-\theta+\lceil n/\lfloor \log q \rfloor \rceil (\log\log m+1)} + 2^{-\delta}$, όπου q είναι το μέγεθος της αλγεβρικής ομάδας ElGamal. Ο όρος $2^{-\delta}$ προκύπτει από το γεγονός ότι αν όλα τα Σ-πρωτόκολλα έχουν παραχθεί έντιμα, τότε ο μηχανισμός επαλήθευσης του DEMOS-A συνεπάγεται ότι για κάθε αύξηση απόκλισης κατά 1 η πιθανότητα επιτυχίας αντιπάλου ελαττώνεται κατά 1/2. Ο όρος $2^{-\theta+\lceil n/\lfloor \log q \rfloor \rceil (\log\log m+1)}$ είναι το σφάλμα ορθότητας των Σ-πρωτοκόλλων.

2. Το DEMOS-A επιτυγχάνει ιδιωτικότητα/ αντίσταση στον καταναγκασμό, εάν το πρόβλημα Decisional Diffie-Hellman ορισμένο στην αλγεβρική ομάδα ElGamal δε μπορεί να επιλυθεί σε υποεκθετικό χρόνο.

6. Το σύστημα DEMOS-2

Όπως κάθε σύστημα όπου βασίζεται στη χρήση κωδικών, το DEMOS-A έχει το πλεονέκτημα ότι η ψηφοφορία είναι πολύ απλή διαδικασία από πλευράς ψηφοφόρου χωρίς να απαιτούνται υπολογιστικά ισχυρές συσκευές ψηφοφορίας. Για να επιτευχθεί αυτό όμως, προηγείται ένα πολύπλοκο προπαρασκευαστικό βήμα από πλευράς ΕΑ, το οποίο καθιστά μάλλον απαγορευτική την επεκτασιμότητα του DEMOS-A σε εθνικές εκλογές. Το ζήτημα επιλύεται στο σύστημα DEMOS-2 όπου ακολουθεί την clien-side encryption λογική, δηλαδή οι ψηφοφόροι τώρα διαθέτουν υπολογιστικά ισχυρές, πλην όμως έμπιστες από πλευράς ιδιωτικότητας συσκευές, για την τοπική κρυπτογράφηση υποβολής των ψήφων μέσω ενός ΤΡΚΕ σχήματος. Η ΕΑ τώρα απλώς ζητείται να αποδείξει μέσω ενός Σ-πρωτοκόλλου την ορθότητα του περιβάλλοντος εκλογικής διαδικασίας, ενώ οι ψηφοφόροι καλούνται να ρίξουν ένα νόμισμα και ένα επιλέξουν μία από δύο ισοδύναμες κρυπτογραφήσεις της ψήφου τους, κατά αναλογία με την επιλογή όψεων στο DEMOS-A. Με αυτό τον τρόπο το DEMOS-2 επιτυγχάνει επίσης άμεση επαληθευσιμότητα στο standard μοντέλο (με σφάλμα τώρα $2^{-\theta}+2^{-\delta}$), εξάγοντας την απαιτούμενη τυχαιότητα από τα νομίσματα των ψηφοφόρων.

7. Το Helios ως ceremony ηλεκτρονικής ψηφοφορίας

Τα θεωρήματα άμεσης επαληθευσιμότητας των DEMOS-A και DEMOS-2 αποτυπώνουν μαθηματικά την εξάρτηση της ασφάλειας από τη συμμετοχή των έντιμων ψηφοφόρων στην επαλήθευση, το οποίο όπως προαναφέρθηκε, αποτέλεσε και το κίνητρο για την επέκταση του πλαισίου ασφάλειας στο ceremony μοντέλο. Στην παρούσα διατριβή, επιλέχθηκε το Helios [2] ως υπόδειγμα μελέτης του ceremony μοντέλου, λόγω (i) του σχεδιασμού του που αφήνει σημαντική ελευθερία στον ανθρώπινο παράγοντα και (ii) της δημοτικότητας του ως ένα αναγνωρισμένο αξιόπιστο σύστημα. Ειδικότερα, αναλύθηκε η ασφάλεια του Helios για διάφορες κατανομές ανθρώπινων συμπεριφορών σε θεωρητικό επίπεδο και αποτιμήθηκε η ασφάλεια του σε πειραματικό επίπεδο μέσω δεδομένω από πραγματικές εκλογές που χρησιμοποίησαν το Helios. Τα αποτελέσματα ήταν μαλλον αποθαρρυντικά, καθώς αποδεικνύεται ότι ακόμη και ένα εξαιρετικά καταρτισμένο εκλογικό σώμα όπως τα μέλη του Διεθνούς Συλλόγου Κρυπτογραφικής Έρευνας (IACR) εμφανίζουν τρωτά σημεία ως ψηφοφόροι, επιτρέποντας σε ένα επιτεθέμενο να ανατρέψει με σημαντική πιθανότητα προς όφελος του το εκλογικό αποτέλεσμα, όταν οι διαφορές μεταξυ των ποσοστών των υποψηφίων δεν είναι ιδιαίτερα μεγάλες (της τάξης μέχρι 5%).

8. Συμπεράσματα και μελλοντικές κατεθύνσεις

Τα συστήματα DEMOS-A και DEMOS-2 θέτουν ψηλά τον πήχη ως προς τις απαιτήσεις ασφάλειας των ηλεκτρονικών εκλογών. Το πλήρες κρυπτογραφικό πλαίσιο που προτείνεται αποτελεί βάση για μια ωριμότερη μαθηματικά ανάλυση ασφάλειας των συστημάτων ηλεκτρονικής αλλά και παραδοσιακής ψηφοφορίας. Εντούτοις, ενδιαφέροντα τεχνικά ζητήματα εξακολουθούν να παραμένουν προς διερεύνηση. Ενδεικτικά αναφέρουμε τα παρακάτω:

- Κατασκευή ενός νέου μελους της οικογένειας DEMOS το οποίο εκτός από άμεση

επαληθευσιμότητα και ιδιωτικότητα, επιτυγχάνει αντίσταση στον καταναγκασμό απέναντι σε όλους (και όχι μόνο παθητικούς) αντιπάλους.

- Ενίσχυση των DEMOS-A και DEMOS-2 με μηχανισμούς *λογοδοσίας* [accountability].
- Μελέτη των πιθανών σεναρίων βέλτιστης ασφάλειας συστημάτων ψηφοφορίας, δεδομένων των αντικρουμένων τάσεων προς ύψιστη ακεραιότητα και προς ύψιστη μυστικότητα.
- Υπολογιστικά ταχύτερους αλγόριθμους για την υλοποίηση του DEMOS-A με σκοπό την εφαρμογή του στη μέγιστη δυνατή κλίμακα.

ACKNOWLEDGEMENTS

I am most grateful to my supervisor, Prof. Aggelos Kiayias, for entrusting me and selecting me as a member of the CRYPTO.SEC research group. Under his guidance, I had the opportunity to evolve as a scientist and as a person.

I would like to give special mention to Dr. Bingsheng Zhang, who along with Prof. Aggelos Kiayias and myself, formed a strong team producing all the noted research results that constitute the technical aspect of this thesis. In addition, I wish to thank all the renowned scientists that have been my coauthors in my publications so far.

I strongly believe that my PhD years would not have been so productive without the support of my colleagues at the CRYPTO.SEC group, both at an intellectual and a personal level. Throughout those years, we built a healthy working environment that any researcher would desire.

Finally, I wish to thank all the members of the advisory and examination committees for their valuable comments, that have been proven really helpful for the completion of this thesis.

LIST OF PUBLICATIONS

[1] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 468–498. Springer, 2015.

[2] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. DEMOS-2: scalable E2E verifiable elections without random oracles. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 352–363. ACM, 2015.

[3] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. Ceremonies for end-to-end verifiable elections. *IACR Cryptology ePrint Archive*, 2015:1166, 2015.

[4] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. On the necessity of auditing for election privacy in e-voting systems. In Sokratis K. Katsikas and Alexander B. Sideridis, editors, *E-Democracy - Citizen Rights in the World of the New Computing Paradigms - 6th International Conference, E-Democracy 2015, Athens, Greece, December 10-11, 2015, Proceedings*, volume 570 of *Communications in Computer and Information Science*, pages 3–17. Springer, 2015.

[5] Nikos Chondros, Bingsheng Zhang, Thomas Zacharias, Panos Diamantopoulos, Stathis Maneas, Christos Patsonakis, Alex Delis, Aggelos Kiayias, and Mema Roussopoulos. D-DEMOS: A distributed, end-to-end verifiable, internet voting system. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*, pages 711–720. IEEE Computer Society, 2016.

[6] Alex Delis, Konstantina Gavatha, Aggelos Kiayias, Charalampos Koutalakis, Elias Nikolakopoulos, Lampros Paschos, Mema Roussopoulos, Georgios Sotirellis, Panos Stathopoulos, Pavlos Vasilopoulos, Thomas Zacharias, and Bingsheng Zhang. Pressing the button for european elections: verifiable e-voting and public attitudes toward internet voting in greece. In Robert Krimmer and Melanie Volkamer, editors, *6th International Conference on Electronic Voting: Verifying the Vote, EVOTE 2014, Lochau / Bregenz, Austria, October 29-31, 2014*, pages 1–8. IEEE, 2014.

[7] Foteini Baldimtsi, Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. Indistinguishable proofs of work or knowledge. *IACR Cryptology ePrint Archive*, 2015:1230, 2015.

[8] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 669–684. ACM, 2013.

[9] Nikos Chondros, Alex Delis, Dina Gavatha, Aggelos Kiayias, Charalampos Koutalakis, Ilias Nicolacopoulos, Lampros Paschos, Mema Roussopoulos, Giorge Sotirelis, Panos Stathopoulos, Pavlos Vasilopoulos, Thomas Zacharias, Bingsheng Zhang, and Fotis Zygoulis. Electronic voting systems - from theory to implementation. In Alexander B. Sideridis, Zoe Kardasiadou, Constantine P. Yialouris, and Vasilios Zorkadis, editors, *E-Democracy, Security, Privacy and Trust in a Digital World - 5th International Conference, E-Democracy 2013, Athens, Greece, December 5-6, 2013, Revised Selected Papers*, volume 441 of *Communications in Computer and Information Science*, pages 113–122. Springer, 2013.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

The findings in this PhD thesis reflect the author's contribution to the goals of the interdisciplinary FINER e-voting project, which research team consisted of lawyers, political political scientists, distributed systems experts, and, of course, cryptographers. The FINER project took place from March 2013 till September 2015 and aimed at the construction of a fully functional and provably secure e-voting system that could be applied at a national scale without single point failure. The project run in six stages as follows:

1. A complete study of the fundamental requirements that the e-voting system should satisfy and the level that these requirements are fulfilled in traditional elections.
2. Introduction of a formal framework for the security analysis of the e-voting system providing mathematical definitions of the aforementioned requirements.
3. Design of the e-voting system, encompassing all necessary cryptographic algorithms and protocols and the explicit description of the human protocol.
4. Formal analysis of the security of the e-voting system under the above framework.
5. Implementation of the e-voting system in a fully distributed setting.
6. Real-world evaluation of the system's usability via pilot experiments.

The FINER project was successfully completed with the introduction of the DEMOS-A, DEMOS-2 and D-DEMOS e-voting systems that achieve the highest level of integrity to date while preserving the standard e-voting privacy requirements. The author had a leading role in the completion of stages **2,3** and **4** and supported the execution of stages **1** and **6**. The implementation of DEMOS-A has been tested in two pilot polls executed at the European Elections in May 2014 (747 participants) and the National Elections in January 2015 (400 participants). The current version of DEMOS-A is integrated in the electronic platform of the General Confederation of the Workers of Greece.

As a member of the FINER research team, the author had the opportunity to experience all the phases constituting the realisation of a scientific idea, from its birth as a concept till its full deployment, filtered by the views of all involved researchers having various yet complementary technical backgrounds. In this PhD thesis, the author attempted to capture this holistic experience by making the reader aware of the theoretical and sociopolitical motivation that is often hidden behinds the lines of the formal mathematical language text.

1. INTRODUCTION

The evolution of civilisation and the development of tools have been in a strong correlation throughout the course of humanity. An invention that was motivated by an immediate practical need, has often been in long-term the turning point for a new era, in a way that most likely not even the inventor could foresee. The invention of the wheel aimed at the improvement of the craft of pottery, but eventually revolutionised travelling and transportation. Writing was initially utilised for facilitating long-distance trade, resulting as the ultimate means for recording history, generating literature masterpieces and educating the next generations. The steam engine played a critical role for the transition to the Industrial Age, while without radio transmission we would probably never reached the intercultural proximity of today.

From a political perspective, the "invention" widely accepted as human's greatest achievement is democracy. Taking various forms through history, democracy is the only political system to date that provides, directly or indirectly, people with the power to people to control the fate of their state, as indicated by its etymology, from the Greek "*demos*" (people) and "*kratos*" (power). Like any great invention, democracy has a wide impact on civilisation that goes beyond the straight political aspect. Experience has shown that a democratic society is the ideal environment for the preservation of the most noble values, such as freedom of speech, freedom of conscience and equal rights.

Political activity in a modern democratic state comprises compositions of individual democratic procedures. At high level, a democratic procedure consists of three well-defined parts :

1. An *electorate* formed by the people legitimate to vote,
2. A *voting system*, which serves as means to record and evaluate the electorate's will, and
3. A *verdict*, which derives from the consensus according to the evaluated electorate's will.

The dependence among democratic procedures can be rather complicated. For instance, specifying the electorate or the voting-system for a direct democratic procedure (e.g. national elections) may derive by the verdict of an indirect procedure (e.g. legislation), which in turn can be executed with respect to the rules decided by a root procedure (e.g. constitutional amendment).

As given away by its title, this PhD thesis concentrates on the concept of voting systems. If democracy is the political system that brings power closest possible to people, then a voting system is certainly the channel to accomplish this link. A modern voting system must incorporate mechanisms for optimising accessibility of the electorate and guarantee integrity of the election result while protecting the voters' secrecy. If it manages so, then it paves the way for establishing a politically healthy democratic society. On the other hand,

due to their crucial role in democracy, voting systems have often been top priority targets for attackers that wish to tamper the election result and/or coerce voters to vote against their intention. Voting systems that allow people to sell their votes, or lack verification procedures that convince an auditor of the validity of the election result with minimum doubt, undermine the foundations of any democratic state they are deployed.

Most interestingly, in voting literature, it has been observed that integrity and secrecy are requirements in an inherently contradictory relation with respect to voting. Indeed, if one does not give weight to privacy, then open ballot voting where the voters are publicly associated with their votes supports maximum assurance of tally integrity. Conversely, if people put unreserved trust in the election authorities responsible for collecting and tallying the votes, then no thorough auditing is necessary, so it is easier for a voting system to conceal the voters' intention. One can think of settings when solely verifiable integrity or privacy is strongly desired. For example, legislation procedures are run via open ballot election in many countries' Parliaments, whereas in small scale board elections, vote collection tally may be done collectively in the presence of all participants, so protecting secrecy is the only demanding goal. However, in most cases, a voting system designer is facing conflicting challenges on the road to achieve the best of the integrity and secrecy trends.

From the viewpoint of contemporary cryptography, this PhD thesis investigates the feasibility of fully functional voting combined with top-tier security, with respect to the fundamental research question on whether new technologies can be a fruitful ground for the development of reliable voting systems. The latter puts forth the concept of *electronic voting (e-voting)* which is outlined in the following section.

1.1 e-Voting in Democratic Procedures

In an e-voting system, election preparation, vote collection and/or tally is executed by electronic devices, partially or fully managed by human authorities. The motivation for introducing e-voting was originally three-fold; (i) facilitating the participation of social groups with considerable physical barriers, (ii) reduction of election cost, and (iii) acceleration of the election preparation, vote casting and tally phase. E-voting emerged in the 60s via punch-card systems, followed by systems based on either optical scan voting, ballot encryption, or vote-code typing. By today, e-voting systems have been used in several countries either in pilot executions (Australia, England, Ireland, Italy, Norway) or binding elections at a municipality or national level (Belgium, Brazil, Canada, Estonia, India, the Netherlands, Switzerland, USA).

Based on their infrastructure, e-voting systems are classified into two major categories:

1. *On-site e-voting systems*, where the election is executed in polling stations, and supervision by human authorities is similar to traditional elections.
2. *Remote e-voting (i-voting) systems*, where the voters submit their votes using de-

vices (PCs, notebooks, tablets, smartphones) that have internet access.

In this thesis, we focus on the construction and security analysis of remote e-voting systems. Clearly, any functional and reliable remote e-voting system can be easily adopted to the on-site setting, whereas the opposite does not definitely hold.

E-voting debate:

Involving computers for carrying out democratic procedures enjoys the advantage of boosting efficiency beyond human limitations. Unfortunately, this power raises significant concerns regarding potential security risks; any deviation of the machine from the predetermined voting or tally algorithm may result in a massive alteration of the election result, at a scale which is incomparable to any human error or deliberate dishonest act. Furthermore, hacking into a single on-site voting device could be enough for breaching the privacy of every voter's ballot, while at the remote setting (i-voting) the voter's free will is jeopardised by the possible uncontrollable presence of a coercer. Consequently, both integrity and secrecy can be under major threat, if the e-voting infrastructure is assailable.

The aforementioned concern is escalated when actual cases of security breach are recorded in real world e-voting runs. E-voting scepticism began to grow rapidly after the well-known incident in the 2000 Presidential Elections in Florida, where malformed punch cards for direct-recording electronic (DRE) voting devices, arose suspicions regarding result integrity [81, 76]. In the Netherlands, a number of security defects in the utilised Nepad voting machines initially pointed out by the action group "We don't trust computers" ("Wij vertrouwen stemcomputers niet") [1] caused the abandoning of e-voting instantiated by the RIES e-voting system in 2008 (cf. Section 2.4). Due to the same reason, Ireland (2010) dropped its e-voting project scheduled to run via Nepad machines.

An additional issue that obstructs the spread of e-voting is the people's low confidence on electoral processes that are managed by computers. It is widely believed that trust plays a crucial role for a society embracing the e-voting concept. As prominent example, in Norway, e-voting did not proceed beyond pilot level based on surveys revealing people's fear of losing their privacy[2].

Cryptography and e-voting:

By the above discussion, it should be clear that the introduction of advanced e-voting systems which, along with efficiency, are characterised by undisputed reliability, is essential for infusing trust in the e-voting idea into the public. The way to face this challenge scientifically, is the design of e-voting systems that enable people to actively participate in the election verification procedure and their security is supported by rigorous mathematical proofs. This is where modern cryptography comes as an invaluable tool; the development of efficient cryptographic techniques during the last decades has armed research with encryption and authentication schemes, proof systems, auditing mechanisms, and robust security frameworks that can be applied for the construction of systems able to arguably

[1] URL: `http://wijvertrouwenstemcomputersniet.nl/Wij_vertrouwen_stemcomputers_niet` .

[2] URL: `https://www.regjeringen.no/en/aktuelt/Internet-voting-pilot-to-be-discontinued/id764300/` .

support the adoption of e-voting against any rational debate. In the following section, we recap the formal treatment integrity verification in cryptographic and e-voting literature.

1.2 End-to-end Verifiability and e-Voting

Besides advancing participation and reduction of election cost and time, several state-of-the-art e-voting systems [25, 32, 84, 29, 2, 12, 99, 96] support an attractive and highly non-trivial security feature that traditional voting unavoidably misses by its nature. Namely, the voter can verify that her vote was properly cast, recorded and tallied into the election result without relying to the honesty of any of the election administrators. This strong property is named *end-to-end (E2E) verifiability* and is usually interpreted as the ability of the voter to verify that her vote was (i) cast-as-intended, (ii) recorded-as cast, and (iii) tallied-as-recorded [3]. This understanding of verifiability was an outcome of the works of Chaum [26] and Neff [80], that introduced the generation of receipts which could be used for simple voter verification while achieving privacy. Subsequently, E2E verifiability has been in the center of e-voting security study (cf. Subsection 2.5.1 for related work).

This PhD thesis brings in a strong E2E verifiability definition, according to which the adversary is allowed to control the entire election by corrupting all election administrators, all the voters' clients and a portion of the electorate. As it will be explained in Section 3.3 (cf. Remark 3.2), this definition advances the global verifiability approach in [70] by incorporating tools that allow the explicit specifying of the verifiability goal, which remains elusive in [70].

End-to-end verifiability prior to this thesis:

So far, and till the construction of the DEMOS family of e-voting systems presented in this thesis, E2E verifiability could not be justified without trusted setup assumptions. Under a strong cryptographic definition, E2E verifiability could provenly hold only assuming the existence of a *trusted randomness source* that could be either a function modelled as a random oracle (cf. Subsection 2.3.1) [2, 12, 96], or some randomness beacon [25, 32, 84, 29, 99].

1.3 Client-side encryption vs. code-voting

Up to the present moment, numerous noticeable e-voting systems [23, 37, 24, 47, 41, 25, 62, 65, 32, 67, 94, 29, 2, 36, 84, 50, 12, 96, 99] have been introduced, adding to cryptographic literature novel directions or ameliorating existing techniques. Regarding the vote submission mechanism, e-voting systems are separated into the following two categories:

[3] Prior definitions referring to the weaker notions of *individual* and *universal* verifiability are found in [23, 89, 64, 69, 33].

1. *Client-side encryption e-voting systems*, where the voters use supporting devices to encrypt their votes and submit them to the system using their credentials.

2. *Code-voting e-voting systems*, where the voters receive pre-encoded ballots and submit their votes simply by typing an encoding (e.g. vote-code) that corresponds to their option selection.

The two approaches enjoy complementary benefits and weaknesses. In a client-side encryption e-voting system, knowing a voter's credential does not violate her privacy. Moreover, the cryptographic workload is distributed among the voters' clients, so the system can easily adopt to large scale election settings. On the other hand, a code-voting system has the advantage of supporting election executions under minimum computational requirements from the voters' side, while they preserve privacy even when a voter's client is corrupted, since the encoding of the election options is done in a random fashion. Therefore, whether a client-side cryptography or a code-voting e-voting system should be deployed, depends on the given election specifications.

In Table 1.1, we illustrate the classification of a list of e-voting systems, according to their infrastructure and vote submission method.

Table 1.1: Classification of well-known e-voting systems prior to this thesis.

	Client-side encryption	**Code-voting**
On-site	[12]	[32, 94, 29, 84]
Remote	[23, 37, 24, 47, 41, 62, 65, 67, 2, 36, 50, 96]	[25, 99]

1.4 Objectives and Contributions of this Thesis

The main objective this thesis investigates, is the feasibility of E2E verifiability *in the standard model*, which denotes that verification is executed with assuming the existence of a trusted randomness source. As already mentioned in Section 1.2, until the writing of this thesis, E2E verifiability in an all-malicious setting could provenly hold only under certain setup assumption for randomness.

In order to illustrate why previous techniques did not work, we elaborate on the previous statement. By its design, Helios -and other client-side encryption E2E verifiable systems as [41, 67, 12, 96]- requires the voter to utilise a voter supporting device to prepare a ciphertext and after an indeterminate number of trials, the voter will cast the produced ciphertext. The submitted ciphertexts should be accompanied by a proof of proper computation. While such proofs are easy to construct based on e.g., [40], they can be argued either (i) interactively or (ii) using a *non-interactive zero-knowledge (NIZK) proof* [16]. Interaction is insufficient in E2E verifiability setting since a corrupt election authority together with a corrupt voter may cook up a malformed proof that is indistinguishable from a proper

one. As a result, the non-interactive approach is mandatory. However, NIZK proofs can be sound only under setup assumptions as a *random oracle* or a *common reference string (CRS)* [54]. If the CRS is setup by the election authority, then, in case it is malicious, it will know and exploit the trapdoor; on the other hand, the voters are not interacting with each other and hence cannot setup the CRS by employing a standard multi-party computation protocol [53, 28].

On the other hand, in the case of Remotegrity/Scantegrity -and other vote-code based E2E verifiable systems as [25, 32, 84, 99]- the random coins need to be obtained from the randomness beacon in order to prove the result correct. It is easy to verify that the system is insecure in terms of E2E verifiability in case the randomness beacon is biased. As before, the only parties active are the election authority and the voters who cannot implement a randomness beacon that is required in the construction.

As a consequence of the aforementioned technical restrictions, the following question remained open:

Q1. *Can the integrity of the election result be proven in the standard model, i.e. without believing in trusted hardware, random oracles or randomness beacons?*

This PhD thesis answers this question affirmatively by introducing the *DEMOS-A* and *DEMOS-2* e-voting systems. DEMOS-A is a remote code-voting system that achieves *E2E verifiability in the standard model*, as long as a publicly accessible bulletin board remains consistent. The core idea for this accomplishment is a novel mechanism for extracting randomness from the entropy injected to the system by the voters' entanglement, This entropy is *internal* with respect to the election environment, a fact that removes the requirement for an external randomness source. Furthermore, DEMOS-A preserves voter privacy given the hardness of a standard cryptographic problem (Decisional Diffie-Hellman problem). On the negative side, DEMOS-A does not avoid the weakness inherent in any code-voting system, that is, the difficulty of deploying the system in a large scale election setting due to high computational overhead at the setup phase for the administrator servers' side. In order to resolve this issue, this thesis presents the client-side encryption e-voting system *DEMOS-2* that addresses the scalability limitations of DEMOS-A while still being E2E verifiable in the standard model, at the cost of putting trust in the voting device for privacy.

The second objective studied in this thesis is the effect of the human factor in the security of an E2E verifiable e-voting system. The security analysis of DEMOS-A provides evidence of a strong correlation between the active participation of honest voters in the auditing procedure and the (parameterised) level of E2E verifiability that can be guaranteed. A natural question follows from this observation:

Q2. *At what extent can human behaviour, even within protocol specifications, affect the security of an e-voting system?*

This thesis follows a formal cryptographic direction to deal with this matter. Motivated by the *ceremony framework* introduced by Ellison [44] for the analysis of network protocols, it proposes an extension of standard e-voting security modelling, where human nodes are separated from computer nodes and are formalised as finite state machines (transducers) with limited power, hence incapable of performing cryptographic operations. As a case study of the extended ceremony framework, Helios stands out in terms of the range of possible human behaviour due to (i) the dependence of E2E verifiability on (i.a) the statistics related to the Benaloh audit rate performed by the voters and (i.b) the portion of voters that look up their votes in the bulletin board after election using their ballot trackers and (ii) the dependence of privacy on the trustees auditing the correct uploading of the public key, stemming from the lack of public key infrastructure (PKI) to support authentication of posted data.

In summary, the contributions of this PhD thesis comprise:

1. The introduction of two remote e-voting systems, (i) the vote-coding based DEMOS-A and (ii) the client-side encryption based DEMOS-2 that enrich both major e-voting categories with a member that achieves *E2E verifiability in the standard model* for the first time. The two systems are proven secure under a formal security framework (see below) and their voter privacy/passive coercion resistance holds assuming the hardness of the extensively studied Decisional Diffie-Hellman problem. These two systems give birth to the *DEMOS family of e-voting systems* sharing the attribute of E2E verifiability in the standard model.

2. The introduction of a robust cryptographic framework for the security analysis of e-voting systems. The said framework captures definitions of E2E verifiability, voter privacy and *passive coercion resistance* (often referred as receipt-freeness). The latter property denotes the inability of an e-voting system to allow the voters to prove how they voted or sell their votes, even against an adversary that observes network traffic and requests from the voter the transcript containing their personal view of interaction with the system. The suggested framework is extended to the ceremony model, suitable for the formal study of human behaviour in an election procedure.

3. A thorough analysis of the Helios e-voting system under the ceremony framework. This analysis is threefold consisting of (i) a rigorous mathematical characterisation of classes of voter behaviours that are assailable or resistant to attacks on verifiability, (ii) an evaluation of the expected E2E verifiability guarantee of Helios based on the previous theoretical context given instantiations of real world Helios applications as well as simulation data, and (iii) a presentation of a standard *man-in-the-middle* attack against Helios's privacy, in cases where election guidelines do not encourage trustees (modelled as human nodes) to verify the correct posting of the election public key in the bulletin board.

Challenges not covered in this thesis:

Despite the fact this thesis's subject area covers a significant portion of e-voting cryptographic issues, still important research directions are out-of-scope. For example, full coercion resistance against a coercer that actively affects the voter at election period cannot be achieved by either DEMOS-A or DEMOS-2. For this reason, a formal definition of full coercion resistance is left out of the proposed security framework. Concurrently to the writing of the thesis, an extension of DEMOS-A to full coercion e-voting system is under construction by Kiayias, Teague, Zacharias and Zikas. A high-level idea of the construction is provided in Section 7.2.

Furthermore, this thesis does not deal with the issue of fault tolerance. Therefore, as most existing e-voting systems, DEMOS-A and DEMOS-2 have single points of failure, specifically the election preparation authority, the vote collection authority, and the bulletin board. Adding a fully distributed fault tolerant member to the DEMOS family has been addressed by Chondros *et al.* [35], designers of the *D-DEMOS e-voting system* which is briefly described in in Section 7.1.

1.5 Roadmap

The rest of the thesis is organised as follows:

- In Chapter 2, we introduce the reader to the notation and the mathematical notions and cryptographic primitives that will be the background technical material for this thesis. In addition, we recap some well-known e-voting systems and cite the existing work related to formal security modelling of e-voting systems.
- In Chapter 3, we present our full security framework along with its extension to the ceremony model.
- In Chapter 4, we provide a detailed description and security analysis of the DEMOS-A e-voting system.
- In Chapter 5, we provide a detailed description and security analysis of the DEMOS-2 e-voting system.
- In Chapter 6, we perform a scrupulous case study of Helios, modelled as an e-voting ceremony.
- In Chapter 7, we summarise the D-DEMOS distributed E2E verifiable system and the core ideas in the backbone of the upcoming fully coercion resistant E2E verifiable system by Kiayias, Teague, Zacharias and Zikas.
- In Chapter 8, we conclude the thesis with an overview of its main results, along with directions for future work.

This thesis is structured in modular manner, so that reading after the background chapter can be grouped in almost independent sections. In particular, Sections 3.1, 3.3 3.4 and Chapter 4 suffice for the understanding of DEMOS-A, while Sections 3.1, 3.3 3.5 and Chapter 5 cover the presentation of DEMOS-2. In addition, the reader interested in the ceremony model and Helios's case study may focus on Section 3.6 and Chapter 6.

2. BACKGROUND AND RELATED WORK

In this chapter, we present the background material of this thesis which includes (i) the notation used in our text (cf. Section 2.1), (ii) the definitions of basic mathematical and cryptographic notions (cf. Section 2.2), and (iii) the cryptographic primitives that serve as building blocks for the upcoming e-voting constructions (cf. Section 2.3), presented here in their abstract form. Furthermore, we provide an overview of selected e-voting systems (cf. Section 2.4) and summarise the literature on e-voting security modelling (cf. Section 2.5).

2.1 Notation

The notation that will be used in the thesis is listed in the comprehensive Table 2.1.

Table 2.1: Notation.

NOTATION	MEANING
$\|x\|$	The length of string x.
$\triangleq$	is defined to be equal to
$[n]$	The set of integers $\{1, \ldots, n\}$.
$y \leftarrow \mathcal{A}(x)$	The algorithm $\mathcal{A}$ on input x outputs y.
$y \leftarrow \mathcal{A}(x; r)$	The probabilistic algorithm $\mathcal{A}$ on input x and randomness r outputs y.
$\mathcal{A}^{\mathcal{B}}$	The algorithm $\mathcal{A}$ is given access to the code of $\mathcal{B}$.
poly(x)	polynomial in x
negl(x)	negligible in x
$\langle \mathcal{P}(w), \mathcal{V} \rangle(x, z)$	The prover $\mathcal{P}$ on private input w interacts with the verifier $\mathcal{V}$ for proving statement x using common auxiliary input z.
$x \xleftarrow{\$} S$	x is sampled uniformly at random from the set S.
$x \xleftarrow{\mathbf{D}} S$	x is sampled according to distribution **D** over the set S.
$\Pr_{\mathbf{D}}[E]$	The probability of event E according to distribution **D**.
$\Pr[E]$	The probability of event E (the distribution is implied).
$H_\infty(X)$	min entropy of random variable X

2.2 Basic Definitions

In this section, we recap the definitions of fundamental notions that will be used in the technical context of this thesis.

Definition 2.1 (***Negligible function***)**.** *Let $f : \mathbb{N} \longrightarrow \mathbb{R}$ be a function. We say that f is* negligible*, if it is asymptotically smaller than the inverse of any polynomial. Namely, for every constant c, there is an integer n_c such that (s.t.) for every $n > n_c$, it holds that $f(n) < \frac{1}{n^c}$. We write* negl(n) *to denote that f is negligible in n.*

Definition 2.2 (***Witness relation***)**.** *Let $\mathcal{L}$ be a language in $\mathcal{NP}$. Let $\mathcal{M}$ be the polynomial time (PT) Turing Machine (TM) and let $p(\cdot)$ be the polynomial s.t.*

$$\mathcal{L} = \{x \mid \exists w : |w| \leqslant p(|x|) \text{ and } \mathcal{M}(x, w) = 1\} \ .$$

The witness relation $R_{\mathcal{L}}$ *for $\mathcal{L}$ is defined as*

$$(x, w) \in R_{\mathcal{L}} \Leftrightarrow |y| \leqslant p(|w|) \text{ and } \mathcal{M}(x, w) = 1 \ .$$

We write $R_{\mathcal{L}}(x)$ to denote the set of witness of x, $\{w \mid (x, w) \in R_{\mathcal{L}}\}$.

Definition 2.3 (***Interactive Turing Machine*** [52])**.** *An* interactiveTuringMachine(ITM) *is a TM with a a read-only input tape, a read-only random tape, a read-and-write work tape, a write-only output tape, a pair of communication tapes, and a read-and-write switch tape consisting of a single cell. One communication tape is read-only, and the other is write-only. Each ITM is associated a single bit $b \in 0, 1$, called its identity. An ITM is said to be* active*, in a configuration, if the content of its switch tape equals the machine's identity. Otherwise the machine is said to be* idle*.*

Definition 2.4 (***Min entropy***)**.** *Let X be a random variable (r.v.). We say that X has* minentropy k *and we denote by $H_\infty(X) = k$, if*

$$\max_x \mathsf{Pr}[X = x] = 2^{-k} \ .$$

Equivalently, $H_\infty(X)$ is defined as

$$H_\infty(X) = -\log\big(\max_x \mathsf{Pr}[X = x]\big) \ .$$

Definition 2.5 (***Indistinguishability***)**.** *Let $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be two r.v. ensembles. We say that $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are* computationally *(resp.* statistically*)* indistinguishable*, if for every probabilistic polynomial time (PPT) (resp. unbounded) adversary $\mathcal{A}$ it holds that*

$$\left| \Pr_{x\leftarrow X_\lambda}[\mathcal{A}(x) = 1] - \Pr_{x\leftarrow Y_\lambda}[\mathcal{A}(x) = 1] \right| = \mathsf{negl}(\lambda) \ .$$

Definition 2.6 (***Cryptographic pairing***)**.** *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be multiplicative groups of prime order p and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be a function. We say that e is a* cryptographic pairing *if for every pair of generators g_1, g_2 of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, the following properties are satisfied:*

1. ***Bilinearity****: for every $x, y \in \mathbb{Z}_q$,*

$$e(g_1^x, g_2^y) = e(g_1, g_2)^{xy} \ .$$

2. ***Non-degeneracy****: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity element in $\mathbb{G}_T$.*
3. ***Computability****: $e(\cdot, \cdot)$ can be computed efficiently.*

The cryptographic pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ may be of one of the following types:

- ***Type 1****: $\mathbb{G}_1 = \mathbb{G}_2$.*
- ***Type 2****: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently computable homomorphism ϕ from $\mathbb{G}_1$ to $\mathbb{G}_2$.*
- ***Type 3****: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$.*

If e is of Type 1, then it is called symmetric*, otherwise (i.e., it is of Type 2 or 3), it is called* asymmetric*.*

Definition 2.7 (***DLOG assumption***)**.** *Let λ be the security parameter and* GGen *be a group generator that on input 1^λ outputs the description $\langle \mathbb{G} \rangle$, the order q and a generator g of some (multiplicative) cyclic group $\mathbb{G}$. We say that the* discrete logarithm *(DLOG) assumption holds for* GGen *if for every PPT algorithm $\mathcal{A}$, it holds that*

$$\mathsf{Pr}\left[\left(\langle \mathbb{G} \rangle, q, g\right) \leftarrow \mathsf{GGen}(1^\lambda); x \xleftarrow{\$} \mathbb{Z}_q : x \leftarrow \mathcal{A}\left(\langle \mathbb{G} \rangle, q, g, g^x\right)\right] = \mathsf{negl}(\lambda) \ .$$

Definition 2.8 (***DDH assumption***)**.** *Let λ be the security parameter and* GGen *be a group generator that on input 1^λ outputs the description $\langle \mathbb{G} \rangle$, the order q and a generator g of some (multiplicative) cyclic group $\mathbb{G}$. We say that the* decisional Diffie – Hellman *(DDH) assumption holds for* GGen *if the following r.v. ensembles are computationally indistinguishable:*

- $\left\{\left(\langle \mathbb{G} \rangle, q, g\right) \leftarrow \mathsf{GGen}(1^\lambda); x, y \xleftarrow{\$} \mathbb{Z}_q : (g, g^x, g^y, g^{xy})\right\}_{\lambda \in \mathbb{N}}$ *and*

- $\left\{ \left(\langle \mathbb{G} \rangle, q, g \right) \leftarrow \mathsf{GGen}(1^\lambda); x, y, z \xleftarrow{\$} \mathbb{Z}_q : (g, g^x, g^y, g^z) \right\}_{\lambda \in \mathbb{N}}$.

Definition 2.9 (***SXDH assumption***). *Let* λ *be the security parameter and* BGGen *be a bilinear group generator that on input* 1^λ *outputs the descriptions* $\langle \mathbb{G}_1 \rangle, \langle \mathbb{G}_2 \rangle, \langle \mathbb{G}_T \rangle$ *and the order* q *of three (multiplicative) cyclic groups* $\mathbb{G}_1, \mathbb{G}_2$ *and* $\mathbb{G}_T$ *respectively, two generators* g_1, g_2 *of* $\mathbb{G}_1, \mathbb{G}_2$ *and the description* $\langle e \rangle$ *of a cryptographic pairing* $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. *We say that the* symmetric external Diffie – Hellman *(SXDH) assumption holds for* BGGen *if for* $i = \{1, 2\}$ *the following r.v. ensembles are computationally indistinguishable:*

- $\left\{ \left(\langle \mathbb{G}_1 \rangle, \langle \mathbb{G}_2 \rangle, \langle \mathbb{G}_T \rangle, q, g_1, g_2, \langle e \rangle \right) \leftarrow \mathsf{BGGen}(1^\lambda); x, y \xleftarrow{\$} \mathbb{Z}_q : (g_i, g_i^x, g_i^y, g_i^{xy}) \right\}_{\lambda \in \mathbb{N}}$

  and

- $\left\{ \left(\langle \mathbb{G}_1 \rangle, \langle \mathbb{G}_2 \rangle, \langle \mathbb{G}_T \rangle, q, g_1, g_2, \langle e \rangle \right) \leftarrow \mathsf{BGGen}(1^\lambda); x, y, z \xleftarrow{\$} \mathbb{Z}_q : (g_i, g_i^x, g_i^y, g_i^z) \right\}_{\lambda \in \mathbb{N}}$.

2.3 Cryptographic Primitives

This section comprises a list of all the cryptographic primitives that will be the foundation for the construction of the e-voting systems discussed in this thesis. Here, we present these primitives in their abstract form. Accordant instantiations will be provided in the description sections of the e-voting systems that each primitive is applied.

2.3.1 Cryptographic hash functions

A hash function H is a function that maps data of arbitrary size to an output of fixed size. A hash value is often referred as *digest*. A *cryptographic hash function* is a hash function that additional satisfies some security standard. In particular, a cryptographic hash function H with digests of length $\ell(\lambda)$, where ℓ is a polynomial in the security parameter λ, satisfies at least one of the following security properties:

- ***First preimage resistance****:* for every PPT adversary $\mathcal{A}$ and every $y \in \{0,1\}^{\ell(\lambda)}$,

$$\mathsf{Pr}[x \leftarrow \mathcal{A}(y) : H(x) = y\,] = \mathsf{negl}(\lambda)\ .$$

- ***Second preimage resistance****:* for every PPT adversary $\mathcal{A}$ and every input x,

$$\mathsf{Pr}[x' \leftarrow \mathcal{A}(x) : H(x) = H(x')\,] = \mathsf{negl}(\lambda)\ .$$

- ***Collision resistance****:* for every PPT adversary $\mathcal{A}$,

$$\mathsf{Pr}[(x, x') \leftarrow \mathcal{A}(1^\lambda) : H(x) = H(x')\,] = \mathsf{negl}(\lambda)\ .$$

As it is commonly done in cryptographic literature, we restrict to *collision resistant hash functions*, for the rest of this thesis. It is easy to see that collision resistance implies second preimage resistance which in turn implies first preimage resistance.

Modelling hash functions as random oracles:

A *random oracle (RO)* is an oracle that for every distinct query returns a truly random response. For all identical queries, the oracle acts as a function, i.e. it returns the same response. When a cryptographic hash function utilised in some construction is replaced by a RO, then the security of the construction is proven in the *RO model*. Settling for security in the RO model has the advantage of yielding protocols significantly more efficient than the respective ones in the standard model [10]. However, security in the RO model has been proven controversial, as there exist cases of cryptographic schemes secure in the RO model, but completely broken when the RO is instantiated with any cryptographic hash function [21].

2.3.2 Linear secret sharing schemes

In a (t, k)-*secret sharing scheme (SSS)*, a dealer splits a secret s into k shares denoted by $\|s\|_1, \ldots, \|s\|_k$ and issues each share $\|s\|_i$ to player P_i, $i \in [k]$. The original secret s can be recovered from any collection of more than t shares. In terms of security, any collection of $t-1$ shares reveals no information about s. An SSS is *linear* if for any two secrets s, s' the i-th share of $s + s'$ is equal to the sum of the i-th share of s and s'. Formally,

$$\forall i \in [k] : \|s + s'\|_i = \|s\|_i + \|s'\|_i \ .$$

2.3.3 Homomorphic commitment schemes

Let λ be the security parameter and $\mathcal{M}$ be a message space. A *commitment scheme* $\mathcal{CS}$ consists of three algorithms $\mathsf{CS.Gen}, \mathsf{CS.com}, \mathsf{CS.Ver}$ described below:

- The *generation* algorithm CS.Gen that on input λ outputs a commitment key ck.
- The *commitment* algorithm CS.Com that on input ck, a message $m \in \mathcal{M}$ and an opening string r, outputs a commitment c to m. We write $c = \mathsf{CS.Com}(\mathsf{ck}, m; r)$, or simply $c = \mathsf{CS.Com}(\mathsf{ck}, m)$ when r is implied.
- The *verification* algorithm CS.Ver that on input ck, a commitment c and a pair (m, r) outputs 1 or 0.

In a commitment scheme, all honestly generated commitments should verify, while it should be infeasible for an attacker to create a commitment that can be opened to different values or obtain any information about the message to which a commitment is generated. Formally, $\mathcal{CS}$ satisfies the following properties:

1. **(*Perfect*) *Correctness*:** for every $m \in \mathcal{M}$ and every r,
$$\Pr\left[\mathsf{ck} \leftarrow \mathsf{CS.Gen}(1^\lambda); c \leftarrow \mathsf{CS.Com}(\mathsf{ck}, m, r) : \mathsf{CS.Ver}\big(\mathsf{ck}, c, (m, r)\big) = 1\right] = 1 \ .$$
2. ***Binding**:* for every PPT adversary $\mathcal{A}$,
$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathsf{CS.Gen}(1^\lambda); (c, m, r, m', r') \leftarrow \mathcal{A}(\mathsf{ck}) : \\ \mathsf{CS.Ver}\big(\mathsf{ck}, c, (m, r)\big) = \mathsf{CS.Ver}\big(\mathsf{ck}, c, (m', r')\big) = 1\end{array}\right] = \mathsf{negl}(\lambda) \ .$$
3. ***Hiding**:* for every PPT adversary $\mathcal{A}$,
$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathsf{CS.Gen}(1^\lambda); (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda); \\ b \xleftarrow{\$} \{0,1\}; c^* \leftarrow \mathsf{CS.Com}(\mathsf{ck}, m_b, r) : b \leftarrow \mathcal{A}(c^*, \mathsf{st})\end{array}\right] = \mathsf{negl}(\lambda) \ .$$

A commitment scheme is *perfectly binding*, if the binding property holds for unbounded adversaries with zero error. In addition, a commitment scheme is *(additively) homomorphic* if for every $m, m' \in \mathcal{M}$ and every commitment key ck,

$$\mathsf{CS.Com}(\mathsf{ck}, m) \cdot \mathsf{CS.Com}(\mathsf{ck}, m') = \mathsf{CS.Com}(\mathsf{ck}, m + m') \ .$$

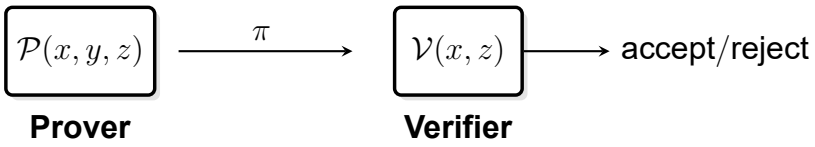
2.3.4 Interactive proof systems

An *interactive proof system (IPS)* for some language $\mathcal{L}$ is a pair of ITMs, $(\mathcal{P}, \mathcal{V})$, where the potentially unbounded *prover* $\mathcal{P}$ interacts with the computationally bounded *verifier* $\mathcal{V}$ in order to prove that a statement x is in $\mathcal{L}$. We consider IPSs where the prover has a private input y (typically a witness w for x, if $\mathcal{L} \in \boldsymbol{\mathcal{NP}}$), and $\mathcal{V}$ is PPT, while there is an additional common auxiliary input z that refers to a priori information available to $\mathcal{P}$ and $\mathcal{V}$. We write $\langle \mathcal{P}(y), \mathcal{V} \rangle(x, z)$ to denote the interaction of $\mathcal{P}$ and $\mathcal{V}$ which outputs accept or reject, depending on whether or not $\mathcal{V}$ is convinced of its interaction with $\mathcal{P}$. An IPS $(\mathcal{P}, \mathcal{V})$ satisfies the following properties:

1. **(*Perfect*) *Completeness*:** for every $x \in \mathcal{L}$, there exists a string y s.t. for every $z \in \{0,1\}^*$,
$$\Pr\left[\mathsf{accept} \leftarrow \langle \mathcal{P}(y), \mathcal{V} \rangle(x, z)\right] = 1 \ .$$
2. ***Soundness*:** for every $x \notin \mathcal{L}$, and for every (potentially malicious) prover $\mathcal{P}^*$ and $y, z \in \{0,1\}^*$,
$$\Pr\left[\mathsf{accept} \leftarrow \langle \mathcal{P}^*(y), \mathcal{V} \rangle(x, z)\right] = \mathsf{negl}(|x|) \ .$$

When we require the soundness property to hold only against PPT provers, then we get the relaxed notion of *computationally sound IPS* or *argument*.

In the special case of a *non-interactive (NI) proof system*, where the interaction is only one-round, the prover generates a proof π for statement x that sends to the verifier, which must decide on the proof's acceptance. For a NI proof system, we illustrate the interaction as below:



In the following subsections, we recall the definitions of widely usde types of IPS (and arguments) that will be deployed in this thesis. For an extended study of IPSs, we refer the reader to [52, Chapter 4].

2.3.4.1 Zero-knowledge proofs

Intuitively, an IPS $(\mathcal{P}, \mathcal{V})$ is zero-knowledge, if the verifier can not obtain any information about $\mathcal{P}$'s proof strategy besides the validity of the statement $x \in \mathcal{L}$. This is modelled via the existence of an efficient *simulator* that the (potentially cheating) verifier could run locally and obtain a string that looks similar to its view when engaging in an actual interaction with $\mathcal{P}$. Formally, a *zero-knowledge (ZK) proof* for some language $\mathcal{L}$ [55] is an IPS $(\mathcal{P}, \mathcal{V})$ that, achieves completeness, soundness, and the following property:

3. ***Perfect* (resp. *statistical*) (resp. *computational*) *zero-knowledge*:** for every (resp. unbounded) (resp. PPT) verifier $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S}^*$ s.t. the following r.v. ensembles are identical (resp. statistically indistinguishable) (resp. computationally indistinguishable):

   - $\left\{\langle \mathcal{P}^*(y), \mathcal{V}\rangle(x, z)\right\}_{x\in\mathcal{L}, y, z\in\{0,1\}^*}$, for every y s.t. $|y|$ is polynomial in $|x|$ and
   - $\left\{\mathcal{S}^*(x, z)\right\}_{x\in\mathcal{L}, z\in\{0,1\}^*}$.

   The distinguishing gap is considered as a function of $|x|$.

When we require only computational soundness for $(\mathcal{P}, \mathcal{V})$, then we get the relaxed notion of *zero-knowledge argument*. Furthermore, if we require the zero-knowledge property to hold only against the (honest) verifier $\mathcal{V}$, then we get the relaxed notion of *honest-verifier zero-knowledge (HVZK)*.

2.3.4.2 Proofs of knowledge

An IPS $(\mathcal{P}, \mathcal{V})$ is a proof of knowledge for a language in $\mathcal{NP}$ when the prover, besides the validity of the statement, convinces the verifier that it knows something. This is modelled via the existence of an efficient *knowledge extractor* that, given the code of any convincing prover, can output witnesses of the said statement. Formally, a *proof of knowledge (PoK) with error* $\kappa : \mathbb{N} \longrightarrow [0, 1]$ for some language $\mathcal{L} \in \mathcal{NP}$ with witness relation $R_{\mathcal{L}}$, is an IPS $(\mathcal{P}, \mathcal{V})$ that achieves completeness and the following property:

2. ***Proof of knowledge with error*** $\kappa(\cdot)$**:** there exists a PPT knowledge extractor $\mathcal{K}$ and a polynomial $q(\cdot)$ s.t. for every $x \in \mathcal{L}$ and for every (potentially malicious) prover $\mathcal{P}^*$ and $y, z \in \{0,1\}^*$,

$$\Pr\left[\mathcal{K}^{\mathcal{P}^*(x,y,z)}(x)\right] > q\big(p_x^* - \kappa(|x|)\big), \text{ where } p_x^* \triangleq \Pr\left[\text{accept} \leftarrow \langle \mathcal{P}(y), \mathcal{V}\rangle(x,z)\right] > \kappa(|x|) \ .$$

It is easy to see that when $\kappa(\cdot)$ is a negligible function, then PoK implies the standard soundness property. Moreover, if we require that PoK holds only against computationally bounded provers, then we get the relaxed notion of *argument of knowledge (AoK)*.

2.3.4.3 Σ protocols

A Σ protocol is a special case of an IPS where the interaction is in three rounds and all the coins of the verifier $\mathcal{V}$ are *public*, i.e. provided to the prover $\mathcal{P}$. In the first round, $\mathcal{P}$ sends to $\mathcal{V}$ a *commitment* message a. In the second round $\mathcal{V}$ provides $\mathcal{P}$ with a *challenge* c and in the third round, $\mathcal{P}$ replies with a *response* r. The interaction is illustrated in Figure 2.1.

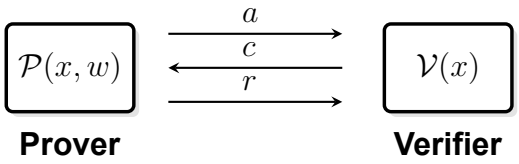


Figure 2.1: A Σ protocol execution for statement x with prover's private input w.

Formally, an IP $(\mathcal{P}, \mathcal{V})$ is a Σ *protocol* for some language $\mathcal{L} \in \mathcal{NP}$ with witness relation $R_{\mathcal{L}}$, if it satisfies the following properties.

1. **(*Perfect*) *Completeness*:** for every $x \in \mathcal{L}$ and every $w \in R_{\mathcal{L}}(x)$,

$$\Pr\left[a \leftarrow \mathcal{P}(x,w); c \leftarrow \mathcal{V}(x,c); r \leftarrow \mathcal{P}(x,a,c,w) : \text{accept} \leftarrow \mathcal{V}(x,a,c,r)\right] = 1 \ .$$

2. ***Special Soundness*:** there exists a witness extractor $\mathcal{K}$ s.t. for every x and every pair of accepting transcripts (a,c,r), (a,c',r'),

$$c \neq c' \Rightarrow \mathcal{K}(x,a,c,r,c',r') \in R_{\mathcal{L}}(x) \ .$$

3. ***Special HVZK*:** there exists a PPT simulator $\mathcal{S}$ that on input $x \in \mathcal{L}$ and a (possibly maliciously sampled) challenge c outputs a simulated transcript $(\tilde{a}, c, \tilde{r})$ which follows the same distribution as a transcript (a,c,r) generated by an execution of $(\mathcal{P}, \mathcal{V})$.

Observe that the existence of the witness extractor in special soundness implies the PoK property with negligible error if the challenge space is large (superpolynomial), while special HVZK implies the perfect HVZK (but not the ZK) property.

For simplicity, in this thesis, we deviate from standard terminology by also characterising as Σ protocols, the three-round public-coin and special-sound IPSs, where special HVZK holds only against polynomially bounded verifiers.

Disjunctive Σ protocols :

In [40], Cramer, Damgård and Schoenmakers introduced a generic technique for transforming a 3-round public-coin special sound HVZK IPS for some language $\mathcal{L}$ into a 3-round public-coin special sound IPS that achieves *witness indistinguishability* [45] for some language $\mathcal{L}_\Gamma$ defined with respect to (w.r.t.) $\mathcal{L}$ and some monotone access rule Γ (cf. [40] for details). Here, we restrict to the most common case used in e-voting constructions where we build upon Σ protocol $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L}$ with witness relation $R_\mathcal{L}$ to obtain a Σ protocol $(\mathcal{P}_\vee^m, \mathcal{V}_\vee^m)$ for the language

$$\mathcal{L}_\vee^m = \big\{(x_1, \ldots, x_m) \mid \exists i : x_i \in \mathcal{L}\big\}$$

with witness relation

$$R_{\mathcal{L}_\vee^m} = \big\{\big((x_1, \ldots, x_m), (w_i, i)\big) \mid (x_i, w_i) \in R_\mathcal{L}\big\} \ .$$

Let $(x_1, \ldots, x_m) \in \mathcal{L}_\vee^m$ and i s.t. $x_i \in \mathcal{L}$. The interaction in $\langle \mathcal{P}_\vee^m(w_i, i), \mathcal{V}_\vee^m \rangle (x_1, \ldots, x_m)$, where $(w_i, i) \in R_{\mathcal{L}_\vee^m}(x_1, \ldots, x_m)$ is as follows:

- *1st round*: for every $j \in [m] \backslash \{i\}$, $\mathcal{P}_\vee^m$, chooses a random challenge c_j from the challenge space $\mathcal{CS}$ of $(\mathcal{P}, \mathcal{V})$ and runs the special HVZK simulator $\mathcal{S}$ of $(\mathcal{P}, \mathcal{V})$ on input c_j to receive a simulated transcript $(\tilde{a}_j, c_j, \tilde{r}_j)$. Next, it executes the 1st round of $(\mathcal{P}, \mathcal{V})$ to generate a commitment a_i for statement x_i. It sends the commitment $\mathbf{a} = (\tilde{a}_1, \ldots, \tilde{a}_{i-1}, a_i, \tilde{a}_{i+1}, \ldots, \tilde{a}_m)$ to $\mathcal{V}_\vee^m$.
- *2nd round*: $\mathcal{V}_\vee^m$ provides $\mathcal{P}_\vee^m$ with a random challenge $\mathbf{c}$ to $\mathcal{P}_\vee^m$.
- *3rd round*: upon receiving $\mathbf{c}$, $\mathcal{P}_\vee^m$, computes $c_i = \big(\bigoplus_{j \in [m] \backslash \{i\}} c_j\big) \oplus \mathbf{c}$. Then, it executes the 3rd round of $(\mathcal{P}, \mathcal{V})$ for statement x_i on challenge c_i to generate a response r_i. It sends the response $\mathbf{r} = \big((c_1, \tilde{r}_1), \ldots, (c_{i-1}, \tilde{r}_{i-1}), (c_i, r_i), (c_{i+1}, \tilde{r}_{i+1}), \ldots, (c_m, \tilde{r}_m)\big)$ to $\mathcal{V}_\vee^m$.
- *Verification* : $\mathcal{V}_\vee^m$ accepts $(\mathbf{a}, \mathbf{c}, \mathbf{r})$ if and only if (iff) $c_i = \big(\bigoplus_{j \in [m] \backslash \{i\}} c_j\big) \oplus \mathbf{c}$ and all transcripts $(\tilde{a}_1, c_1, \tilde{r}_1), \ldots, (\tilde{a}_{i-1}, c_{i-1}, \tilde{r}_{i-1}), (a_i, c_i, r_i), (\tilde{a}_{i+1}, c_{i+1}, \tilde{r}_{i+1}), \ldots, (\tilde{a}_m, c_m, \tilde{r}_m)$ are accepting w.r.t. $(\mathcal{P}, \mathcal{V})$.

Proposition 2.1. *Let λ be the security parameter and let m be polynomial in λ. If $(\mathcal{P}, \mathcal{V})$ is a Σ protocol for language $\mathcal{L}$, then $(\mathcal{P}_\vee^m, \mathcal{V}_\vee^m)$ is a Σ protocol for language $\mathcal{L}_\vee^m$.*

Proof.

1. ***Perfect Completeness*** is straightforward form the perfect completeness of $(\mathcal{P}, \mathcal{V})$.

2. ***Special Soundness*:** given two accepting transcripts $(\mathbf{a}, \mathbf{c}, \mathbf{r})$ and $(\mathbf{a}, \mathbf{c}', \mathbf{r}')$ s.t. $\mathbf{c} \neq \mathbf{c}'$, we construct a knowledge extractor $\mathcal{K}_{\vee}^{m}$ as follows:

   (i). $\mathcal{K}_{\vee}^{m}$ locates the index t s.t. $c_t \neq c'_t$ (the existence of t is guaranteed by the fact that $\mathbf{c} \neq \mathbf{c}'$).

   (ii). $\mathcal{K}_{\vee}^{m}$ runs the witness extractor $\mathcal{K}$ of $(\mathcal{P}, \mathcal{V})$ on input the (accepting) transcripts (a_t, c_t, r_t), (a_t, c'_t, r'_t) and obtains the output w_t of $\mathcal{K}$.

   (iii). $\mathcal{K}_{\vee}^{m}$ returns the value (w_t, t)

   By the special soundness of $(\mathcal{P}, \mathcal{V})$, (w_t, t) is a witness for $(x_1, \ldots, x_m) \in \mathcal{L}_{\vee}^{m}$.

3. ***Special HVZK*:** the simulator $\mathcal{S}_{\vee}^{m}$ for $(\mathcal{P}_{\vee}^{m}, \mathcal{V}_{\vee}^{m})$ on input a challenge $\mathbf{c}$ chooses random challenges $c_1, \ldots, c_{m-1}$ and computes $c_m = \left(\bigoplus_{j\in[m-1]} c_j\right) \oplus \mathbf{c}$. Then, it invokes the sHVZK simulator $\mathcal{S}$ of $(\mathcal{P}, \mathcal{V})$ m times on inputs $c_1, \ldots, c_m$ and upon receiving the m simulated transcripts $(\tilde{a}_1, c_1, \tilde{r}_1), \ldots, (\tilde{a}_m, c_m, \tilde{r}_m)$ that $\mathcal{S}$, it returns

$$(\tilde{\mathbf{a}}, \mathbf{c}, \tilde{\mathbf{r}}) = \left((\tilde{a}_1, \ldots, \tilde{a}_m), \mathbf{c}, \langle(c_1, \tilde{r}_1), \ldots, \langle(c_m, \tilde{r}_m)\right) .$$

   The special HVZK holds by the special HVZK of $(\mathcal{P}, \mathcal{V})$ and the fact that for every $i \in [m]$, the distributions

$$\left\{\mathbf{c} \leftarrow \mathcal{V}_{\vee}^{m}; c_j \xleftarrow{\$} \mathcal{CS}, \;\; j \in [m]\backslash\{i\} : \left(c_1, \ldots, c_{i-1}, \left(\bigoplus_{j\in[m]\backslash\{i\}} c_j\right) \oplus \mathbf{c}, c_{i+1}, \ldots, c_m\right)\right\}$$

   and

$$\left\{\mathbf{c} \leftarrow \mathcal{V}_{\vee}^{m}; c_m \xleftarrow{\$} \mathcal{CS} : \left(c_1, \ldots, c_{m-1}, \left(\bigoplus_{j\in[m]\backslash\{i\}} c_j\right) \oplus \mathbf{c}\right)\right\}$$

   are identical.

■

2.3.4.4 Non-Interactive zero-knowledge proofs

If a ZK proof is non-interactive (NIZK), then the prover must convince the verifier in a single round while preserving the ZK property. NIZK proofs [16] are very useful when minimising communication overhead is important and have various applications, e.g. in the construction of digital signatures and public-key cryptosystems that are secure under chosen ciphertext attacks. Unfortunately, the following negative result by Glodreich and Oren, implies that it is impossible to have NIZK proofs (even with no auxiliary input) in the standard model for non-trivial languages.

Theorem 2.1 (Goldreich & Oren [54])**.** *Let $\mathcal{L}$ be a language for which there exists a NIZK proof or a two-round ZK proof with auxiliary input. Then, $\mathcal{L} \in \mathcal{BPP}$.*

By Theorem 2.1, in order to construct some NIZK proof for a language outside $\mathcal{BPP}$, one must consider some setup assumption. Such an assumption is the existence of a *common reference string (CRS)* that is honestly generated by an algorithm CRS.Gen(1^{λ})

and provided as additional common input. In the CRS model, the pair $(\mathcal{P}, \mathcal{V})$ is a *non-interactive zero-knowledge proof* for some language $\mathcal{L} \in \mathcal{NP}$, if the following properties hold:

1. **(*Perfect*) *Completeness*:** for every $x \in \mathcal{L}$ and every $w \in R_{\mathcal{L}}(x)$,

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{CRS.Gen}(1^\lambda); \pi \leftarrow \mathcal{P}(\mathsf{crs}, x, w) : \mathsf{accept} \leftarrow \mathcal{V}(\mathsf{crs}, x, \pi)\right] = 1 \ .$$

2. ***Soundness*:** for every (potentially malicious) prover $\mathcal{P}^*$, there exists a negligible function $\epsilon(\cdot)$ s.t.

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{CRS.Gen}(1^\lambda); (x, \pi) \leftarrow \mathcal{P}^*(\mathsf{crs}) : \mathsf{accept} \leftarrow \mathcal{V}(\mathsf{crs}, x, \pi) \wedge (x \notin \mathcal{L})\right] < \epsilon(\lambda).$$

3. ***Perfect* (resp. *statistical*) (resp. *computational*) *zero-knowledge*:** there exists a pair of PPT simulators $(\mathcal{S}_{\mathsf{crs}}, \mathcal{S})$ s.t. for every PPT verifier $\mathcal{V}^*$, the following random variables are identical (resp. statistically indistinguishable) (resp. computationally indistinguishable):

   - $\left\{\mathsf{crs} \leftarrow \mathsf{CRS.Gen}(1^\lambda); (x, w) \leftarrow \mathcal{V}^*(\mathsf{crs}); \pi \leftarrow \mathcal{P}(\mathsf{crs}, x, w) : (\mathsf{crs}, \pi)\right\}$, for every $x \in \mathcal{L}$ s.t. $|x|$ is polynomial in λ, and every $w \in R_{\mathcal{L}}(x)$

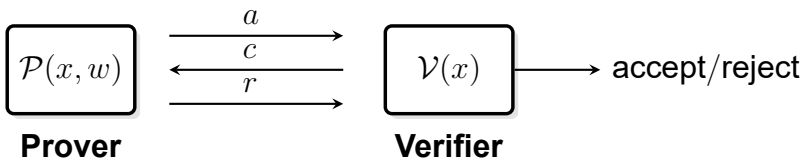
     and

   - $\left\{(\mathsf{crs}, \mathsf{td}) \leftarrow \mathcal{S}_{\mathsf{crs}}(1^\lambda); (x, w) \leftarrow \mathcal{V}^*(\mathsf{crs}); \pi \leftarrow \mathcal{S}(\mathsf{crs}, x, w, \mathsf{td}) : (\mathsf{crs}, \pi)\right\}$, for every $x \in \mathcal{L}$ s.t. $|x|$ is polynomial in λ, and every $w \in R_{\mathcal{L}}(x)$.

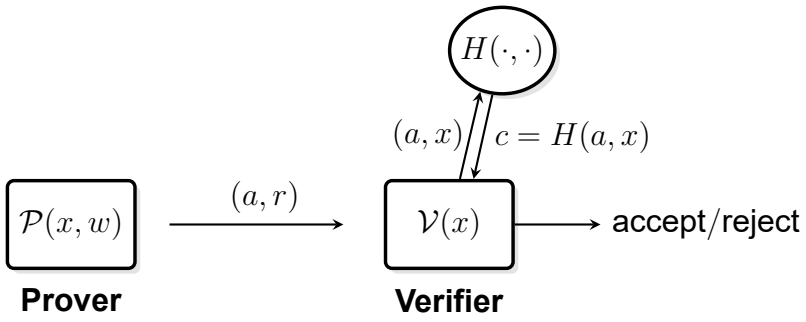
   The distinguishing gap is considered as a function of λ. We stress that the above definition expresses the stronger notion of *adaptive NIZK*, where in soundness and ZK properties the malicious prover and the cheating verifier respectively, may select the statement after seeing the CRS.

   NIZK proofs in the RO model :

   An alternative setup assumption under which efficient NIZK proofs for languages in $\mathcal{NP}$ are feasible is the existence of a RO. A standard methodology for constructing NIZK proofs in the RO model is the *Fiat-Shamir transformation* [46] of a Σ protocol with interaction



   to the NIZK proof



where $H(\cdot,\cdot)$ is a hash function modelled as a RO (cf. Subsection 2.3.1). Namely, $\mathcal{P}$ generates the proof (a,r) by computing r as if given the challenge $c = H(a,x)$ and $\mathcal{V}$ runs the verification algorithm on input (x,a,c,r).

2.3.5 Public key encryption schemes

A *public key encryption (PKE) scheme* $\mathcal{PKE}$ is a triple of algorithms $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ defined as follows:

- The *key generation* algorithm PKE.Gen that on input 1^λ outputs the partial public key and secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- The *encryption* algorithm PKE.Enc that on input pk and a message M in some message space $\mathcal{M}$ outputs a ciphertext C.
- The *decryption* algorithm PKE.Dec that on input sk and a ciphertext C either outputs a message M or aborts.

IND-CPA security of PKE schemes :

A standard semantic security model for PKE schemes is *indistinguishability under chosen plaintext attack (IND-CPA)* , which is defined by a game between a challenger Ch and a PPT adversary $\mathcal{A}$, as described in Figure 2.2.

Definition 2.10 (***IND-CPA security of PKE schemes***)**.** *The PKE scheme $\mathcal{PKE}$ achieves IND-CPA security, if for every PPT algorithm $\mathcal{A}$, it holds that*

$$\Pr\left[\mathcal{G}^{\mathsf{PKE},\mathcal{A}}_{\mathsf{IND-CPA}}(1^\lambda) = 1\right] = 1/2 + \mathsf{negl}(\lambda).$$

One of the most prominent PKE schemes in cryptographic literature is the *ElGamal cryptosystem* [49] that is IND-CPA secure under the DDH assumption.

The IND-CPA game $\mathcal{G}^{\mathsf{PKE},\mathcal{A}}_{\mathsf{IND-CPA}}(1^\lambda)$:

1. The challenger Ch runs PKE.Gen(1^λ) to generate the pair pk, sk. It provides $\mathcal{A}$ with pk.
2. $\mathcal{A}$ sends to challenge messages M_0, M_1 to Ch.
3. Ch flips a coin $b \in \{0,1\}$, computes
$$C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, M_b)$$
and sends C to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b^* and wins the game if only if (iff) $b = b^*$.

Figure 2.2: The IND-CPA game for the PKE scheme $\mathcal{PKE}$ between the challenger Ch and the PPT adversary $\mathcal{A}$.

2.3.6 Threshold public key encryption schemes

Let $\mathsf{Ser}_1, \ldots \mathsf{Ser}_k$ be a set of k decryption servers. A (t,k)-*threshold public key encryption (TPKE) scheme* $\mathcal{TPKE}$ is a quintuple of algorithms $(\mathsf{TPKE.Gen}, \mathsf{TPKE.Combine}, \mathsf{TPKE.Enc}, \mathsf{TPKE.Dec}, \mathsf{TPKE.Recon})$ defined as follows:

- The *partial key generation* algorithm TPKE.Gen that on input 1^λ outputs the partial public key and secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each server Ser_i, $i \in [k]$.
- The *public key construction* algorithm TPKE.Combine that on input $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ computes the public key pk.
- The *encryption* algorithm TPKE.Enc that on input pk and a message M in some message space $\mathcal{M}$ outputs a ciphertext C.
- The *partial decryption* algorithm TPKE.Dec that on input sk_i and a ciphertext C either outputs a partial decryption share D_i or aborts.
- The *plaintext reconstruction* algorithm TPKE.Recon that on input a set of t partial decryption shares $D_{i_1}, \ldots, D_{i_t}$ outputs the message M or aborts.

IND-CPA security of TPKE schemes :

We define the security of TPKE scheme via a game between a challenger Ch and a PPT adversary $\mathcal{A}$, as described in Figure 2.3.

Definition 2.11 (***IND-CPA security of TPKE schemes***). *The (t, k)-TPKE scheme $\mathcal{TPKE}$ achieves IND-CPA security, if for every PPT algorithm $\mathcal{A}$, it holds that*

$$\Pr\left[\mathcal{G}^{\mathsf{TPKE},\mathcal{A}}_{\mathsf{IND-CPA}}(1^\lambda) = 1\right] = 1/2 + \mathsf{negl}(\lambda).$$

The most common instantiation of a TPKE scheme is the (t, k)-*threshold El Gamal cryptosystem* [83] that is IND-CPA secure under the DDH assumption.

The IND-CPA game $\mathcal{G}^{\mathsf{TPKE},\mathcal{A}}_{\mathsf{IND-CPA}}(1^\lambda)$:

1. The challenger Ch runs $\mathsf{TPKE.Gen}(1^\lambda)$ for all decryption servers $\mathsf{Ser}_1, \ldots \mathsf{Ser}_k$ to generate the pairs $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_k, \mathsf{sk}_k)$. It provides $\mathcal{A}$ with $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$. In addition, it initiates the set of corrupted decryption servers **Corrupt** as empty.
2. Throughout the game, $\mathcal{A}$ can make a server corruption request $i \in \{1, \ldots, k\}$. Upon receiving i, Ch sends sk_i to $\mathcal{A}$ and then adds Ser_i to **Corrupt**.
3. $\mathcal{A}$ sends to challenge messages M_0, M_1 to Ch.
4. Ch flips a coin $b \in \{0, 1\}$, computes

   $$C \leftarrow \mathsf{TPKE.Enc}\big(\mathsf{TPKE.Combine}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n), M_b\big)$$

   and sends C to $\mathcal{A}$.
5. $\mathcal{A}$ outputs a bit b^* and wins the game if only if the two following conditions hold:
   (a) $|\textbf{Corrupt}| < t$.
   (b) $b^* = b$.

Figure 2.3: The IND-CPA game for the (t, k)-TPKE scheme $\mathcal{TPKE}$ between the challenger Ch and the PPT adversary $\mathcal{A}$.

2.4 An Overview of Selected e-Voting Systems

In this section, we provide a short overview of some selected e-voting systems, in order to familiarise the reader with e-voting status in the course of the last decade. In our description, we denote the number of voters and election options by n and m, respectively. Even though there are references to some standard cryptographic primitives, several of which are recalled at length in Section 2.3, the systems' description remains mostly at a high level, without necessitating a profound cryptographic background.

2.4.1 RIES

The Rijnland Internet Election System (RIES) [62] is patented by Piet Maclaine Pont and Rijnland Water Board and its design was based on the master thesis of Maclaine Pont's student, Herman Robers [88]. RIES was used from 2004 to 2006 for formal elections of the Dutch District Water Boards, and in 2006 to allow expatriates to vote for the Dutch parliament elections through the Internet.

In RIES, an election authority generates a PKE key pair, $(\mathsf{sk}, \mathsf{pk})$, and n symmetric encryption keys $s_1, \ldots, s_n$ for the scheme Sym.Enc that are distributed to the voters. Using s_ℓ, the voter V_ℓ obtains (i) a unique identity generated as $\mathsf{ID}_\ell = \mathsf{Sym.Enc}(s_\ell, \mathsf{ElectionID})$, where the ElectionID is public, and (ii) m vote-codes $C_{\ell,j} = \mathsf{Sym.Enc}(s_\ell, P_j)$ that correspond to options $\mathsf{opt}_1, \ldots, \mathsf{opt}_m$. Using all keys, the election authority publishes a *pre-election table* of hashed values $\big\{\big(H(\mathsf{ID}_\ell), H(C_{\ell,1}), \ldots, H(C_{\ell,m})\big)\big\}_{\ell\in[n]}$, where $H(\cdot)$ is a public collision-resistant hash function. During voting, V_ℓ generates her vote for selection opt_{j_ℓ} as a PK of $(\mathsf{ID}_\ell, C_{\ell,j_\ell})$ under pk and sends it to the election authority. The individual audit information of V_ℓ is $(\mathsf{ID}_\ell, C_{\ell,j_\ell})$. When election ends, the election authority decrypts all cast votes and publishes a *post-election table* $\{(\mathsf{ID}_\ell, C_{\ell,j_\ell})\}_{\ell\in[n]}$ to a publicly accessible bulletin board, sorted with respect to to the IDs. The tally can be performed easily by hashing all the values in the post-election table and matching the hashed vote-codes in the pre-election table.

E2E verifiability is guaranteed in RIES, since the tally can be computed by any public auditor and the voters can directly check that their votes are counted correctly using their receipt. However, this indisputable verification mechanism also serves as proof of how the voters have voted, hence RIES lacks a stromg level of voter privacy. The use of RIES was terminated in 2008, after internet voting was banned in the Netherlands due to reported security flaws of the Nepad voting machines that were deployed. A detailed criticism on RIES's security was later published in [56], when the system's code was released as open-source.

2.4.2 Civitas/JCJ

Civitas [36] is a remote e-voting system that its design is based upon the JCJ voting scheme of Juels, Catalano and Jakobsson [64]. In Civitas, a committee of *tabulation tellers* generates a PKE key pair and a *registar* prepares a registration key for each voter. When the voter registers, it engages in a protocol with a committee of *registration tellers* to obtain a credential for anonymous authentication. The voter casts her vote by encrypting her credential and her choice along with proofs that the generated encrypted ballot is well-formed. The votes are shuffled via a standard verifiable *mix-net* [23]. At the tally phase, the tabulation tellers collectively discard all unauthorised votes via the anonymous authentication credentials, decrypt and publish the result on the bulletin board. As it is mentioned in [36], the voters' trust on their clients is essential for integrity, therefore E2E verifiability in an all-malicious setting can not be achieved in Civitas. However, the system

features an advanced resistance mechanism against voter coercion, relying on the voters' ability to fake their credentials.

2.4.3 Helios

Helios is a web-based open-audit voting system developed by Adida [2], deployed extensively in the real world, e.g. the International Association of Cryptologic Research (IACR), the Catholic University of Louvain, and Princeton University. Helios culminates a long line of previous schemes that employ homomorphic encryption type of voting, [37, 41, 67], by adjusting the ballot auditing mechanism proposed by Benaloh [11].

At the setup phase of Helios, an election authority generates and distributes voters' credentials while it invites a committee of *trustees* to collectively generate and upload a TPKE public key. When voting, the voters encrypt their ballots under the public key via their supporting devices and may choose either (i) to submit their ballots authenticating with their credentials or (ii) to verify the validity of their ballots (Benaloh audit), after which they are prompted to the creation of a new encrypted ballot. All cast encrypted votes are posted in the bulletin board and can be located via a *ballot tracker* (hash of the encrypted vote). After voting has ended, the trustees compute the tally and publish the results in the bulletin board along with necessary election verification data.

Helios is considered as one of the cutting-edge examples of e-voting technology. The system is studied at length in this thesis (cf. Chapter 6) as case study for the human modelling security framework introduced in Section 3.6.

2.4.4 Remotegrity (combined with Scantegrity II)

Remotegrity, as presented in [99], is a component that is applied for providing assurance of correct posting of the voters' votes. Hence, Remotegrity combined with an E2E verifiable on-site e-voting system like the well-established Prêt à Voter [32] or Scantegrity I/II [29, 27] leads to an E2E verifiable remote e-voting system. Here, we briefly describe a version of Remotegrity built upon Scantegrity II [27]. The system's description considers a *publicly verifiable coin flip* function $R(\cdot)$ that all parties have oracle access.

At election preparation, an election authority generates a secret seed K for some pseudorandom generator. Using the seed, it creates a set of $2n$ Remotegrity ballots that each of them consist of (i) a Scantegrity ballot which includes a serial number and m confirmation vote-codes which are randomly assigned to each of the candidates and (ii) an authorization card (AC) with its own serial number, a set of (four) authentication codes under scratch-off, an acknowledgement code, and a lock-in code under scratch-off. Each of the n voters receives a pair of Remotegrity ballots. Next, the election authority *commits* to the validity of Scantegrity ballot generation by posting in the bulletin board all ballots' information in encrypted form, under a *two-layer permutation*. It also creates a table used for public tally initialised as empty. For each Scantegrity ballot, the two-layer permutation

of its encryption implies a consistent link with the respective cells in the tally table.

During voting, the voter chooses one of the two received Remotegrity ballots to vote leaving the other for audit after election end. Then, she engages in the Remotegrity protocol with the election authority to authenticate and cast her vote which consists of the serial number of the Scantegrity ballot used for voting, the vote-code for the option of her choice and the serial number of the audit ballot.

At the tally phase, the election authority counts the votes for each option by marking the right cells of the tally table. The proof of marking consistency is by disclosing one of the two permutation layers that link the Scantegrity ballot used for voting with the respective cells in the tally table. Which of the two permutations is going to be disclosed, is determined by the output of the coin flip function $R(\cdot)$. Finally, the election authority proves ballot generation consistency by decrypting the information in the bulletin board associated with all auditing ballots. Then, the voters can verify the election execution using their cast vote-codes and the ballot chosen for auditing.

The cut-and-choose nature of both aforementioned proof mechanisms, allows election verification without compromising privacy. E2E verifiability in Remotegrity/ Scantegrity II relies on the assumption that $R(\cdot)$ is a randomness beacon.

2.4.5 Norwegian/Scytl

The Norwegian e-voting system is a variant of the Scytl e-voting protocol[1]. The system was applied in trial executions for 2011 municipal elections and the 2013 parliamentary election. A detailed analysis of the system that was applied in the trial run of 2011 municipal elections can be found in [50], while an improved version is studied in [51]. The e-voting project in Norway was abandoned because of people's concern for their votes' privacy.

Before voting launch, an election authority provides a *receipt generator*, a *ballot box*, and a *decryption service* with independent private keys and posts their corresponding public keys in the bulletin board. The three keys satisfy a linear relationship which allows the extraction of the correct receipt codes via algebraic operations on the encrypted votes, while preserving privacy. Using its private key, the receipt generator pseudorandomly generates a list of m receipt codes for each voter (one for each election option) that are handed to the voters before the election.

The voters submit their votes to the ballot box by encrypting them using the public key corresponding to the decryption service's private key. The ballot box uses its own private key to double encrypt the vote and provides it to the receipt generator which recomputes the correct receipt code for the voter upon the double encrypted vote and sends it to the voter for confirmation. The latter operation does not affect the voter's privacy as the receipt generator cannot decrypt a ciphertext produced under ballot box's key.

When the ballot box closes, it sends every voter's final submitted encrypted ballot to the

[1] http://www.scytl.com/products/election-day/scytl-online-voting

decryption service, in random order. The decryption service decrypts all the ciphertexts and publishes the resulting ballots in random order.

The Norwegian system's security is based on the algebraic properties of the underlying cryptographic primitives and a carefully designed task assignment over the three online election administrators (receipt generator, ballot box, decryption service). However, in an all-malicious setting, E2E verifiability cannot be achieved. Namely, a malicious election authority holds all the keys. Hence, it can deceive the voter by providing her with the receipt codes for the candidate she has chosen while creating ciphertexts for another candidate and tally over encryptions of its choice.

2.5 Literature on e-Voting Security Modelling

2.5.1 Modelling verifiability

In [23], Chaum suggested for the first time that anonymous communication can lead to voting systems with *individual verifiability*.The notion of *universal verifiability* has been introduced in [89], and formally defined in [64]. Kremer, Ryan and Smyth [69] the verifiability of Helios 2.0 in a symbolic framework framework. A formal definition is also provided in [33].

End-to-end verifiability in the sense of cast-as-intended, recorded-as-cast, tallied-as-recorded was an outcome of the works in [26] and [80]. The term of E2E verifiability (or more precisely, E2E integrity) also appeared in [38]. In [85], Popoveniuc *et al.* proposed a definition of E2E verifiability via a list of properties. Küsters, Truderung and Vogt [70] introduced symbolic and computational definitions of verifiability. In [72], showed that individual verifiability and universal verifiability are not sufficient to guarantee the "global" verifiability of an e-voting system. In [73], the same authors introduced a new type of attacks that they name clash attacks, which compromise the integrity of Helios, for variants where the ballots are not linked with the identities of the voters. Furthermore, a computational cryptographic definition of verifiability is proposed by Smyth, Frink and Clarkson in [95].

2.5.2 Modelling privacy and receipt-freeness

Benaloh and Fischer [37] provided a computational definition of privacy while receipt-freeness has been first studied by Benaloh and Tuinstra [13]. Chevallier-Mames *et al.* [33] introduced definitions for unconditional of privacy and receipt-freeness. Formal definitions for privacy and receipt-freeness have been proposed in the context of applied pi calculus [42] and the universal composability model [57, 78]. In [72], the level of privacy of an e-voting system is measured w.r.t. to the observation power the adversary has in a protocol run.

In [14], Bernhard *et al.* proposed a game-based notion of ballot privacy and study the privacy of Helios. Their definition was extended by Bernhard, Pereira and Warinschi [15] by allowing the adversary to statically corrupt election authorities. Both these definitions, although they imply a strong indistinguishability property, do not consider receipt-freeness. We note that our game-based definition captures both privacy and receipt-freeness while restricted to a single EA (and it can easily be extended by including a set of trustees that the adversary may corrupt).

As we have mentioned in the introduciton, modelling coercion resistance is out of the scope of this work. We refer the reader to [64, 42, 78, 71, 4] for formal definitions of coercion resistance in the cryptographic, symbolic and universal composability [19] model.

3. FRAMEWORK

Formalising e-voting security is alone a challenging problem. An electronic election is a complex procedure that incorporates a set of subprotocols and algorithms which in turn rely on various cryptographic primitives. Despite that fact that the requirements an e-voting system should satisfy are known and widely accepted (cf. [82, 63, 98, 85, 34] for a complete view), following a strict mathematical approach for proving e-voting security demands a series of non-trivial steps.

In this chapter, we introduce a complete cryptographic framework for the study of e-voting systems. First, we abstract the entities involved in an e-voting system which we typically express as a quintuple of algorithms and interactive protocols (cf. Section 3.1). We recall the most significant requirements that an e-voting system should satisfy (cf. Section 3.2). Then, we focus on the formalisation of E2E verifiability and voter privacy, including *passive coercion resistance (PCR)*, i.e. the property of preventing voters from proving the way they voted. Specifically, we provide game-based definitions of E2E verifiability and voter-privacy/PCR (cf. Sections 3.3 and 3.4) as well as simulation-based definition of voter-privacy/PCR (cf. Section 3.5).

Furthermore, we extend our framework to investigate the importance of the human factor in e-voting security (cf. Section 3.6). For this reason, we separate the human nodes from the computer nodes in our study, along the lines of the *ceremony model* proposed by Ellison in [44].

At the end of the chapter (cf. Section 3.7), we describe informally an adaptation of the universally composable incoercibility model introduced in [4] to the e-voting environment. The latter will serve as formal basis for follow up e-voting constructions in the full (active) coercion resistance setting.

3.1 Syntax and Correctness of an e-Voting System

In this section, we specify the entities involved in an e-voting system, fixing the standard notation for the rest of the thesis (cf. Subsections 3.1.1 and 3.1.2). Then, we describe the algorithm and protocols that constitute an e-voting system (cf. Subsection 3.1.3) and formally determine a correct election execution (cf. Subsection 3.1.4).

3.1.1 Preliminaries

We use λ as the security parameter and consider three additional parameters; the number of voters n, options m, and trustees k, all of which are thought as polynomial in λ.

For an e-voting system $\boldsymbol{\mathcal{VS}}$, we fix the set of options $\mathcal{O} = \{\mathsf{opt}_1, ..., \mathsf{opt}_m\}$. We denote by $\mathcal{U} \subseteq 2^{\mathcal{O}}$ the collection of subsets of options that the voters are allowed to choose to vote for (which may include a “blank” option too). The option selection $\mathcal{U}_\ell$ of voter V_ℓ is an element

in $\mathcal{U}$.

Let $\mathcal{U}^*$ be the set of vectors of option selections of arbitrary length. Let f be the *election evaluation function* from $\mathcal{U}^*$ to the set $\mathbb{Z}_+^m$ so that $f(\mathcal{U}_1, \ldots, \mathcal{U}_n)$ is equal to an m-vector which i-th location is equal to the number of times opt_j was chosen in the option selections $\mathcal{U}_1, \ldots, \mathcal{U}_n$.

3.1.2 Entities involved in an e-voting system

The entities involved in an e-voting system $\boldsymbol{\mathcal{VS}}$ are the following:

- The *election authority* EA that prepares all the election information.
- The *voters* $\mathcal{V} = \{V_1, \ldots, V_n\}$, possibly equipped with *voting supporting devices (VSDs)*.
- The *vote collector* VC that realises the digital ballot box functionality.
- The set of *trustees* $\mathcal{T} = \{T_1, \ldots T_k\}$ responsible for computing the tally and announcing the election result.
- A publicly accessible and consistent *bulletin board* BB where the election result and all audit information is posted.

3.1.3 Protocols and algorithms in an e-voting system

We formalise an e-voting system $\boldsymbol{\mathcal{VS}}$ as a quintuple of algorithms and interactive protocols $\langle$**Setup**, **Cast**, **Tally**, **Result**, **Verify**$\rangle$ specified as follows:

The **Setup**$(1^\lambda, \mathcal{O}, \mathcal{V}, \mathcal{U}, \mathcal{T})$ *protocol* :

The setup phase is executed by the EA, the BB, the voters $\mathcal{V} = \{V_1, ..., V_n\}$, the trustees $\mathcal{T} = \{T_1, \ldots T_k\}$ and the VC. The protocol generates $\boldsymbol{\mathcal{VS}}$'s public election information **info** (which include $\mathcal{O}, \mathcal{V}, \mathcal{U}$) and the voter credentials $\text{cr}_1, \ldots, \text{cr}_n$. In addition, the EA distributes $\text{cr}_1, \ldots, \text{cr}_n$ to the voters $V_1, \ldots, V_n$ and posts an public *election transcript* τ initialised as **info** on BB. After the protocol execution, each trustee T_i has a private state st_i, and the VC has a private state st_{vc}.

The **Cast** *protocol* :

The voting phase is executed by the voters, the VC and the BB. Specifically, a voter V_ℓ on input $(\textbf{info}, \text{cr}_\ell, \mathcal{U}_\ell)$, submits her vote v_ℓ to VC. The VC updates its state st_{vc} and BB updates the election transcript τ. Upon successful termination, the voter V_ℓ receives some *individual audit information* audit_ℓ. We denote by view_ℓ the view of the voter V_ℓ in the **Cast** protocol .

The **Tally** *protocol* :

After voting period ends, the tally phase is executed by the VC, the BB and the trustees. In particular, the VC provides each trustee with the set of cast votes $\mathsf{V}_{\mathsf{tally}}$. Then, the trustees collectively compute the election result and upon completion, they update the public transcript τ in the BB.

The **Result**(τ) *algorithm* :

The election result can be computed by any party by parsing the election transcript τ.

The **Verify**(τ, audit) *algorithm* :

The verification algortihm outputs a value in $\{\mathsf{accept}, \mathsf{reject}\}$, where audit is a voter's individual audit information obtained after the voter's engagement in the **Cast** protocol.

3.1.4 Correctness of an e-voting system

The correctness of an e-voting system is defined as follows.

Definition 3.1 (***Correctness***)**.** *Let $m, n, k \in \mathbb{N}$ and let $\mathcal{VS}$ be an e-voting system with m options, n voters and k trustees w.r.t. the evaluation election function f. We say that $\mathcal{VS}$ has (perfect)* correctness*, if for any honest execution of $\mathcal{VS}$ that results in a public transcript τ where the voters $V_1, \ldots, V_n$ cast votes for options $\mathcal{U}_1, \ldots, \mathcal{U}_n$ and obtained individual audit information* $\mathsf{audit}_1, \ldots, \mathsf{audit}_n$ *, it holds that*

$$\textbf{Result}(\tau) = f(\mathcal{U}_1, \ldots, \mathcal{U}_n) \textit{ AND } \bigwedge_{\ell=1}^{n} \big(\textbf{Verify}(\tau, \mathsf{audit}_\ell) = 1\big) \ .$$

3.2 Security Properties of an e-Voting System

We enumerate the most significant requirements that should be satisfied by every reliable e-voting system (cf. Subsection 3.2.1). Subsequently, we introduce informally the security properties that we focus in our study (cf. Subsection 3.2.2).

3.2.1 E-voting system requirements

An ideal e-voting system would address a specific list of requirements. We briefly recall some fundamental e-voting properties, as stated in [34, Section 2]. For an extensive description, we refer the reader to [82, 63, 34].

- *Equal access*: all eligibile voters should have an equal and unrestricted access to election centers.

- *Voter eligibility*: only the voters listed as eligible on the electoral roll should be allowed to vote.
- *One voter-one vote*: the voting system should not permit voters to vote twice.
- *Fault tolerance*: the system's infrastructure should be resilient to the faulty behaviour of possibly up to a number of components or parts.
- *End-to-end (E2E) verifiability* that comprises (i) *individual verifiability*, i.e., voters are able to verify that correct counting of their vote, and (ii) *universal verifiability*, where any party, even an outsider, is able to verify that a well defined set of votes has been collected and they have been included in the final tally according to the election system.
- *Fairness*: the voting system should ensure that no partial results become known prior to the end of the election procedure.
- *Privacy*: it should be impossible for a coalition of parties to extract any information about a voter's ballot beyond what can be inferred from the public tally and the parties inside knowledge.
- *Coercion resistance*: the voting system should not facilitate for any party to coerce voters to vote in a certain way, thus violating the voters' free will. Coercion resistance is a strong property that implies privacy and the inability of the voting system to allow voters to prove the way they voted. For the latter property, which strength lies between privacy and coercion resistance, we will be using the term *passive coercion resistance (PCR)*, as opposed to *active coercion resistance*, where the attacker actively controls the voter's interaction with the system [4] [1].

3.2.2 Modelling security

The security models presented in this thesis allow for the typical study of the properties related to privacy and integrity preservation during and after an election execution. In particular, E2E verifiability, fairness, secrecy and PCR are formally treated, while mechanisms enriching the DEMOS family with a fully coercion-resistant on-site e-voting system are discussed in a more intuitive approach. Nevertheless, the equal access, eligibility and one voter-one vote properties are captured in our systems' design. On the other hand, formalising and realising fault tolerance is out of scope of this thesis. We refer the reader to [35] for a detailed study of fault tolerant e-voting, where the definitions presented here are extended to a fully distributed setting.

[1] The term *receipt-freeness* has been used in the literature as an alternative both to passive coercion resistance [20, 13] and active coercion resistance [78, 97, 4]. To avoid confusion, we avoid the use of receipt-freeness in our terminology and refer explicitly to the two cases of coercion resistance, along the lines of [4].

In detail, we address E2E verifiability, fairness, secrecy and PCR summarised in Section 3.2.1 by formalising the following properties:

End-to-end verifiability:

A major contribution of this thesis, is the realisation of an e-voting system that achieves top-tier level of verifiability. To this end, we model E2E verifiability under a severe adversarial setting, where the attacker may control the entire election (except from the view of the BB) by corrupting the EA, the VC, all the trustees and a fraction of the electorate (cf. Section 3.3). We prove that our constructions are end-to-end verifiable *in the standard model*, i.e. without assuming any trusted hardware, or the existence of an RO or a randomness beacon.

Voter privacy/PCR:

We provide a pair of integrated definitions for voter privacy/PCR, thus capturing the fairness, secrecy and PCR requirements. Our first definition is game-based (cf. Section 3.4) considering a *simulator* that generates fake voters' views that are indistinguishable from the real views (this captures PCR). Our second definition is simulation-based (cf. Section 3.5) requirung indistinguishability between an ideal and a real election setting.

Furthermore, in Section 3.6, we extend our model to analyse the importance of human behaviour as a factor of e-voting security. To achieve this, we separate the human from the computer nodes in an election run and adjust the game-based definitions for E2E verifiability and voter privacy/PCR to the new setting.

Finally, in Section 3.7, we provide a less pedant description of a full coercion resistance framework for the security analysis of a novel remote e-voting system we design concurrently with the writing of this thesis. This framework is an adaptation of the universally composable incoercibility model introduced in [4] to the e-voting case.

Remark 3.1 (*Securely delegating election verification*)**.** A most desirable feature of an e-voting system which achieves E2E verifiability and voter privacy/PCR (like the systems in the DEMOS family), is that it allows the voters to *delegate* the whole verification procedure to a trusted auditor of their choice. This is crucially important in settings where the voter lacks the computational means to perform demanding cryptographic operations necessary for a complete auditing.

3.3 Definition of End-to-end Verifiability

In order to define E2E verifiability formally, we introduce a suitable notation; given that option selections are elements from a set of m choices, we encode them as m-bit strings, where the bit in the j-th position is 1 if and only if option opt_j is selected. Further, we aggregate the election results as the list with the number of votes each option has received.

Thus, the **Result** algorithm outputs a vector in $\mathbb{Z}_+^m$, i.e., the range of the election evaluation function f. In this case, a result is feasible if and only if the sum of all its coordinates is no greater than the number of voters. Subsequently, we introduce the tools that enable reaching a robust definition.

Introducing a vote extractor:

We postulate the existence of a *vote extractor* algorithm $\mathcal{E}$, not necessarily running in polynomial-time, that explains the election transcript: namely, $\mathcal{E}$ receives input of the form (τ, A) where τ is an election transcript and $A = \{\mathsf{audit}_\ell\}_{\ell \in \mathcal{V}_{\mathsf{succ}}}$ is a set of individual audit information obtained during the **Cast** protocol. By $\mathcal{V}_{\mathsf{succ}}$, we denote the set of honest voters that voted successfully. Given such input, $\mathcal{E}$ will compute $n - |\mathcal{V}_{\mathsf{succ}}|$ vectors $\langle \mathcal{U}_\ell \rangle_{V_\ell \in \mathcal{V} \setminus \mathcal{V}_{\mathsf{succ}}}$ in $\{0,1\}^m$ which correspond to the choices of all the voters outside of $\mathcal{V}_{\mathsf{succ}}$ and can be either (i) a option selection if the voter has voted adversarially or (ii) a zero vector if the voter has not voted successfully. In case such values cannot be defined, $\mathcal{E}$ returns the symbol $\perp$. In the special case where all voters are honest and have voted successfully (i.e., $\mathcal{V}_{\mathsf{succ}} = \mathcal{V}$), $\mathcal{E}$ returns no value (outputs the empty set).

Introducing a metric:

Using the notion of vote extractor, we will be capable to express the actual result encoded in an election transcript. The next step is to specify a measure of *deviation* from the actual election result, as such deviation is the objective of the adversary in an E2E verifiability attack. Thus, it is natural to equip the space of results with a *metric*. In our framework, we use the metric d_1 derived by the ℓ_1-norm, $\|\cdot\|_1$ scaled to half, i.e.,

$$\begin{array}{rccl} \mathsf{d}_1: & \mathbb{Z}_+^m \times \mathbb{Z}_+^m & \longrightarrow & \mathbb{R} \\ & (R, R') & \longmapsto & \frac{1}{2} \cdot \sum_{i=1}^{n} |R_i - R'_i| \end{array}$$

where R_i, R'_i is the i-th coordinate of w, w' respectively.

The intuition that motivates the d_1 metric approach can be clarified in the subsequent generic example: let $R \in \mathbb{Z}_+^m$ be the election result that corresponds to the true voter intent of n voters, and $R' \in \mathbb{Z}_+^m$ be the published election result. Denote by $\mathsf{max}(\mathcal{U})$, the maximum cardinality of an element in $\mathcal{U}$ and observe that two encodings of option selections are within $\mathsf{max}(\mathcal{U})$ distance. Therefore, intuitively, if the adversary wants to present R' as the result of the election, it may do that by manipulating the votes of at least $\mathsf{d}_1(R, R')/\mathsf{max}(\mathcal{U})$ voters.

The E2E Verifiability game:

We define the E2E Verifiability game, $G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}$, between the adversary $\mathcal{A}$ and a challenger Ch using a vote extractor $\mathcal{E}$. The game takes as input the security parameter, λ, the number of options, m, the number of voters, n, and the number of trustees k. The game

E2E Verifiability Game $G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$

- The adversary $\mathcal{A}$ chooses a list of options $\mathcal{O} = \{\mathsf{opt}_1, ..., \mathsf{opt}_m\}$, a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a set of trustees $\mathcal{T} = \{T_1, \dots T_k\}$ and the set of allowed option selections $\mathcal{U}$. It provides Ch with the sets $\mathcal{O}, \mathcal{V}, \mathcal{T}, \mathcal{U}$ along with **info** and voter credentials $\mathsf{cr}_1, \dots, \mathsf{cr}_n$. Throughout the game, the challenger Ch plays the role of the BB.
- $\mathcal{A}$ and Ch engage in an interaction where $\mathcal{A}$ schedules the **Cast** protocols of all voters. For each voter V_ℓ, $\mathcal{A}$ can either completely control the voter or allow Ch to operate on their behalf, in which case $\mathcal{A}$ provides a option selection $\mathcal{U}_\ell$ to Ch. Then, Ch engages with $\mathcal{A}$ in the **Cast** protocol so that $\mathcal{A}$ plays the role of VC and V_ℓ's VSD. Provided the protocol terminates successfully, Ch obtains the individual audit information audit_ℓ on behalf of V_ℓ.
- Finally, $\mathcal{A}$ posts the election transcript τ to the BB.

Let $\mathcal{V}_{\mathsf{succ}}$ be the set of honest voters (i.e., those controlled by Ch) that terminated successfully. The game returns a bit which is 1 if and only if the following conditions hold true:

1. $|\mathcal{V}_{\mathsf{succ}}| \geqslant \theta$, (i.e., at least θ honest voters terminated successfully).
2. $\forall \ell \in [n]$: if $V_\ell \in \mathcal{V}_{\mathsf{succ}}$, then $\mathbf{Verify}(\tau, \mathsf{audit}_\ell) = \mathsf{accept}$ (i.e., the voters in $\mathcal{V}_{\mathsf{succ}}$ verify their ballot successfully).

and either one of the following two conditions:

3. (a) If $\perp \neq \langle \mathcal{U}_\ell \rangle_{V_\ell \in \mathcal{V} \setminus \mathcal{V}_{\mathsf{succ}}} \leftarrow \mathcal{E}(\tau, \{\mathsf{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}})$, then

$$\mathsf{d}_1(\mathbf{Result}(\tau), f(\langle \mathcal{U}_1, \dots, \mathcal{U}_n \rangle)) \geqslant \delta \ .$$

   (b) $\perp \leftarrow \mathcal{E}(\tau, \{\mathsf{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}})$.

Figure 3.1: The E2E Verifiability Game between the challenger Ch and the adversary $\mathcal{A}$ w.r.t. the vote extractor $\mathcal{E}$.

is also parameterised by δ, which is the deviation amount (according to the metric $\mathsf{d}_1(\cdot, \cdot)$) that the adversary wants to achieve and θ, the minimum number of voters that $\mathcal{A}$ must allow to vote honestly and terminate successfully.

The adversary $\mathcal{A}$ starts by selecting the voter, option, and trustee identities for given parameters n, m, k. It also determines the allowed ways to vote as described by the set $\mathcal{U}$. Then, $\mathcal{A}$ fully controls the election by corrupting the EA, the VC, all the trustees $\mathcal{T} = \{T_1, \dots T_k\}$ and all the VSDs. In addition, it manages the **Cast** protocol executions where it assumes the role of the VC. For each voter, $\mathcal{A}$ may choose to corrupt her or to allow the challenger to play on her behalf. In the second case, $\mathcal{A}$ provides the honest voter with the option selection that will use in the **Cast** protocol. Finally, $\mathcal{A}$ completes the election execution which results to the complete election transcript published in the BB.

The adversary will win the game provided that all θ honest voters that completed the **Cast** protocol successfully will also audit the result successfully, while either (a) the deviation of the tally is at least δ or (b) the extractor fails to produce the option selection of the dishonest voters. The attack game is specified in detail in Figure 3.1.

Definition 3.2 (***E2E-Verifiability***)**.** *Let $\epsilon \in [0,1]$ and $m, n, k, \delta, \theta \in \mathbb{N}$ with $\delta > 0$ and $0 < \theta \leqslant n$. Let $\mathcal{VS}$ be an e-voting system with m options, n voters and k trustees w.r.t. the evaluation election unction f. We say that $\mathcal{VS}$ achieves* E2E verifiability with error ϵ*, for a number of at least θ honest successful voters and tally deviation δ if there exists a (not necessarily polynomial-time) vote-extractor $\mathcal{E}$ such that for any adversary $\mathcal{A}$*

$$\mathsf{Pr}[G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^{\lambda}, m, n, k) = 1] \leqslant \epsilon \ .$$

Remark 3.2 (*The significance of vote extractor*)**.** In the only previous works where end-to-end verifiability was considered at a "global level" as we do here [72, 70], it was expressed with respect to a set of "good" runs γ of the e-voting protocol in the sense that a judge could test whether the protocol operated within the set γ. Even though sufficiently expressive, this formulation has the disadvantage that the set γ remains undetermined and thus the level of verifiability that is offered by the definition hinges on the proper definition of γ which may not be simple. Using our language the notion of a good run becomes explicit: a run of the e-voting protocol is good provided that the extractor $\mathcal{E}$ produces votes for the malicious voters which if they are added to the votes of the honest voters they produce a result that does not deviate from the published result according to the $\mathsf{d}_1(\cdot,\cdot)$ metric. Note that our vote extractor may require super-polynomial time, in the same way that the set of good runs γ may have a membership test of super-polynomial complexity. We remark that the use of a super-polynomial extractor to define properly the inputs of the malicious participants and hence the soundness of a multi-party protocol is not novel to our work. For example see, Micali, Pass and Rosen [77] where they used a similar construct to prove security of their general multi-party computation protocol.

3.4 Game-based Definition of Voter Privacy/PCR

The definition of privacy concerns the actions that may be taken by the adversary in order to obtain information about the option selections of the honest voters. We specify the goal of the adversary in a very general way; for an attack to succeed, we ask that there is an election result, for which the adversary is capable of distinguishing how the honest voters have voted, while it has access to (i) the individual audit information that the voters obtained after ballot-casting as well as (ii) a set of protocol views that are consistent with all the honest voters' views in the **Cast** protocol instances they participated and the adversary has monitored.

Introducing a view simulator:

Observe that any system secure against the aforementioned attack scenario would possess also PCR, i.e., voters cannot prove how they voted by showing the individual audit information they obtain from the **Cast** protocol or even presenting their view in the **Cast** protocol. Given that in the privacy definition we allow the adversary to observe the view of the voter in the **Cast** protocol, we must allow the voter to be able to "lie" about her view, otherwise an attack could be trivially mounted. Note that this would require the voter's input to the **Cast** protocol to be delivered via an *untappable channel*; in particular, the adversary should not have any side-channel information about the voters' credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$.

In order to capture the PCR property as described above, we utilise an efficient *view simulator* $\mathcal{S}$ that provides a simulated view of the voter in the **Cast** protocol. Intuitively, $\mathcal{S}$ captures the way the voter can lie about her option selection in the **Cast** protocol in case she is coerced to present her view after she completes the ballot-casting procedure. It is imperative that the simulated view is indistinguishable from the actual view the voter obtains.

The Voter Privacy/PCR game:

We formalise the privacy of an election via a *Voter Privacy/PCR* game, denoted by $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}$ between the adversary $\mathcal{A}$ and a challenger Ch. The game takes as input the security parameter, λ, the number of options, m, the number of voters n, and the number of trustees k, while it is parameterised by the maximum allowed number of corrupted voters t. The game returns 1 or 0 depending on whether the adversary wins. An important feature of the game is the presence of a view simulator $\mathcal{S}$ that provides simulated views of the voters to $\mathcal{A}$.

The adversary $\mathcal{A}$ starts by selecting the voter, option and trustee identities for given parameters m, n, k and determines the allowed ways to vote. In addition, $\mathcal{A}$ may corrupt the VC and all-but-one trustees. Throughout the game, the challenger Ch plays the role of the EA and the single honest trustee, denoted by T_h. After the end of the **Setup** protocol and prior to the voting phase, the challenger flips a coin b that will determine its behaviour during the course of the game. Subsequently, the adversary will schedule all **Cast** protocols selecting which voters it prefers to corrupt and which ones it prefers to allow to vote honestly. The adversary is allowed to corrupt at most t voters and their VSDs. The voters that remain uncorrupted and their VSDs are operated by Ch and they are given two candidate option selections to choose. Ch will select which of the two candidates the voter will use in the **Cast** protocol according to the bit b. The adversary receives the individual audit information that is obtained by each voter as well as either (i) the actual view of each voter during the **Cast** protocol, if $b = 0$, or (ii) a *simulated* view, if $b = 1$ (this addresses the PCR aspect). Upon completion of ballot-casting, Ch and $\mathcal{A}$ engage in the **Tally** protocol and posts the election result. Subsequently, $\mathcal{A}$ will attempt to guess b. The attack is successful provided that the adversary has corrupted up to t voters, the election tally is the same w.r.t. the two alternatives provided for each honest voter and the adversary manages to guess the challenger's bit b. The game is presented in more detail in Figure 3.2.

Voter Privacy/PCR Game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m, k)$

- $\mathcal{A}$ on input $1^\lambda, n, m, k$, chooses a list of options $\mathcal{O} = \{\text{opt}_1, ..., \text{opt}_m\}$, a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a set of trustees $\mathcal{T} = \{T_1, ..., T_k\}$ a trustee $T_h \in \mathcal{T}$ and the set of allowed option selections $\mathcal{U}$. It provides Ch with the sets $\mathcal{P}, \mathcal{V}, \mathcal{U}$ as well as the trustee identity T_h.

  Throughout the game, $\mathcal{A}$ corrupts the VC and all the trustees besides T_h, while the challenger Ch plays the role of the EA and T_h.

- Ch and $\mathcal{A}$ engage in the **Setup** protocol on input $(1^\lambda, \mathcal{P}, \mathcal{V}, \mathcal{U})$. After the protocol execution, Ch obtains **info** and voter credentials $\text{cr}_1, \ldots, \text{cr}_n$.

- Ch flips a coin $b \in \{0, 1\}$.

- The adversary $\mathcal{A}$ and the challenger Ch engage in an interaction where $\mathcal{A}$ schedules the **Cast** protocols of all voters which may run concurrently. For each voter $V_\ell \in \mathcal{V}$, the adversary chooses whether V_ℓ is corrupted and acts as follows:

  - If V_ℓ is corrupted, then Ch provides cr_ℓ to $\mathcal{A}$, and then they engage in a **Cast** protocol where $\mathcal{A}$ plays the role of VC, V_ℓ and her VSD while Ch plays the role of EA and BB.
  - If V_ℓ is not corrupted, $\mathcal{A}$ provides two option selections $\langle \mathcal{U}^0_\ell, \mathcal{U}^1_\ell \rangle$ to the challenger Ch which operates on V_ℓ's behalf, using her VSD and $\mathcal{U}^b_\ell$ as the V_ℓ's input. The adversary $\mathcal{A}$ is allowed to corrupt the VC and observe the network trace of the **Cast** protocol where Ch plays the roles of V_ℓ, EA, and BB. When the **Cast** protocol terminates, the challenger Ch provides to $\mathcal{A}$: (i) the individual audit information audit_ℓ that V_ℓ obtains from the protocol, and (ii) if $b = 0$, the current view of the internal state of the voter V_ℓ, view_ℓ, that the challenger obtains from the **Cast** execution, or if $b = 1$, a simulated view of the internal state of V_ℓ produced by $\mathcal{S}(\text{view}_\ell)$.

- Ch and $\mathcal{A}$ engage in the **Tally** protocol, where $\mathcal{A}$ is allowed to observe the network trace of that protocol.

- Finally, $\mathcal{A}$ using all information collected above (including the contents of the BB) outputs a bit b^*.

Let $\mathcal{V}_{\text{corr}}$ be the set of corrupted voters and let $\mathcal{V}_{\text{succ}}$ be the set of honest voters that terminated successfully. The game returns a bit which is 1 iff the following hold true:

1. $b = b^*$ (i.e., the adversary guesses b correctly).
2. $|\mathcal{V}_{\text{corr}}| \leqslant t$ (i.e., the number of corrupted voters is bounded by t).
3. $f(\langle \mathcal{U}^0_\ell \rangle_{V_\ell \in \mathcal{V} \setminus \mathcal{V}_{\text{succ}}}) = f(\langle \mathcal{U}^1_\ell \rangle_{V_\ell \in \mathcal{V} \setminus \mathcal{V}_{\text{succ}}})$ (i.e., the election result w.r.t. the set of honest votes does not leak b).

Figure 3.2: The Voter-privacy/PCR game between the challenger Ch and the adversary $\mathcal{A}$ w.r.t. the view simulator $\mathcal{S}$.

Definition 3.3 (***Game-based Voter Privacy/PCR***). *Let $m, n, k, t \in \mathbb{N}$ with $t \leqslant n$. Let $\mathcal{VS}$ be an e-voting system with m options, n voters and k trustees w.r.t. the evaluation election unction f. We say that $\mathcal{VS}$ achieves* voter privacy/PCR *for at most t corrupted voters, if there is a PPT view simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$:*

$$\left| \mathsf{Pr}[G^{\mathcal{A},\mathcal{S}}_{t\text{-}\mathsf{priv}}(1^\lambda, n, m) = 1] - 1/2 \right| = \mathsf{negl}(\lambda).$$

Remark 3.3. Our game-based voter privacy/PCR definition is close in spirit to witness indistinguishability of IPSs [45]. Namely, the adversary's challenge is to distinguish between two possible lists of option selections (the witnesses) that produce the same tally when restricted to just the honest voters. A potentially stronger privacy requirement would be a simulation-based formulation akin to zero-knowledge in interactive proof systems, e.g., as the one suggested for ballot privacy in [15]. The definition of [15] is incomparable to ours because even though it is simulation-based and it captures malicious behaviour of a subset of multiple trustees, it does not consider PCR. In the following section, we present our simulation-based approach of privacy that additionally captures th PCR property.

3.5 Simulation-based Definition of Voter Privacy/PCR

In our additional simulation-based security definition of voter privacy/PCR, we model privacy as indistinguishability between an ideal world experiment and a real world experiment, described below.

The ideal world experiment:

We consider the ideal functionality $\mathcal{F}_{\mathsf{priv}}$ defined in Figure 3.3 that captures the essential aspects of the election functionality from a privacy perspective (we stress that this is not a full ideal functionality as it is not intended to capture correctness or verifiability which we model separately). All the voters $V_1, \ldots, V_n$ and the EA are modelled as dummy parties that simply forward the inputs they receive from the environment $\mathcal{Z}$ to the ideal functionality $\mathcal{F}_{\mathsf{priv}}$. Note that the environment $\mathcal{Z}$ can schedule all the election entities arbitrarily. The ideal world adversary $\mathcal{S}$ active in the experiment interacts with $\mathcal{F}_{\mathsf{priv}}$ and provides output to $\mathcal{Z}$ which makes a final decision outputting a bit. Note that the interaction between $\mathcal{Z}$ and $\mathcal{S}$ is restricted in this way (in the spirit of [18]) since in our setting it is impossible (we use no setup assumptions to achieve stronger notions of simulation-based security, such as *universal composability* (UC), [19]). We denote the output of the environment in the ideal experiment by $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{priv}},\mathcal{S},\mathcal{Z}}(\lambda)$.

The real world experiment:

In the real world, the entities $\mathsf{EA}, \mathsf{VC}, \mathsf{BB}$ and $\mathcal{T} = T_1, \ldots, T_k$, $\mathcal{V} = V_1, \ldots, V_n$, participate in the e-voting system $\mathcal{VS} = (\textbf{Setup}, \textbf{Cast}, \textbf{Tally}, \textbf{Result}, \textbf{Verify})$ in the presence of an adversary $\mathcal{A}$ who statically corrupts up to t_1 trustees and up to t_2 voters with their VSDs.

The VSDs of the honest voters remain uncorrupted. The voters and the trustees run the protocol on command by the environment $\mathcal{Z}$. Furthermore, $\mathcal{A}$ schedules the **Cast** protocol executions, observes the network traffic and obtains all the individual audit information that the voters receive at their engagement in the **Cast** protocol . The adversary is not allowed to communicate with the environment (we only consider stand-alone security); when it terminates, its output is provided to the environment which in turn will produce a single bit as an output. We denote the output of the environment in the real world experiment by $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda)$.

The ideal functionality $\mathcal{F}_{\mathsf{priv}}$

- Upon receiving $(sid, \mathsf{init}, \mathcal{O}, \mathcal{V}, \mathcal{U})$ from EA, it parses $\mathcal{O}$ as options $\{\mathsf{opt}_1, ..., \mathsf{opt}_m\}$, $\mathcal{V}$ as voters $\{V_1, ..., V_n\}$, and $\mathcal{U}$ voting option selections $\mathcal{U} \subseteq 2^{\mathcal{O}}$. It sets the election status to 'vote' and initiliazes a list records as empty. Finally, it sends $(sid, \mathsf{vote}, \mathcal{O}, \mathcal{V}, \mathcal{U})$ to the adversary $\mathcal{S}$.
- Upon receiving $(sid, \mathsf{cast}, \mathcal{U}_\ell)$ from V_ℓ, if the election status is 'vote' and $\mathcal{U}_\ell \in \mathcal{U}$, then it sends $(sid, \mathsf{cast}, V_\ell)$ to $\mathcal{S}$.
- Upon receiving $(sid, \mathsf{cast}, V_\ell)$ from $\mathcal{S}$, if the election status is 'vote' and $(sid, \mathsf{cast}, \mathcal{U}_\ell)$ was sent before by V_ℓ, it adds $(V_\ell, \mathcal{U}_\ell)$ to records.
- Upon receiving (sid, tally) from $\mathcal{S}$, if the election status is 'vote', it sets the election status to 'tally' and computes the election result $\tau \leftarrow f(\langle \mathcal{U}_\ell \rangle_{(V_\ell, \mathcal{U}_\ell) \in \mathsf{records}})$. Finally, it sends τ to $\mathcal{S}$.

Figure 3.3: The ideal functionality $\mathcal{F}_{\mathsf{priv}}$ for voter privacy and PCR interacting with the ideal world adversary $\mathcal{S}$.

The objective of the adversary is to obtain sufficient information about the honest voters' option selection so that, in collaboration with the environment, is able to distinguish the real from the ideal world execution. The e-voting system is private if for all real-world adversaries $\mathcal{A}$, there is a simulator $\mathcal{S}$ s.t. it is impossible for any environment $\mathcal{Z}$ to distinguish between the real and ideal world experiment. Formally, we have the following definition.

Definition 3.4 (***Simulation-Based Voter Privacy/PCR***)**.** *Let $m, n, k, t_1, t_2 \in \mathbb{N}$ with $t_1 \leqslant k$ and $t_2 \leqslant n$. Let $\mathcal{VS}$ be an e-voting system with m options, n voters and k trustees w.r.t. the evaluation election unction f. We say that $\mathcal{VS}$ achieves* $(t_1, t_2)-$simulation $-$ based voter privacy$/$PCR *if for every PPT adversary $\mathcal{A}$ controlling up to t_1 trustees and up to t_2 voters, there is an adversary $\mathcal{S}$ in the ideal world experiment, such that for every environment $\mathcal{Z}$ it holds that*

$$\left| \Pr[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{priv}},\mathcal{S},\mathcal{Z}}(\lambda) = 1] - \Pr[\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda)] = 1 \right| = \mathsf{negl}(\lambda) \ .$$

The security setting in the simulation-based privacy definition is illustrated in Figure 3.4.

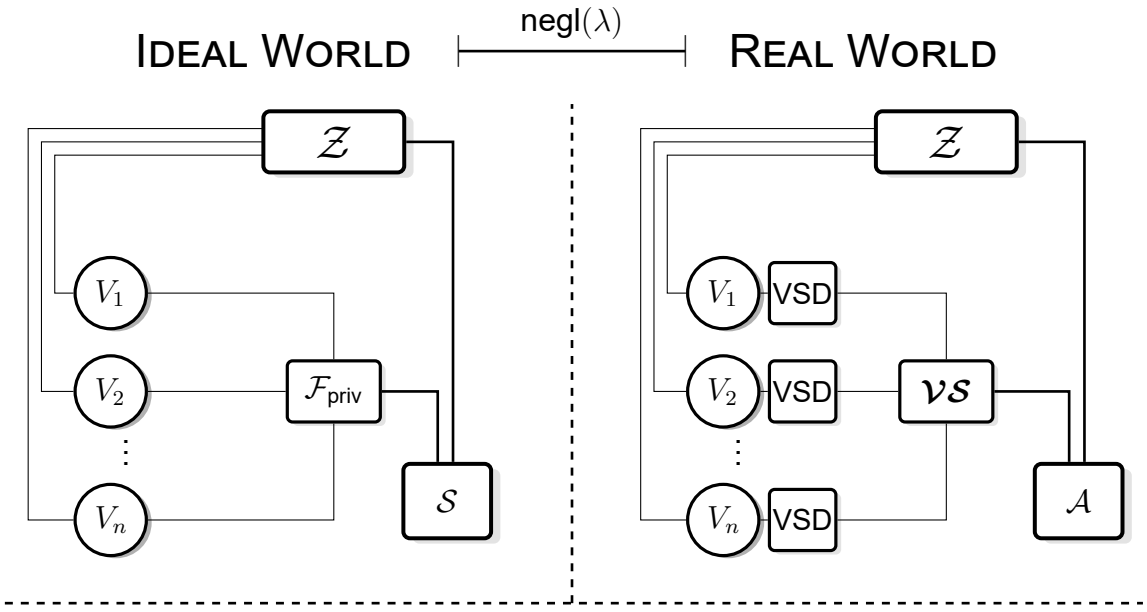


Figure 3.4: The real world-ideal world setting in Definition 3.4.

3.6 The Ceremony Model for e-Voting Systems

A ceremony as introduced by Ellison in [44] is an extension of a network protocol that involves human nodes along side computer nodes. As it will be accentuated in Chapter 6, separating the humans from their supporting devices is not only meaningful but also crucial in certain cases, for a complete study of e-voting security. In the current section, we present our framework for ceremonies in e-voting by suitably extending the modelling introduced in Sections 3.1, 3.3 and 3.4. In our ceremony setting, human nodes will be modelled as probability distributions over a support set of simple finite state machines.

Like in Section 3.1, an e-voting ceremony $\boldsymbol{\mathcal{VC}}$ is associated with three parameters set to be polynomial in the security parameter λ; the number of voters n, the number of options m and the number of trustees k. We use the notation $\mathcal{O} = \{\text{opt}_1, ..., \text{opt}_m\}$ for the set of options, $\mathcal{V} = \{V_1, ..., V_n\}$ for the set of voters and $\mathcal{T} = \{T_1, \ldots, T_k\}$ for the set of trustees. The allowed ways to vote is determined by the collection of subsets $\mathcal{U} \subseteq 2^{\mathcal{O}}$ an the option selection $\mathcal{U}_\ell$ of voter V_ℓ is an element in $\mathcal{U}$. The election evaluation function f is defined as in Subsection 3.1.1.

The entities involved in an e-voting ceremony comprise:

- *The human nodes* are the trustees $T_1, \ldots, T_k$, the voters $V_1, \ldots, V_n$ and the *credential distributor (*CD*)*. The latter additional entity is responsible for issuing the credentials generated at the setup phase to the voters. Note that in practice, the CD may be an organization of more than one human nodes executing another ceremony but we do not model this as part of the e-voting ceremony. Here we make the simplifying choice of modelling CD as a single human node (that is able to identify voters using

an external identification mechanism operating among humans).

- *The computer nodes* are the voting supporting devices (VSDs), the trustee supporting devices (TSDs), the auditing supporting devices (ASDs), the election authority (EA), the bulletin board (BB) and the vote collector VC.

The interaction among the entities involved in an e-voting ceremony is depicted in Figure 3.5.

3.6.1 The entities of an e-voting ceremony

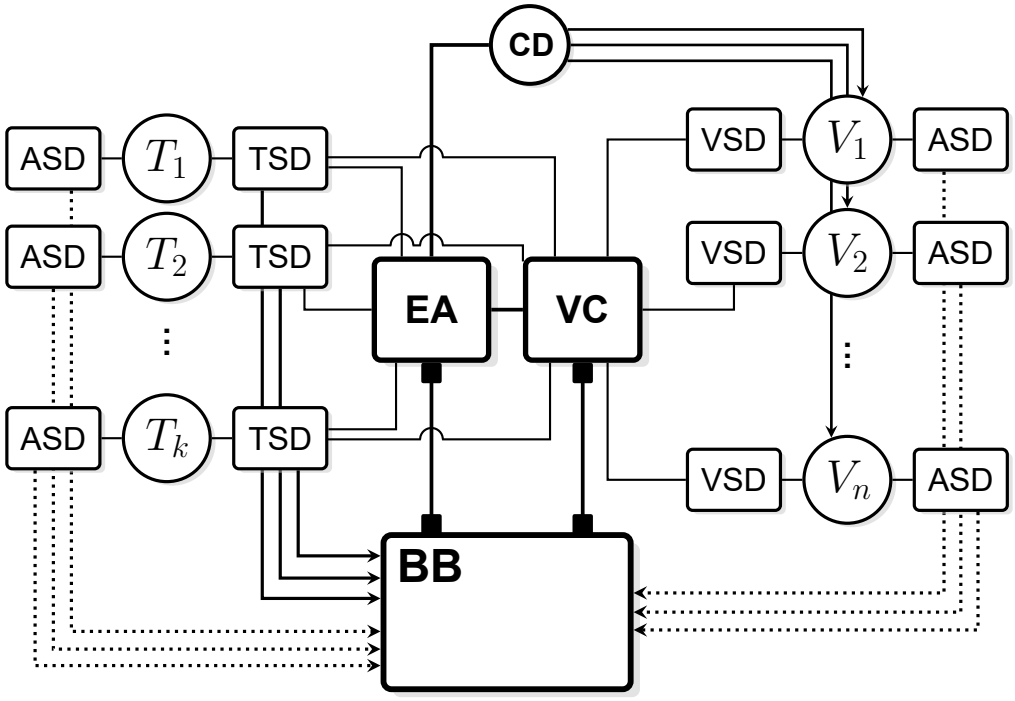


Figure 3.5: The entities and the channels active in an e-voting ceremony. The human nodes and the computer nodes used are shown as circles and rectangles respectively. Each voter or trustee human node, interacts with two computer nodes (supporting devices) while the CD human node interacts with the EA. The dotted lines denote read-only access on the BB.

3.6.2 Modelling human nodes

We model each human node as a collection of simple finite state machines that can communicate with computer nodes (via a user interface) as well as with each other via direct communication. Specifically, we consider a -potentially infinite- collection of *transducers*, i.e. finite state machines with an input and an output tape, that are additionally equipped with a communication tape. Our model of computation for the human nodes is a variant of the probabilistic transducer notion defined below:

Definition 3.5 (***Transducers with communication tape***)**.** *A probabilistic* transducer with communication tape *is a six-tuple* $\langle Q, \Sigma, \Delta, \delta, q_{\text{in}}, F \rangle$ *where*

- Q *is a finite state of states.*
- Σ *is a finite input alphabet.*
- Δ *is a finite output alphabet.*
- δ *is a transition relation from* $Q \times (\Sigma \cup \{\Lambda\})$ *to* $Q \times (\Delta \cup \{\Lambda\}) \times (\Delta \cup \{\Lambda\})$*, where* Λ *is the null symbol. The transition relation associates an input pair of a state* q *and a symbol* w *with a set of possible transition triples of a new state* q' *and two symbols* y', z' *written in the output and communication tape respectively.*
- *For every* $(q, w) \in Q \times (\Sigma \cup \{\Lambda\})$ *and* $(q', y', z') \in Q \times (\Delta \cup \{\Lambda\}) \times (\Delta \cup \{\Lambda\})$*, we assign a probability* $\mathsf{Pr}\left[\delta\big((q, w), (q', y', z')\big)\right]$ *that the transision from* (q, w) *to* (q', y') *occurs. It holds that*
$$\sum_{(q', y') \in Q \times (\Delta \cup \{\Lambda\})} \mathsf{Pr}\left[\delta\big((q, w), (q', y', z')\big)\right] = 1 \ .$$
- $q_{\text{in}} \in Q$ *is the initial state.*
- $F \subseteq Q$ *is the set of accepting states.*

We restrict the size of each voter transducer to depend only on the number of options m. Note that this has the implication that the voter transducer *cannot be used to perform cryptographic operations*, which require polynomial number of steps in λ. Transducers may interact with computer nodes, (supporting devices) and use them to produce ciphertexts and transmit them to other computer nodes. Transducer collections corresponding to voter nodes, trustee nodes and the CD will be denoted as the sets $\mathcal{M}^V$, $\mathcal{M}^T$, and $\mathcal{M}^{\mathsf{CD}}$ respectively. We assume that all sets $\mathcal{M}^V, \mathcal{M}^T$ and $\mathcal{M}^{\mathsf{CD}}$ are polynomial time samplable, i.e., one can produce the description of a transducer from the set in polynomial-time and they have an efficient membership test.

3.6.3 Syntax of an e-voting ceremony

The syntax of an e-voting ceremony $\boldsymbol{\mathcal{VC}}$ is a natural extension of the syntax of an e-voting system presented in Section 3.1. In particular, $\boldsymbol{\mathcal{VC}}$ is a quintuple of algorithms and ceremonies denoted by $\langle$**Setup**, **Cast**, **Tally**, **Result**, **Verify**$\rangle$ together with the sets of transducers $\mathcal{M}^V, \mathcal{M}^T$ and $\mathcal{M}^{\mathsf{CD}}$ that express the human node operations; these are specified as follows:

The **Setup**$(1^\lambda, \mathcal{O}, \mathcal{V}, \mathcal{U}, \mathcal{T})$ *ceremony* :

The setup phase is a ceremony executed by the EA, the BB the VC, a transducer $M^{\mathsf{CD}} \in \mathcal{M}^{\mathsf{CD}}$ describing the behaviour of CD, the transducers $M_i^T \in \mathcal{M}^T$, $i = 1, \ldots, k$ describing the behaviour of the trustees $T_1, \ldots T_k$ respectively and their TSDs. The ceremony generates $\boldsymbol{\mathcal{VC}}$'s public parameters **info** (which include $\mathcal{O}, \mathcal{V}, \mathcal{U}$) and the voter credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$. After the ceremony execution, each TSD has a private state st_i, each trustee T_i obtains a secret $\overline{s}_i$, the VC has a private state $\mathsf{st}_{\mathsf{vc}}$ and the CD obtains the credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$. In addition, the EA posts an election transcript τ initialised as **info** on BB. At the end of the **Setup**, the CD will provide $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$ to the voters $V_1, \ldots, V_n$.

The **Cast** *ceremony* :

The voting phase is a ceremony executed by the VC, the BB, VC, a transducer $M_{i_\ell} \in \mathcal{M}^V$ that determines the behaviour of voter V_ℓ and her supporting devices VSD_ℓ, ASD_ℓ. V_ℓ executes the **Cast** ceremony according to the behaviour M_{i_ℓ} as follows: M_{i_ℓ} has input $(\mathsf{cr}_\ell, \mathcal{U}_\ell)$, where cr_ℓ is the voter's credential and $\mathcal{U}_\ell$ represents the option selection of V_ℓ. All communication between the voter V_ℓ and VC, BB happens via VSD_ℓ. BB has input τ and VC has input $\mathsf{st}_{\mathsf{VC}}$. Upon successful termination, M_{i_ℓ}'s output tape contains a individual audit information audit_ℓ returned by VSD_ℓ. If the termination is not successful, M_{i_ℓ}'s output tape possibly contains a special symbol 'Complain', indicating that voter V_ℓ has decided to complain about the incorrect execution of the election procedure. In any case of termination (successful or not), M_{i_ℓ}'s output tape may contain a special symbol 'Audit', indicating that V_ℓ has taken the decision to use her individual audit information audit_ℓ to perform verification at the end of the election; in this case, the individual audit information audit_ℓ will be provided as input to the ASD of V_ℓ. At the end of the ceremony, EA updates its state and BB updates the public transcript τ as necessary.

The **Tally** *ceremony* :

After voting period ends, the tally phase is a ceremony executed by the VC, the BB and the trustees $M_i^T \in \mathcal{M}^T$, $i = 1, \ldots, k$ as well as their TSDs. Namely, the VC provides each trustee with the set of cast votes $\mathsf{V}_{\mathsf{tally}}$. Then, the trustees collectively compute the election result and upon successful termination, they update the public transcript τ in the BB.

The **Result**(τ) *algorithm* :

The election result can be computed from any party by parsing the election transcript.

The **Verify**(τ, audit) *algorithm* :

The verification algorithm outputs a value in $\{0, 1\}$, where audit is a voter's individual audit information obtained after the voter's engagement in the **Cast** protocol.

3.6.4 Correctness of an e-voting ceremony

The correctness of $\boldsymbol{\mathcal{VC}}$ is defined as follows:

Definition 3.6 (CORRECTNESS OF AN E-VOTING CEREMONY). *Let $m, n, k \in \mathbb{N}$. Let $\mathcal{VC}$ be an e-voting ceremony with m options, n voters and k trustees w.r.t. the evaluation election unction f. We say that $\mathcal{VC}$ has (perfect) correctness, if for any honest execution of $\mathcal{VC}$ w.r.t. any* CD *behaviour specified in $\mathcal{M}^{\mathsf{CD}}$ and any set of trustees' behaviours specified in $\mathcal{M}^T$ that result in a public transcript τ where the voters $V_1, \ldots, V_n$ cast votes for options $\mathcal{U}_1, \ldots, \mathcal{U}_n$ following any of the behaviours specified in $\mathcal{M}^V$ and received individual audit information* $\mathsf{audit}_1, \ldots, \mathsf{audit}_n$*, it holds that*

$$\textbf{Result}(\tau) = f(\mathcal{U}_1, \ldots, \mathcal{U}_n) \textit{ AND } \bigwedge_{\ell=1}^{n} \left(\textbf{Verify}(\tau, \mathsf{audit}_\ell) = 1\right) .$$

3.6.5 End-to-end verifiability of an e-voting ceremony

The formal definition of E2E verifiability in the ceremony model builds upon the game-based definition introduced in Section 3.3. Namely, we utilise a *vote extractor* algorithm $\mathcal{E}$ (not necessarily efficient) that receives as input the election transcript τ and the set of individual audit information obtain **Cast** ceremony and attempts to explain the election result. The tally deviation which is the objective of the attacker is measured via the d_1 metric derived by the ℓ_1-norm, $\|\cdot\|_1$ scaled to half, i.e.,

$$\begin{array}{rccl} \mathsf{d}_1: & \mathbb{Z}_+^m \times \mathbb{Z}_+^m & \longrightarrow & \mathbb{R} \\ & (R, R') & \longmapsto & \frac{1}{2} \cdot \sum_{i=1}^{n} |R_i - R'_i| \end{array}$$

where R_i, R'_i is the i-th coordinate of R, R' respectively. The E2E verifiability game for e-voting ceremonies is described below.

The E2E Verifiability Ceremony game:

Let $\mathcal{D} = \langle \mathbf{D}_1, \ldots, \mathbf{D}_n, \mathbf{D}_1^T, \ldots, \mathbf{D}_k^T, \mathbf{D}^{\mathsf{CD}} \rangle$ be a vector of distributions that consists of the distributions $\mathbf{D}_1, \ldots, \mathbf{D}_n$ over the collection of voter transducers $\mathcal{M}^V$, the distributions $\mathbf{D}_1^T, \ldots, \mathbf{D}_k^T$ over the collection of trustee transducers $\mathcal{M}^T$ and the distribution $\mathbf{D}^{\mathsf{CD}}$ over the collection of CD transducers $\mathcal{M}^{\mathsf{CD}}$. We define the E2E verifiability Ceremony game $G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}$ between the adversary $\mathcal{A}$ and a challenger Ch w.r.t. $\mathcal{D}$ and the vote extractor $\mathcal{E}$ which takes as input the security parameter λ, the number of voters n, the number of options m, and the number of trustees k and is parameterised by (i) the deviation amount δ, (according to the metric $\mathsf{d}_1(\cdot,\cdot)$) that the adversary wants to achieve, (ii) the number of honest voters θ, that terminate the **Cast** ceremony successfully and (iii) the number of honest voters ϕ, that submit a complaint in case of unsuccessful termination during the **Cast** ceremony.

As in Section 3.3, the adversary fully controls the election by corrupting the EA, the VC and all the trustees $\mathcal{T} = \{T_1, \ldots T_k\}$. In addition, it corrupts all the voters VSDs, while the CD remains honest during the setup phase. The adversary manages the **Cast** ceremony executions where it assumes the role of both the VC and the voter's VSD. For each voter V_ℓ, the adversary may choose to corrupt V_ℓ or to allow the challenger to play on her behalf.

Note that the challenger retains the control of the ASD[2] for honest voters and samples for each honest voter a transducer from the corresponding distribution. If a voter V_ℓ is uncorrupted, then the adversary provides the option selection that V_ℓ should use in the **Cast** ceremony; the challenger samples a transducer $M_{i_\ell} \xleftarrow{\mathbf{D}_\ell} \mathcal{M}^V$ from voter transducer distribution $\mathbf{D}_\ell$ and then executes the **Cast** ceremony according to M_{i_ℓ}'s description to vote the given option selection and decide whether to audit the election result at the end. The adversary finally posts the election transcript in the BB. The adversary will win the game provided that there are at least θ of honest voters that terminate the ballot-casting successfully and at most ϕ complaining honest voters, but the deviation of the tally is bigger than δ or the extractor fails to produce the option election of the dishonest voters. The entities that are adversarially controlled in the game are presented in Figure 3.6. The attack game is specified in detail in Figure 3.7.

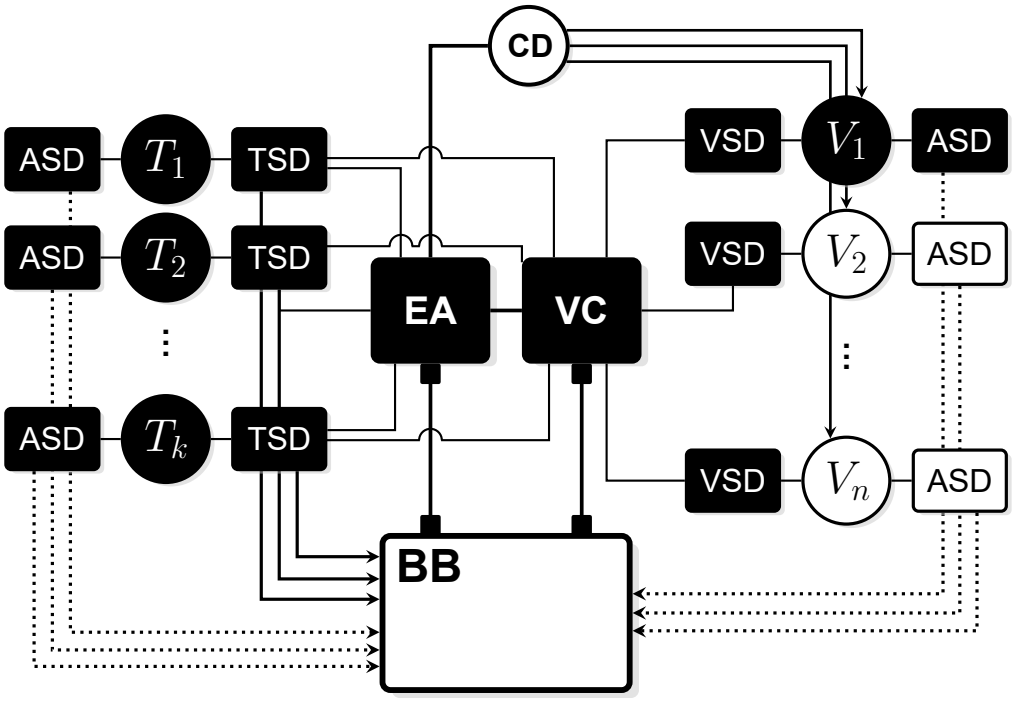


Figure 3.6: The adversarial setting during an attack against E2E verifiability of an e-voting ceremony where the voter V_1 is corrupted. The system nodes that are controlled by the adversary are denoted in black colour.

[2]In the voting phase client-side encryption systems like Helios [2], the voters' ASDs must be live for potential ballot auditing.

E2E Verifiability Ceremony Game $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda, n, m, k)$

- The adversary $\mathcal{A}$ chooses a list of options $\mathcal{O} = \{\text{opt}_1, ..., \text{opt}_m\}$, a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a set of trustees $\mathcal{T} = \{T_1, \dots T_k\}$ and the set of allowed option selections $\mathcal{U}$. It provides Ch with the sets $\mathcal{O}, \mathcal{V}, \mathcal{T}, \mathcal{U}$. Throughout the game, the challenger Ch plays the role of the BB.
- Ch and $\mathcal{A}$ engage in the **Setup** ceremony on input $(1^\lambda, \mathcal{O}, \mathcal{V}, \mathcal{U}, \mathcal{T})$ with $\mathcal{A}$ playing the role of EA, VC and all trustees and their associated TSDs while Ch plays the role of CD (Refer to Fig. 3.6 for an overview of the corrupted nodes) by following the transducer $M^{\text{CD}} \overset{\mathbf{D}^{\text{CD}}}{\leftarrow} \mathcal{M}^{\text{CD}}$. In this way, Ch obtains **info** and the voter credentials $\text{cr}_1, \dots, \text{cr}_n$. . If the CD refuses to distribute the credentials to the voters, then the game terminates.
- $\mathcal{A}$ and Ch engage in an interaction where $\mathcal{A}$ schedules the **Cast** ceremonies of all voters. For each voter V_ℓ, $\mathcal{A}$ can either completely control the voter or allow Ch operate on their behalf. In the latter case, $\mathcal{A}$ provides Ch with an option selection $\mathcal{U}_\ell$. Ch samples a transducer $M_{i_\ell} \overset{\mathbf{D}_\ell}{\leftarrow} \mathcal{M}^V$ and engages with the adversary $\mathcal{A}$ in the **Cast** ceremony so that $\mathcal{A}$ plays the role of VSD_ℓ and VC and Ch plays the role of V_ℓ according to transducer M_{i_ℓ} on input $(\text{cr}_\ell, \mathcal{U}_\ell)$ and its associated ASD_ℓ. Provided the ceremony terminates successfully, Ch obtains the individual audit information audit_ℓ produced by M_{i_ℓ}, on behalf of V_ℓ.
- Finally, $\mathcal{A}$ posts the election transcript τ to the BB.

We define the following subsets of honest voters (i.e., those controlled by Ch):

- $\mathcal{V}_{\text{succ}}$ is the set of honest voters that terminated successfully.
- $\mathcal{V}_{\text{comp}}$ is the set of honest voters s.t. the special symbol 'Complain' is written on the output tape of the corresponding transducer.
- $\mathcal{V}_{\text{audit}}$ is the set of honest voters s.t. the special symbol 'Audit' is written on the output tape of the corresponding transducer.

The game returns a bit which is 1 if and only if the following conditions hold true:

1. $|\mathcal{V}_{\text{succ}}| \geqslant \theta$,
2. $|\mathcal{V}_{\text{comp}}| \leqslant \phi$, (i.e., at most ϕ honest voters complain).
3. $\forall \ell \in [n]$: if $V_\ell \in \mathcal{V}_{\text{audit}}$, then $\textbf{Verify}(\tau, \text{audit}_\ell) = 1$.

and either one of the following two conditions:

4. (a) If $\perp \neq \langle \mathcal{U}_\ell \rangle_{V_\ell \in \mathcal{V} \setminus \mathcal{V}_{\text{succ}}} \leftarrow \mathcal{E}(\tau, \{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}})$, then

$$\text{d}_1(\textbf{Result}(\tau), f(\langle \mathcal{U}_1, \dots, \mathcal{U}_n \rangle)) \geqslant \delta \ .$$

   (b) $\perp \leftarrow \mathcal{E}(\tau, \{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}})$.

Figure 3.7: The E2E Verifiability Ceremony Game between the challenger Ch and the adversary $\mathcal{A}$ w.r.t. the vote extractor $\mathcal{E}$ and the vector of transducer distributions $\boldsymbol{\mathcal{D}} = \langle \mathbf{D}_1, \dots, \mathbf{D}_n, \mathbf{D}_1^T, \dots, \mathbf{D}_k^T, \mathbf{D}^{\text{CD}} \rangle$.

Definition 3.7 (***E2E-Verifiability for e-voting ceremonies***). *Let* $\epsilon \in [0, 1]$ *and* $n, m, k, \delta, \theta, \phi \in \mathbb{N}$ *with* $\theta, \phi \leqslant n$. *The e-voting ceremony* $\boldsymbol{\mathcal{VC}}$ *w.r.t. the election function* f *achieves* E2E

verifiability *with error ϵ, transducer distribution vector $\mathcal{D}$, a number of at least θ honest successful voters, at most ϕ honest complaining voters and tally deviation at most d if there exists a (not necessarily polynomial-time) vote extractor $\mathcal{E}$ such that for every PPT adversary $\mathcal{A}$:*

$$\Pr[G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda, n, m, k) = 1] \leqslant \epsilon.$$

Remark 3.4 (*Universal voter distribution*)**.** We have introduced the collection of transducers $\mathcal{M}^V, \mathcal{M}^T, \mathcal{M}^{\mathsf{CD}}$ to model all possible admissible behaviours that voters, trustees and credential distributors respectively might follow to successfully complete the e-voting ceremony. Note that in the security modelling of the e-voting ceremony, each voter V_ℓ is associated with a distribution $\mathbf{D}_\ell$ over $\mathcal{M}^V$, which captures its voter profile. For instance, the voter V_1 may behave as transducer M_1 with 50% probability, M_2 with 30% probability, and M_3 with 20% probability. In some e-voting systems, the voters can be uniquely identified during the **Cast** ceremonies, e.g. the voter's real ID is used. Hence, the adversary is able to identify each voter V_ℓ and learn its profile expressed by $\mathbf{D}_\ell$. Then, the adversary may choose the best attack strategy depending on $\mathbf{D}_\ell$. Nevertheless, in case the credentials are randomly and anonymously assigned to the voters by the CD, the adversary will not be able to profile voters given his view in the ballot-casting ceremony (recall that in the E2E game the CD remains honest). Therefore, it is possible to unify the distributions to a *universal voter* distribution, denoted as $\mathbf{D}$, which reflects the profile of the "average voter." Specifically, in this case, we will have $\mathbf{D}_1 = \cdots = \mathbf{D}_n = \mathbf{D}$.

3.6.6 Voter privacy/PCR of an e-voting ceremony

The threat model of voter privacy/PCR for ceremonies is an extension of the game-based definition presented in Section 3.4. Specifically, we consider a simulator $\mathcal{S}$ that simulates the voters' views to capture PCR via the ability of the voters to lie about their interaction during the voting phase. Then, we define the following security game that essentially incorporates human behaviour.

The Voter Privacy/PCR Ceremony Game:

Following the same logic as in the E2E Verifiability Ceremony game, we specify a vector of transducer distributions over the collection of voter transducers $\mathcal{M}^V$, trustee transducers $\mathcal{M}^T$ and CD transducers $\mathcal{M}^{\mathsf{CD}}$ denoted by $\boldsymbol{\mathcal{D}} = \langle \mathbf{D}_1, \ldots, \mathbf{D}_n, \mathbf{D}_1^T, \ldots, \mathbf{D}_k^T, \mathbf{D}^{\mathsf{CD}} \rangle$. We then express the threat model as a *Voter Privacy/PCR Ceremony game*, denoted by $G_{t\text{-priv}}^{\mathcal{A},\mathcal{S},\boldsymbol{\mathcal{D}}}$, that is played between an adversary $\mathcal{A}$ and a challenger Ch, that takes as input the security parameter λ, the number of voters n, the number of options m, and the number of trustees k and returns 1 or 0 depending on whether the adversary wins. The game is defined w.r.t. $\boldsymbol{\mathcal{D}}$ and the view simulator $\mathcal{S}$ that provides a simulated view of the voter in the **Cast** ceremony and is parameterised by the maximum allowed number of corrupt voters t. Note that the simulator is not responsible to provide the view of the voter's supporting device (VSD). Intuitively, this simulator captures the way the voter can lie about her choice in the **Cast** ceremony in case she is coerced to present her view after she completes the

ballot-casting procedure. The parties controlled by the adversary during a privacy attack are presented in Figure 3.8.

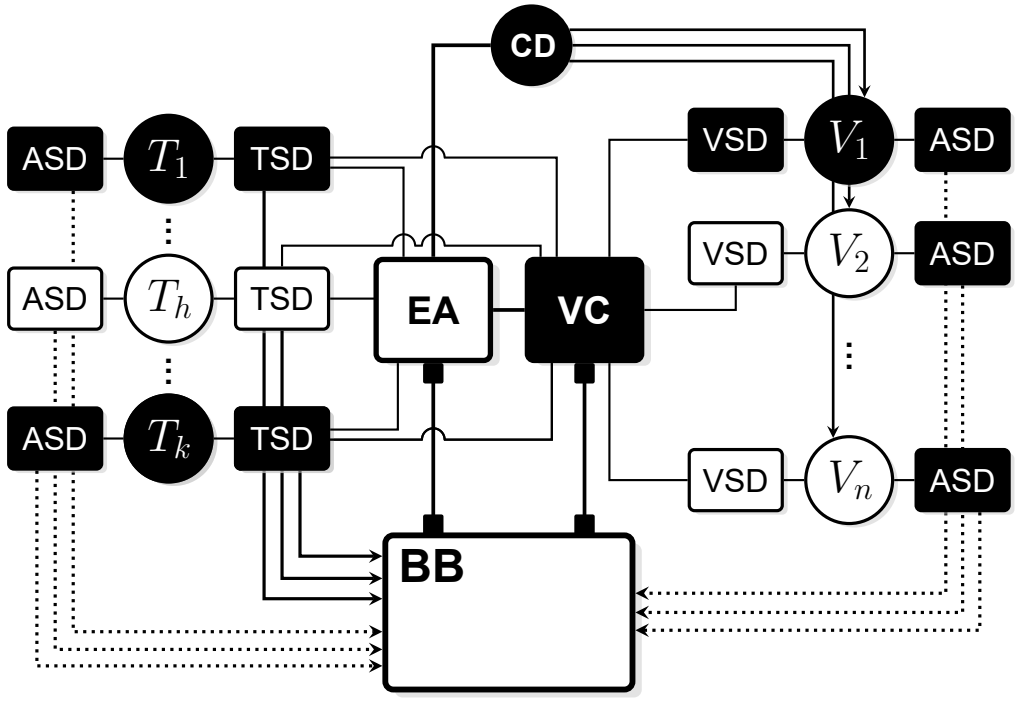


Figure 3.8: The adversarial setting during an attack against voter privacy/PCR an e-voting ceremony where the trustee T_h is honest and the voter V_1 is corrupted. The system nodes that are controlled by the adversary are denoted in black colour.

The adversary starts by selecting the voter, option and trustee identities for given parameters n, m, k. It also determines the allowed ways to vote and selects a single trustee to remain honest together with its TSD and ASD. The challenger subsequently flips a coin b (that will change its behaviour during the course of the game) and will perform the **Setup** ceremony with the adversary playing the role of the CD and of all the trustees and their associated TSDs and ASDs except one trustee that will remain honest. The honest trustee behaviour will be determined by a transducer that is selected at random by the challenger from $\mathcal{M}^T$ according to the corresponding distribution. Subsequently, the adversary will schedule all **Cast** ceremonies selecting which voters it prefers to corrupt and which ones it prefers to allow to vote honestly.

The adversary is allowed to corrupt at most t voters and their VSDs. In addition, $\mathcal{A}$ is allowed to corrupt the VC and the ASDs of all voters. The voters that remain uncorrupted are operated by the challenger and they are given two option selections to vote. For each uncorrupted voter V_ℓ, the challenger first samples a transducer $M_{i_\ell} \xleftarrow{\mathsf{D}_\ell} \mathcal{M}^V$ and then executes the **Cast** ceremony according to M_{i_ℓ}'s description to vote one of its two option selections based on b. The adversary will also receive the individual audit information that

Voter Privacy/PCR Ceremony Game $G_{t\text{-priv}}^{\mathcal{A},\mathcal{S},\boldsymbol{\mathcal{D}}}(1^\lambda, n, m, k)$

- $\mathcal{A}$ on input $1^\lambda, n, m, k$, chooses a list of options $\mathcal{O} = \{\text{opt}_1, ..., \text{opt}_m\}$, a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a set of trustees $\mathcal{T} = \{T_1, ..., T_k\}$ a trustee $T_h \in \mathcal{T}$ and the set of allowed option selections $\mathcal{U}$. It provides Ch with the sets $\mathcal{O}, \mathcal{V}, \mathcal{U}$ as well as the trustee identity T_h.
- Ch flips a coin $b \in \{0, 1\}$ and performs the **Setup** ceremony on input $(1^\lambda, \mathcal{O}, \mathcal{V}, \mathcal{U}, \mathcal{T})$ with the adversary playing the role of the CD and all trustees except T_h, while Ch plays the role of EA and T_h as well as T_h's TSD. The roles of T_h is played by Ch following the transducers $M^{T_h} \overset{\mathbf{D}^{T_h}}{\leftarrow} \mathcal{M}^T$ (Refer to Fig. 3.8 for an overview of the corrupted nodes).
- The adversary $\mathcal{A}$ and the challenger Ch engage in an interaction where $\mathcal{A}$ corrupts the VC and schedules the **Cast** ceremonies of all voters which may run concurrently. $\mathcal{A}$ also controls the ASDs of all voters. At the onset of each voter ceremony, $\mathcal{A}$ chooses whether voter V_ℓ, $\ell = 1, \ldots, n$ and its associated VSD is corrupted or not.
  - If V_ℓ and its associated VSD are corrupted, then no specific action is taken by the challenger, as the execution is internal to adversary.
  - If V_ℓ and its associated VSD are not corrupted, then $\mathcal{A}$ provides Ch with two option selections $\langle \mathcal{U}_\ell^0, \mathcal{U}_\ell^1 \rangle$. The challenger samples $M_{i_\ell} \overset{\mathbf{D}_\ell}{\leftarrow} \mathcal{M}^V$ and sets V_ℓ's input to $(\text{cr}_\ell, \mathcal{U}_\ell^b)$, where cr_ℓ is the credential provided by the adversarially controlled CD. Then, Ch and $\mathcal{A}$ engage in the **Cast** ceremony with Ch controlling V_ℓ (that behaves according to M_{i_ℓ}) and her VSD, while the adversary $\mathcal{A}$ observes the network interaction. When the **Cast** ceremony terminates, the challenger Ch provides to $\mathcal{A}$: (i) the individual audit information audit_ℓ that V_ℓ obtains from the ceremony, and (ii) if $b = 0$, the current view of the internal state of the voter V_ℓ that the challenger obtains from the **Cast** execution, or if $b = 1$, a simulated view of the internal state of V_ℓ produced by $\mathcal{S}(\text{view}_{\text{Ch}})$, where view_{Ch} is the current view of the challenger.
- Ch performs the **Tally** ceremony playing the role of EA, T_h and its associated TSD following M_h^T while $\mathcal{A}$ plays the role of all other trustees.
- Finally, $\mathcal{A}$ terminates returning a bit b^*.

Denote the set of corrupted voters as $\mathcal{V}_{\text{corr}}$. The game returns a bit which is 1 if and only if the following hold true:

1. $b = b^*$ (i.e., the adversary guesses b correctly).
2. $|\mathcal{V}_{\text{corr}}| \leqslant t$ (i.e., the number of corrupted voters is bounded by t).
3. $f(\langle \mathcal{U}_\ell^0 \rangle_{V_\ell \in \mathcal{V} \setminus \mathcal{V}_{\text{corr}}}) = f(\langle \mathcal{U}_\ell^1 \rangle_{V_\ell \in \mathcal{V} \setminus \mathcal{V}_{\text{corr}}})$ (i.e., the election result w.r.t. the set of non-corrupted voters does not leak b).

Figure 3.9: The Voter Privacy/PCR Ceremony Game between the challenger Ch and the adversary $\mathcal{A}$ w.r.t. the view simulator $\mathcal{S}$ and the vector of transducer distributions $\boldsymbol{\mathcal{D}} = \langle \mathbf{D}_1, \ldots, \mathbf{D}_n, \mathbf{D}_1^T, \ldots, \mathbf{D}_k^T, \mathbf{D}^{\text{CD}} \rangle$.

is obtained by each voter as well as either (i) the actual view (if $b = 0$) or (ii) a simulated view, generated by $\mathcal{S}$ (if $b = 1$). Upon completion of ballot-casting, the adversary and the challenger will execute the **Tally** ceremony and subsequently, the adversary will attempt to guess b. The attack is successful provided that the election result is the same with

respect to the two alternatives provided for each honest voter by the adversary and the adversary manages to guess the challenger's bit b correctly. The game is presented in detail in Figure 3.9.

Definition 3.8 (***Voter privacy/PCR of an e-voting ceremony***)**.** *Let $m, n, k, t \in \mathbb{N}$ with $t \leqslant n$. Let $\mathcal{VC}$ be an e-voting system with m options, n voters and k trustees w.r.t. the evaluation election unction f. We say that $\mathcal{VC}$ achieves* voter privacy/PCR *for at most t corrupted voters and for transducer distribution vector $\mathcal{D}$, if there is an efficient simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$:*

$$\left|\mathsf{Pr}[G^{\mathcal{A},\mathcal{S},\mathcal{D}}_{t\text{-priv}}(1^\lambda, n, m, k) = 1] - 1/2\right| = \mathsf{negl}(\lambda).$$

Remark 3.5 (*Corruption of the credential distributor*)**.** In our threat model, we assumed that the CD can be malicious in the voter privacy/PCR ceremony game while it is kept honest for E2E verifiiability. This choice is made for consistency with the level of security that Helios [2] as well as most *client-side encryption* e-voting systems can provide regarding credential distribution (e.g. [41, 65, 67, 96]). Namely, since the vote is encrypted in the voter's VSD, knowing the credential of the voter alone does not suffice for breaking her privacy, whereas for E2E verifibiality, it is important that an honest authority verifies the uniqueness of the credentials, otherwise the election is susceptible to "clash attacks" [73]. If one wishes to study the security of *vote-code based* e-voting systems (e.g. [25, 29] and the DEMOS family), then they would have to take the opposite approach. In such systems, the credentials contain encodings of the options that are personal for each voter, therefore the CD has to be honest for voter privacy/PCR. On the other hand, these systems have mechanisms during the **Cast** ceremony, that inherently guarantee resistance against clash attacks, hence corrupting the CD does not affect their E2E verifiability.

From a security analysis of view, studying human behaviour for e-voting ceremonies based on client-side encryption is a more interesting challenge than the vote-code based case (cf. Chapter 6). Hence, we set as default the CD to be honest for E2E verifiability and malicious for voter privacy/PCR.

3.7 Towards Modelling Full Coercion Resistance

Passive coercion resistance in the security framework of this thesis captures the cases of *semi-honest coercion* (we borrow the terminology from [4]), similarly to the incoercibility definition in [20]. In particular, the adversary acts as a passive coercer which demands from the voter some information obtained from her interaction with the voting system, expecting that this information will serve as attestation of the voter's obedience to the adversary's instructions. Thus, if the e-voting system is designed in a way that allows the voter to lie about her interaction, which we model by introducing a view simulator, then resistance against this type of coercion is guaranteed.

Nevertheless, in a full coercion environment, the attacker may actively control the voters' interaction with the system. Resistance against an *active coercer* is a much stronger

requirement, which is not addressed by any of the e-voting systems presented in this thesis.Consequently, we deliberately omit formalising coercion resistance in this work and we refer the reader to [65, 78, 42, 97, 71, 4] for formal definitions in the cryptographic, symbolic and universal composability model.

Despite the above, concurrently to the writing of this thesis, we develop an advanced version of DEMOS-A e-voting system (cf. Chapter 4, designed in order to provide protection against semi-honest and active coercers (cf. Section 7.2 for an overview). For the security analysis of the promising new member of the DEMOS family, we adjust the UC incoercibility framework introduced in [4] to the special case of receipt-free voting. As an indication of subsequent work, we provide intuition of the [4] model here, and discuss the incoercibility mechanism of the upcoming system in Section 7.2.

Coercion resistance in the [4] model:

As any UC cryptographic framework [19], the [4] model considers an environment that schedules the protocol communication and provides all entities with their inputs. The environment may oversee either of the following two worlds:

1. An *ideal world*, where vote collection, election tally and result announcement is administered by an *ideal election functionality* $\mathcal{F}_{\text{elec}}$. The voters and their supporting devices are modelled as dummy parties that simply forward their inputs to $\mathcal{F}_{\text{elec}}$ and all the messages they receive to the environment. The ideal world assumes the presence of an ideal adversary (simulator) $\mathcal{S}$ that takes control of a subset of the involved parties and interacts with $\mathcal{F}_{\text{elec}}$ and the environment.

2. A *real world*, where the EA, the VC, the trustees, the voters and the BB run the e-voting system $\mathcal{VS}$ as described in Section 3.1 and a real world adversary $\mathcal{A}$ that may corrupt an arbitrary subset of entities except the BB.

In each of the two worlds, the environment plays the role of a coercer for a given coercion strategy. The election runs in either of the following two settings:

A. A *coercion setting*, where the coerced parties follow the instructions given by the coercion strategy precisely.

B. An *evasion setting*, where the coerced parties attempt to execute an evasion strategy in order to deceive the coercer.

Given the above terminology, we say that an e-voting system $\mathcal{VS}$ achieves coercion resistance against a coercion strategy $\mathtt{C}$, if there is an evasion strategy $\mathtt{EV}$ s.t. for every real world adversary $\mathcal{A}$, there is an ideal world adversary $\mathcal{S}$ s.t. for every environment $\mathcal{Z}$ the following conditions hold:

(i). $\mathcal{Z}$ cannot distinguish the ideal world coercion setting from the real world coercion setting.

(ii). $\mathcal{Z}$ cannot distinguish the ideal world evasion setting from the real world evasion setting.

The above conditions imply that the realisation of $\mathcal{F}_{\text{elec}}$ via $\mathcal{VS}$ adds only a negligible term to the distinguishing advantage Δ of $\mathcal{Z}$ between the ideal coercion and the ideal evasion setting that unavoidably stems from the parties' attempt to avoid coercion. The aforementioned discussion is illustrated in Figure 3.10.

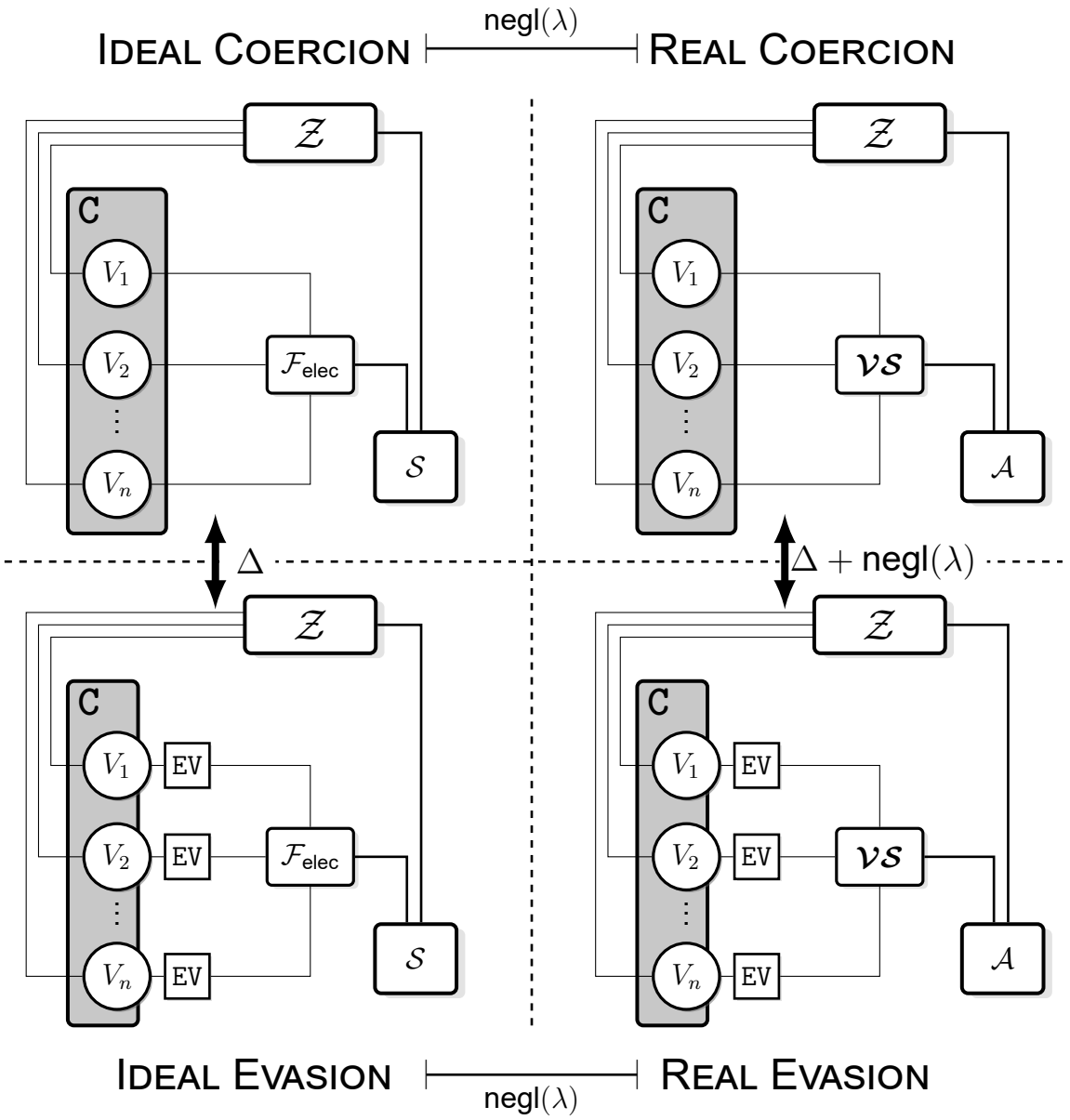


Figure 3.10: The four election sceneries in the [4] incoerciblity model w.r.t. e-voting system $\mathcal{VS}$ and coercion strategy $\mathtt{C}$.

4. THE DEMOS-A E-VOTING SYSTEM

DEMOS-A is an e-voting system that partially in its design borrows ideas from the well-known SureVote and Scantegrity systems [25, 29]. Hence, it is in the spirit of standard code-voting, supporting easy vote casting with minimum computational requirements from the voter's side, as well as the generation of receipts (individual audit information) during the voting phase, that allow the auditing of the election process.

The distinctive attribute that makes DEMOS-A a pioneering e-voting system is its elaborate mechanism for achieving *E2E verifiability in the standard model* for the first time. As already discussed in the introduction, all previous state-of-the-art e-voting systems, required trust in the voters' clients [41, 36, 67], or the existence of a RO [2, 12, 96] or a "randomness beacon" [25, 29, 32, 99] for the verification of the election results. On the contrary, DEMOS-A exploits the randomness inherently and generated by the voters' participation in the voting phase, in order to extract the challenge for the verification of Σ protocols that guarantee BB data consistency.

Furthermore, the tally of the cast votes is computed utilising ElGamal as an additively homomorphic commitment scheme (cf. Subsection 2.3.3). The election process in DEMOS-A is executed preserving privacy, based on the independent random generation of the voters' ballots, the hiding property of the ElGamal commitment scheme and the security of the Shamir linear SSS (cf. Subsection 2.3.2 and [91]) for splitting and distributing the private state of the EA to the trustees at the setup phase.

Chapter roadmap. The presentation of DEMOS-A starts with a high-level overview of the system (cf. Section 4.1) followed by the list of tools applied for its construction (cf. Section 4.2). Subsequently, DEMOS-A is described in detail (cf. Section 4.3) and illustrated with a toy example (cf. Section 4.4)[1]. DEMOS-A is proven secure under the game-based definitions of E2E verifiability and voter privacy/PCR presented in Sections 3.3 and 3.4 respectively (cf. Sections 4.5 and 4.6). The chapter is concluded by commenting on the applicability of DEMOS-A, focusing on its usability, the importance of human factor for its security, while reckoning in its expected scalability limitations.

4.1 Overview of DEMOS-A

The DEMOS-A e-voting system has three phases, setup, ballot-casting and tallying, that parallel the operation of a Σ protocol (cf. Subsection 2.3.4.3).

During the setup phase, the EA produces a series of commitments and pre-audit data that

[1] The initial design of DEMOS-A [68] considered a centralised approach, where the entire election administration was managed by a single EA. The reason was that [68] prioritised the construction of an e-voting system that is E2E verifable in an all malicious setting. In this thesis, we present DEMOS-A according to its latest implementation version, where the vote collection and the tally phase are executed by the VC and the trustees, as modelled in Section 3.1.

correspond to a first round of a Σ protocol that will establish the validity of the commitments. Then, the EA provides the VC and the trustees with the necessary initialisation data. The working tape of the EA gets destroyed at the end of setup for privacy reasons.

During ballot-casting, voters engage with the VC in a protocol that will result in the recording of their votes, as well as in the submission of a *random coin flip* that will be used to produce the challenge for the Σ protocol. The voters will receive some individual audit information as their local output from the ballot-casting protocol for verifying the election result.

In the third and final phase, the trustees collectively compute the tally of the election and complete the Σ protocol by publishing openings to commitments as well as other post-audit data needed for verification. The verification step can take place at any time after the completion of the process using a collection of individual audit information from the ballot-casting phase.

DEMOS-A is designed s.t. the voter implementation during the ballot-casting phase can be expressed as a probabilistic transducer with a communication tape (cf. Definition 3.5) that has a number of states polynomial in the number of candidates m whereas it is independent to n, λ, as in the human node modelling described in 3.6.2. Given that such a machine is severely limited in the computational sense, in order to achieve ballot casting we utilise a code-voting approach (cf. [25]), the EA links vote-codes to commitments posted in the BB, and voters cast their votes by simply submitting to the VC the vote-code they prefer. The commitments have an additive homomorphic property, hence it is possible to tally the result by homomorphically processing them and opening the resulting "tally commitment". The proof that we use in order to ensure verifiability is a conjunction of a cut-and-choose proof with a Σ proof that a committed value belongs to a set. The challenge needed for the Σ proof will be extracted by applying a suitable extraction mechanism to the coin flips of the voter transducers that are collected by the VC.

4.2 Building Blocks of DEMOS-A

In this section, we present the cryptographic tools that DEMOS-A is built upon. Namely, (i) the ElGamal homomorphic commitment scheme, (ii) Shamir's secret sharing scheme (k-out-of-k version) [91] (iii) a generalisation of the Schwartz-Zippel lemma [100] for imperfect randomness, (iv) the Σ protocol for candidate encoding correctness and (v) the challenge extraction mechanism from the voters' coins.

4.2.1 ElGamal homomorphic commitment scheme

In order to achieve integrity against computationally unbounded adversaries, we utilise a perfectly binding commitment scheme. Additionally, DEMOS-A's design requires such a commitment scheme to be additively homomorphic to facilitate the tally and audit process. We address these requirements by instantiating the commitment scheme $\mathcal{CS}$ (cf.

Section 2.3.3) with lifted ElGamal over elliptic curves. We use elliptic curve domain parameters Param $= (p, a, b, g, q)$ produced by the curve generator $\mathcal{G}(1^\lambda)$, consisting of a prime p that specifies the finite field $\mathbb{F}_p$, two elements $a, b \in \mathbb{F}_p$ that specify an elliptic curve $E(\mathbb{F}_p)$ defined by the equation: $E : y^2 = x^3 + ax + b \pmod p$, a base point $g = (g_1, g_2)$ on $E(\mathbb{F}_p)$, and a prime q which is the order of g. We denote the cyclic group generated by g as $\mathbb{G}$, and assume that the DDH assumption holds for the group generator of $\mathbb{G}$. In detail, the ElGamal commitment scheme $\mathcal{EG}$ consists of the following algorithms:

- CS.Gen(1^λ):
  - Generate Param $= (p, a, b, g, q) \leftarrow \mathcal{G}(1^\lambda)$;
  - Choose $x \xleftarrow{\$} \mathbb{Z}_q$;
  - Output ck $\triangleq$ (Param, h) ;
- CS.Com(ck, $m; r$): Output $c = (g^r, g^m h^r)$;
- CS.Ver$\big($ck, $c, (m, r)\big)$: If $c = (g^r, g^m h^r)$, then output accept ; else, output reject ;

It is easy to check that $\mathcal{EG}$ is additively homomorphic. Namely,

$$\begin{aligned}\text{CS.Com}(\text{ck}, m_1; r_1) \cdot \text{CS.Com}(\text{ck}, m_2; r_2) = \\ = (g^{r_1} \cdot g^{r_2}, g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2}) = (g^{r_1+r_2}, g^{m_1+m_2} h^{r_1+r_2}) = \\ = \text{CS.Com}(\text{ck}, m_1 + m_2; r_1 + r_2) \ .\end{aligned}$$

Besides homomorphic additivity, $\mathcal{EG}$ inherits all the correctness and security properties of the ElGamal encryption scheme. Therefore, we claim that $\mathcal{EG}$ satisfies all three properties of a standard commitment scheme, omitting the proof.

Proposition 4.1. *The $\mathcal{EG}$ commitment scheme is correct, perfectly binding and computationally hiding, assuming the DDH assumption holds for the group generator of the cyclic group $\mathbb{G}$.*

4.2.2 (k, k)-Shamir's secret sharing scheme

We apply Shamir's linear SSS [91] to split and share the state of the EA to the trustees during the setup phase. As already mentioned, fault tolerance is out of scope in our work, therefore we may restrict to the simpler k-out-of-k version. In (k, k)-Shamir's SSS, the dealer (EA) shares a secret s by executing the following steps:

1. It chooses a large prime p, so that s, encoded as an integer is in $\mathbb{Z}_p$.
2. It determines the polynomial $f(x) = a_{k-1}x^{k-1} + a_1 x + a_0$ of degree $k - 1$, by setting $a_0 = s$ and choosing $a_1, \ldots, a_{k-1}$ arbitrarily.
3. It sets the share $\|s\|_i$ of the i-th player (trustee T_i) to be the value $f(i)$.

By polynomial interpolation, the secret s can be recovered only from all the k shares, as k polynomial are necessary and sufficient for finding the coefficients of f. In addition, linearity is straightforward from the addition within the polynomial ring $\mathbb{Z}_p[x]$.

4.2.3 Schwartz-Zippel (min-entropy variant)

Recall (cf. Definition 2.4) that the *min-entropy* of a finite random variable X, $H_\infty(X)$, is defined as

$$H_\infty(X) = -\log(\max_x \mathsf{Pr}[X = x]) \ .$$

We show that the following min-entropy variant of the Schwartz-Zippel lemma holds. A similar variant with average conditional min-entropy can be also found in [48].

Lemma 4.1 (***min-entropy Schwartz-Zippel***)**.** *Let $f(x)$ be a non-zero univariate polynomial of degree d over $\mathbb{Z}_q$. Let X be a r.v. following probability distribution* **D** *on $\mathbb{Z}_q$ such that $H_\infty(X) \geqslant \kappa$. Then, the probability that $f(x) = 0$, where x is a sampled according to* **D***, is at most $d2^{-k}$.*

Proof. Any univariate polynomial f of degree d over $\mathbb{Z}_q$ has at most $r \leqslant d$ roots $x_1, \ldots, x_r$. Hence, given $H_\infty(X) \geqslant \kappa$, we conclude that

$$\mathsf{Pr}[x \xleftarrow{\mathbf{D}} \mathbb{Z}_q : f(x) = 0] = \sum_{i \in [r]} \mathsf{Pr}[X = x_i] \leqslant r2^{-k} \leqslant d2^{-k} \ .$$

■

Lemma 4.1 will be apllied for proving the soundness of the Σ protocol described in the following subsection, for the equlatity check of two univariate polynomials f_1, f_2, i.e. test $f_1(x) - f_2(x) = 0$ for random $x \xleftarrow{\mathbf{D}} \mathbb{Z}_q$.

4.2.4 A Σ protocol for candidate encoding correctness

In order to present the Σ protocol with clarity, we outline some necessary excerpts of the description of DEMOS-A that will be explained in detail in Section 4.3.

Each of the n voters is given a ballot that consists of two equivalent parts that contain a list of m vote-codes corresponding to the option list $\{\mathsf{opt}_1, \ldots, \mathsf{opt}_m\}$. The voter flips a coin to choose the part she is going to use for voting (this idea was proposed in [27]). At the setup phase, each ballot is posted in the BB in committed form. Namely, it consists of two sets of commitments $E^{(a)}_{\ell,j}$ for $a \in \{0, 1\}, j \in [m], \ell \in [n]$, and each set commits to a permutation of the encoded options, where option opt_j is encoded as $(n + 1)^{j-1}$.

We emphasise that it is not necessary to prove that each set of the commitments commits to a permutation of the encoded options $\{(n + 1)^0, \ldots, (n + 1)^{m-1}\}$ in an 1-out-of-m

election. This is due to two facts: (i) EA will open one of the two sets of commitments according to the corresponding voter's coin a_ℓ (the set that corresponds to the unused ballot part); therefore, a malicious EA will be caught with probability $1/2$ by each honest voter if any of the committed sets is not a permutation of the encoded options or is an inconsistent permutation of the encoded options w.r.t. the one on the voter's ballot, and (ii) even if we ensure that the set of the commitments commits to a permutation of the encoded options, this does not imply that the permutation is consistent with the one on the voter's ballot. However, in an 1-out-of-m election, only one of the commitments will be used for tally; thus, proving that the set of the commitments commits to an unknown permutation of the encoded options can only provide the guarantee that the tallied commitment commits to an encoded option. Note that this guarantee is important; otherwise, given that we perform homomorphic tallying, it may be feasible for a cheating EA to introduce a large deviation to the actual tally result via a single inconsistent ballot; for instance, EA may commit to $10000 \cdot (n+1)^{j-1}$ for some $j \in [m]$ and thus inject 10000 votes for opt_j.

1ST ROUND: COMMITMENT

$\mathcal{P}(E, j, r)$**:**

- Define b_i s.t. $j = \sum_{i=0}^{\log m - 1} b_i 2^i$;
- For $i = 0, \ldots, \log m - 1$, pick $t_i, z_i, y_i, r_i, w_i, f_i \leftarrow \mathbb{Z}_q$;
- For $i = 0, \ldots, \log m - 1$, compute the following commitments:

  $B_i = \mathsf{CS.Com}(\mathsf{ck}, b_i; r_i)$; $\quad T_i = \mathsf{CS.Com}(\mathsf{ck}, t_i; z_i)$;

  $Y_i = \mathsf{CS.Com}(\mathsf{ck}, (1-b_i)t_i; y_i)$; $\quad W_i = \mathsf{CS.Com}(\mathsf{ck}, w_i; f_i)$;

- For $i = 0, \ldots, \log m - 1$, define A_i, a_i, r'_i s.t. $A_i = B_i^{(n+1)^{2^i}-1} \cdot \mathsf{CS.Com}(\mathsf{ck}, 1; 0) = \mathsf{CS.Com}(\mathsf{ck}, a_i; r'_i)$;
- Define $\{\beta_i, \gamma_i\}_{i=0}^{\log m}$ s.t. $\prod_{i=0}^{\log m - 1}(a_i X + w_i) = \sum_{i=0}^{\log m} \beta_i X^i$ and $\prod_{i=0}^{\log m - 1}(r'_i X + f_i) = \sum_{i=0}^{\log m} \gamma_i X^i$;
- For $i = 0, \ldots, \log m - 1$, set $D_i = \mathsf{CS.Com}(\mathsf{ck}, \beta_i; \gamma_i)$;

Return $\phi_1 = \{B_i, T_i, Y_i, W_i, D_i\}_{i=0}^{\log m - 1}$ and

$\text{state}_\phi = \{t_i, z_i, y_i, r_i, b_i, w_i, f_i\}_{i=0}^{\log m - 1}$.

$\mathcal{P} \longrightarrow \mathcal{V}$**:** Send ϕ_1.

2ND ROUND: CHALLENGE

$\mathcal{V}(E, \phi_1)$**:** Pick $\rho \leftarrow \mathbb{Z}_q$.

$\mathcal{V} \rightarrow \mathcal{P}$**:** Send ρ.

3RD ROUND: RESPONSE

$\mathcal{P}(E, j, r, \mathsf{state}_\phi)$**:**

▸ For $i = 0, \ldots, \log m - 1$, compute the following responses:

$$t'_i = b_i\rho + t_i, z'_i = r_i\rho + z_i, y'_i = -y_i - r_i t'_i \; ; \quad w'_i = a_i\rho + w_i, f'_i = r'_i\rho + f_i \; ;$$

$$\text{Set } \phi_2 = \{t'_i, z'_i, y'_i, w'_i, f'_i\}_{i=0}^{\log m - 1}.$$

$\mathcal{P} \to V$**:** Send ϕ_2.

VERIFICATION

$\mathcal{V}(E, \phi_1, \rho, \phi_2)$**:** Output accept iff

1. For $i = 0, \ldots, \log m - 1$,
   - $B_i^\rho \cdot T_i = \mathsf{CS.Com}(\mathsf{ck}, t'_i, z'_i)$,
   - $(\mathsf{CS.Com}(\mathsf{ck}, 1; 0)/B_i)^{t'_i}/Y_i = \mathsf{CS.Com}(\mathsf{ck}, 0; y'_i)$;
   - $A_i^\rho \cdot W_i = \mathsf{CS.Com}(\mathsf{ck}, w'_i, f'_i)$;
2. $E^{\rho^{\log m}} \prod_{i=0}^{\log m - 1} D_i^{\rho^i} = \mathsf{CS.Com}(\mathsf{ck}, \prod_{i=0}^{\log m - 1} w'_i; \prod_{i=0}^{\log m - 1} f'_i)$;

Figure 4.1: The Σ Protocol $\langle \mathcal{P}(j, r), \mathcal{V} \rangle(E)$ for candidate encoding correctness.

By the above, we require that the EA proves that each commitment commits to one of $(n+1)^{j-1}$ for $j \in [m]$[2]. We formalise the correctness of a single commitment problem as follows. Given a commitment E, the prover wants to convince the verifier that it knows $r \in \mathbb{Z}_q$ such that $E = \mathsf{CS.Com}(\mathsf{ck}, (n+1)^j; r)$ and $j \in \{0, \ldots, m-1\}$. Let (j, r) be the prover's private input, and w.l.o.g. we assume m is a power of 2. For general cases, say $2^t \leqslant m < 2^{t+1}$, we can show that $j \in \{0, \ldots, m-1\}$ via the conjunction $j \in \{0, \ldots, 2^{t+1} - 1\} \wedge (j + 2^t - m) \in \{0, 2^t\}$. Our Σ protocol is described in Figure 4.1. Note that in the 1st round, for efficiency reasons, the prover needs to choose the $\{r_i\}_{i=0}^{\log m - 1}$ such that $\gamma_{\log m} = r$ in previous step. The properties of the Σ protocol are proven in the following theorem.

Theorem 4.1. *If the verifier's challenge has min-entropy κ, then the protocol described in Figure 4.1, is a Σ protocol for knowledge of $j \in \{0, \ldots, m-1\}$ and $r \in \mathbb{Z}_q$ such that $E = \mathsf{CS.Com}(\mathsf{ck}, (n+1)^j; r)$ that achieves*

***1.** perfect completeness,*

[2]For efficiency, EA is only required to show the commitments used for tally commit to valid encoded options. On the other hand, since EA cannot predict which commitments are going to be used for tally before the election, she has to prepare all the Σ protocols in the setup phase, whereas she is only required to complete those Σ protocols for the commitment that will be tallied in the tally phase.

2. *statistical soundness with soundness error* $2^{-\kappa+1+\log\log m}$,

3. *special soundness, and*

4. *computational sHVZK with distinguishing advantage* $\mathsf{Adv}_{\mathsf{zk}}(\mathcal{A}) \leqslant \log m \cdot \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$ *for any PPT adversary* $\mathcal{A}$*, where* $\mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$ *is the distinguishing advantage of* $\mathcal{A}$ *for breaking the hiding property of the ElGamal commitment scheme.*

Proof.
1. It is straightforward to check that protocol in Figure 4.1 achieves perfect completeness.

2. In terms of statistical soundness, the protocol verifies two facts. Namely,

(a). $\{B_i\}_{i\in[0,\log m-1]}$ commits to either 0 or 1, and

(b). E commits to $(n+1)^{\sum_{i=0}^{\log m-1} b_i 2^i} = (n+1)^j$, where b_i is the opening of B_i.

To check the first fact, for each committed b_i and for some c_0 and c_0', the protocol builds the degree 1 polynomial

$$g_1(X) = (1-b_i)(b_iX+t) + c_0 = (1-b_i)b_iX + c_0' \ .$$

By min-entropy Schwartz-Zippel Lemma 4.1, if $H_\infty(\rho) \geqslant \kappa$ and $g_1(\rho) = 0$, then the probability $\Pr[(1-b_i)b_i \neq 0]$ is no more than $2^{-\kappa}$. Hence, with at least $1-2^{-\kappa}$ probability, we have that $(1-b_i)b_i = 0$, which implies $b_i \in \{0,1\}$.

To check the second fact, the protocol first computes $A_i = B_i^{(n+1)^{2^i}-1} \cdot \mathsf{CS.Com}(\mathsf{ck}, 1; 0)$ homomorphically. Let a_i be the opening of A_i. It is easy to see that $a_i = (n+1)^{2^i}$ if $b_i = 1$, whereas $a_i = 1$ if $b_i = 0$. Hence, it holds that

$$a_i = b_i(n+1)^{2^i} + 1 - b_i = (n+1)^{b_i 2^i} \ .$$

Consequently, the protocol just needs to verify that E commits to the product of a_i's. The verifier checks equality between two degree $\log m$ polynomials

$$g_2(X) = \prod_{i=0}^{\log m-1} (a_iX + w_i) = \sum_{i=0}^{\log m} \beta_i X^i \text{ and } g_2'(X) = uX^{\log m} + \sum_{i=0}^{\log m-1} \beta_i^* X^i,$$

where u is the opening of E and β_i^* is the opening of D_i both provided by the (potentially malicious) prover. By min-entropy Schwartz-Zippel lemma, if $H_\infty(\rho) \geqslant \kappa$ and $g_2(\rho) = g_2'(\rho)$, then the probability $\Pr[u = \beta_{\log m}]$ is at least $1 - \log m \cdot 2^{-\kappa}$. Hence, we have that

$$u = (n+1)^{\sum_{i=0}^{\log m-1} b_i 2^i}$$

with at least $1-\log m \cdot 2^{-\kappa}$ probability conditioned on the first fact. Given that all $b_0, \ldots, b_{\log m-1}$ need to be shown in $\{0,1\}$ the entire proof is statistically sound with probability

$$(1-2^{-\kappa})^{\log m}(1-\log m \cdot 2^{-\kappa}) \geqslant 1-\log m \cdot 2^{-\kappa+1} \ .$$

3. Furthermore, the protocol satisfies special soundness, i.e. there exists an extractor that can extract $i \in \mathbb{N}, r \in \mathbb{Z}_q$ if the prover is able to complete the protocol twice with the same ϕ_1 but two distinct challenges but two distinct challenges $\rho^{(1)}$ and $\rho^{(2)}$. Such an extractor is constructed as follows.

(i). For $e \in \{1,2\}$, for $i \in \{-1, 0, \ldots, \log m-1\}$, given $t_i'^{(e)} = b_i\rho^{(e)} + t_i$ and $z_i'^{(e)} = r_i\rho^{(e)} + z_i$, the extractor computes

$$b_i = \frac{t_i'^{(2)} - t_i'^{(2)}}{\rho^{(1)} - \rho^{(2)}} \quad \text{and} \quad r_i = \frac{z_i'^{(2)} - z_i'^{(2)}}{\rho^{(1)} - \rho^{(2)}} \ .$$

(ii). The extractor then outputs

$$b = b_{-1}, i = \sum_{i=0}^{\log m-1} b_i 2^i \quad \text{and} \quad r = r_{-1} \cdot \prod_{i=0}^{\log m-1} \left(r_i^{(n+1)^{2^i}} - r_i \right) \ .$$

4. To show the special HVZK property, we construct a simulator $\mathcal{S}$ that on input $\hat{\rho} \in \mathbb{Z}_q$ can output a transcript that is indistinguishable from the real one. The simulator randomly picks $b_0, \ldots, b_{\log m-1} \leftarrow \{0,1\}$ and generates

$$\{t_i, z_i, y_i, r_i, B_i, T_i, Y_i, t_i', z_i', y_i', w_i, f_i, W_i, w_i', f_i'\}_{i=0}^{\log m-1}$$

according to the protocol description. Then, it generates $\{D_i\}_{j=i}^{\log m-1}$ also according to the protocol and sets

$$D_0 = \mathsf{CS.Com}\Big(\mathsf{ck}, \prod_{i=0}^{\log m-1} w_i'; \prod_{i=0}^{\log m-1} f_i'\Big) \Big/ \Big(E^{\hat{\rho}^{\log m}} \prod_{i=1}^{\log m-1} D_i^{\hat{\rho}^i}\Big) \ .$$

Subsequentely, $\mathcal{S}$ sets $\hat{\phi}_1 = \{B_i, T_i, Y_i, W_i, D_i\}_{j=0}^{\log m-1}$ and $\hat{\phi}_2 = \{t_i', z_i', y_i', w_i', f_i'\}_{i=0}^{\log m-1}$, and it outputs $(\hat{\phi}_1, \hat{\rho}, \hat{\phi}_2)$. It is obvious that all the verification equations hold. Secondly, the distribution of all the variables in $\hat{\phi}_2$ are uniformly random, which is identical to that of a real transcript. Moreover, if the adversary can distinguish the simulated $\hat{\phi}_1$ from that of a real transcript, she must be able to distinguish at least one of the fake $\{B_i\}_{i=0}^{\log m-1}$. By a standard hybrid argument, for any PPT adversary $\mathcal{A}$, the advantage to distinguish the simulated proof is $\mathsf{Adv}_{\mathsf{zk}}(\mathcal{A}) \leqslant \log m \cdot \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$. ■

4.2.5 Producing the Verifier's Challenges

The main difficulty in DEMOS-A's setting is that we would like to extract the challenge of the Σ protocol from the voters' coins, denoted by $\mathbf{a} = \langle a_1, \ldots, a_n \rangle \in \{0,1\}^n$, using a

deterministic algorithm. Recall that some of the voters might be malicious and colluding with the EA. As a result, the entropy of the voters' coins is only contributed by the honest voters while the malicious voters' coins can depend on the honest ones. Further, note that the voters' coins should be ordered by their serial numbers, rather than their submission order. This is because in the latter case, the adversary can schedule the **Cast** protocols of all voters at will, thus may reduce the min-entropy of **a** to be at most $\log \theta$ where θ is the number of honest voters. Such level of entropy is insufficient to provide a sufficiently small verifiability error (i.e., that ideally drops exponentially with θ). For all the uncast ballots, we set their corresponding coins to 0 by default; therefore, **a** is always an n-bit source, regardless of the number of voters that complete the **Cast** protocol.

The voters' coins as a source of randomness

We observe that the voters' coins **a** is a weaker source compared to a *non-oblivious bit-fixing source* [66], as the adversary is able to choose which bit(s) to fix during the coin flipping (source generation) process. On the other hand, if we restrict the adversary $\mathcal{A}$ in our verifiability game from being capable of scheduling **Cast** protocols freely and all voters have to submit their votes sequentially according to a pre-determined order in the ballot casting stage, the source **a** can be viewed as an *adaptive bit-fixing source* [75]; in such case, we can employ the deterministic extractor construction framework from [66] which applies a deterministic low influence function on segments of the source. The *majority function* is proven to be an optimal low influence function thus in this way we obtain a deterministic extractor that generates the challenge. However, this adversarial setting is not realistic in practice as ballot casting might be scheduled adversarially. Nevertheless, we emphasise that even using a non-oblivious bit-fixing source, Kamp and Zuckerman showed that at most n/ℓ bits can be extracted when ℓ out of n bits are fixed [66]. This result implies that if a deterministic extractor is used to generate $\Theta(\lambda)$ random bits, then this will restrict the percentage of corrupted voters to be below $\Theta(\frac{1}{\lambda})$ which might also be not a realistic expectation in practice.

An alternative approach may use a *condenser* as opposed to an extractor. Randomised condensers with a small/constant seed space have been put forth see e.g. [7, 87]; using such a tool one may iterate over all possible seeds and thus be assured that one of the seeds will allow the condenser to produce a sufficiently random challenge. For instance, Barak *et al.* [7] proposed a basic 2-bit seed condenser $\text{Con} : \{0,1\}^n \rightarrow (\{0,1\}^{n/3})^4$ such that for every δ-source X with $0 < \delta < 0.9$, at least one of the 4 output blocks of $\text{con}(X)$ is a $(\delta + \Omega(\delta^2))$-source. Based on the composing lemma ([7, Lemma 5.5]), we can iteratively apply the condenser to achieve any desired constant rate. Given a c-coin condenser $\text{Con} : \{0,1\}^n \longmapsto (\{0,1\}^{\ell})^c$, in order to produce a good challenge, by definition, it should hold that $c \cdot \ell > n$, which means that the condenser will produce c blocks, one of which is guaranteed to be sufficiently random. However as we observe below, we can utilise *ZK* amplification to obtain essentially the same result as with a c-coin condenser by sacrificing very little entropy from the weak source. We explain our technique below.

Let q be the order of the underlying group used in the Σ protocol, and let $\{0,1\}^{\ell_\Sigma}$ be the

challenge space, where $\ell_\Sigma = \lfloor \log q \rfloor$. Assume $n/k \leqslant \ell_\Sigma$ for some $k \in \mathbb{Z}^+$. We evenly partition the voters' coins $\mathbf{a}$ into k blocks, denoted by $\mathbf{a}_1, \ldots, \mathbf{a}_k$. For each block $\mathbf{a}_i$, the EA should prove the correctness of the ballots using a separate Σ protocol with $\mathbf{a}_i$ as its challenge. The verifier only accepts the EA's proof if *all* the Σ protocols are valid. The theorem below shows that the soundness error of this k-times repeated Σ protocol drops exponentially with $\theta - k(\log\log m + 1)$.

Theorem 4.2. *Let q be the order of the underlying group used in the Σ protocol described in Figure 4.1, and let $\lfloor \log q \rfloor$ be the challenge space. Assume that the voters' coins $\mathbf{a} = \langle a_1, \ldots, a_n \rangle \in \{0,1\}^n$ are partitioned in k blocks $\mathbf{a}_1, \ldots, \mathbf{a}_k$, where $n/k \leqslant \lfloor \log q \rfloor$. If $H_\infty(\mathbf{a}) = \theta$, then for all adversarial provers $\mathcal{A}$, we have that*

$$\epsilon(m,n,k,\theta) = \mathsf{Pr}\left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Gen}(\mathsf{Param}, 1^\lambda); (E, x, r, \{\phi_{1,i}\}_{i=1}^k) \leftarrow \mathcal{A}(\mathsf{Param}, \mathsf{ck}); \\ \{\phi_{2,i}\}_{i=1}^k \leftarrow \mathcal{A}(\mathbf{a}_1, \ldots, \mathbf{a}_k) : \\ \mathsf{CS.Ver}\big(\mathsf{ck}, E, (x,r)\big) = 1 \ \wedge x \notin \{(n+1)^0, \ldots, (n+1)^{m-1}\} \ \wedge \\ \wedge \ \forall i \in [k] : \mathsf{accept} \leftarrow \mathcal{V}(E, \phi_{1,i}, \mathbf{a}_i, \phi_{2,i}) \end{array}\right] \leqslant$$
$$\leqslant 2^{k \log\log m - \theta + k} .$$

Proof. According to Theorem 4.1, for each challenge $\mathbf{a}_i$, the Σ protocol described in Figure 4.1 is statistically sound with soundness error $\log m \cdot 2^{-H_\infty(\mathbf{a}_i)+1}$. Hence, for each challenge $\mathbf{a}_i, i \in [k]$,

$$\mathsf{Pr}\left[\begin{array}{l} \mathsf{CS.Ver}\big(\mathsf{ck}, E, (x,r)\big) = 1 \ \wedge \ x \notin \{n^0, \ldots, n^{m-1}\} \ \wedge \\ \wedge \ \mathsf{accept} \leftarrow \mathcal{V}(E, \phi_{1,i}, \mathbf{a}_i, \phi_{2,i}) \end{array}\right] \leqslant 2^{\log\log m - H_\infty(\mathbf{a}_i)+1} .$$

Therefore, we have the overall soundness error

$$\epsilon(m,n,k,\theta) = \mathsf{Pr}\left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Gen}(\mathsf{Param}, 1^\lambda); (E, x, r, \{\phi_{1,i}\}_{i=1}^k) \leftarrow \mathcal{A}(\mathsf{Param}, \mathsf{ck}); \\ \{\phi_{2,i}\}_{i=1}^k \leftarrow \mathcal{A}(\mathbf{a}_1, \ldots, \mathbf{a}_k) : \\ \mathsf{CS.Ver}\big(\mathsf{ck}, E, (x,r)\big) = 1 \ \wedge x \notin \{(n+1)^0, \ldots, (n+1)^{m-1}\} \ \wedge \\ \wedge \ \forall i \in [k] : \mathsf{accept} \leftarrow \mathcal{V}(E, \phi_{1,i}, \mathbf{a}_i, \phi_{2,i}) \end{array}\right] \leqslant$$
$$\leqslant \prod_{i=1}^{k} 2^{\log\log m - H_\infty(\mathbf{a}_i)+1} = 2^{k\log\log m - \sum_{i=1}^k H_\infty(\mathbf{a}_i) + k} =$$
$$= 2^{k\log\log m - H_\infty(\mathbf{a}) + k} \leqslant 2^{k\log\log m - \theta + k} .$$

■

4.3 Description of DEMOS-A

The description of DEMOS-A follows the syntax in Section 3.1. For simplicity, we present the system for *1-out-of-m elections*, i.e. $\mathcal{U} = \{\{\mathsf{opt}_1\}, \ldots, \{\mathsf{opt}_m\}\}$. The commitment scheme, the SSS and the Σ-protocol that are applied in our system, are the ones presented at length in Subsections 4.2.1, 4.2.2 and 4.2.4 respectively. DEMOS-A does require that

the voter performs computations beyond human level (e.g. cryptographic operations) for vote casting, therefore we do not associate VSDs with the voters, which can be seen as a first step towards the ceremony framework.

The **Setup**$(1^\lambda, \mathcal{O}, \mathcal{V}, \mathcal{U}, \mathcal{T})$ *protocol* :

Let $(\mathsf{CS.Gen}, \mathsf{CS.Com}, \mathsf{CS.Ver})$ be the PPT algorithms that constitute the ElGamal commitment scheme presented in Section 4.2.1. The EA runs $\mathsf{CS.Gen}(1^\lambda)$ to generate the commitment key ck. Then, for $\ell \in [n]$, EA executes the following steps:

Ballot generation and distribution:

1. It selects a unique label for the ℓ-th double ballot denoted by tag_ℓ.
2. It selects random permutations $\pi_\ell^{(0)}, \pi_\ell^{(1)}$ over $[m]$. The use of $\pi_\ell^{(0)}$ (reps. $\pi_\ell^{(1)}$) is to shuffle the order that the (vote-code, option) pairs in the part $\mathbf{B}_\ell^{(0)}$ (resp. $\mathbf{B}_\ell^{(1)}$) of the *double ballot* $\mathbf{B}_\ell$ will be posted on the BB (in committed form), in order to support privacy.
3. For $j \in [m]$, it selects unique vote-codes $C_{\ell,j}^{(0)}, C_{\ell,j}^{(1)} \leftarrow \mathbb{Z}_q$, where q is the size of the group of the commitment scheme [3]. The vote-code $C_{\ell,j}^{(0)}$ (resp. $C_{\ell,j}^{(1)}$) is the one that will be associated with option opt_j in part $\mathbf{B}_\ell^{(0)}$ (resp. $\mathbf{B}_\ell^{(1)}$) of $\mathbf{B}_\ell$.
4. For $a \in \{0,1\}$, it prepares the ballot part $\mathbf{B}_\ell^{(a)} = \left\{ \left(\mathsf{opt}_j, C_{\ell,j}^{(a)} \right) \right\}_{j \in [m]}$ and generates the double ballot that we denote by $\boxed{\mathbf{B}_\ell = \left(\mathsf{tag}_\ell, \mathbf{B}_\ell^{(0)}, \mathbf{B}_\ell^{(1)} \right)}$ and is delivered to voter V_ℓ as credential cr_ℓ in the form below:

tag_ℓ	
Vote-code	**Option**
$C_{\ell,1}^{(0)}$	opt_1
$\vdots$	$\vdots$
$C_{\ell,m}^{(0)}$	opt_m
$C_{\ell,1}^{(0)}$	opt_1
$\vdots$	$\vdots$
$C_{\ell,m}^{(0)}$	opt_m

The double ballot $\mathbf{B}_\ell$ in DEMOS-A.

[3] For simplicity in presentation, we commit to the vote-codes using the homomorphic commitment scheme of Section 4.2.1. We stress that since no arithmetic operations are executed in the vote-code commitments, we could use more efficient commitment schemes and in this case vote-codes may be drawn from a domain that is smaller than $\mathbb{Z}_q$ resulting in a more "user-friendly" interface.

Public information preparation:

5. For $j \in [m]$, EA computes $j' = \pi_\ell^{(a)}(j)$ and

   (i). For $a \in \{0,1\}$, it chooses randomness $t_{\ell,j'}^{(a)} \xleftarrow{\$} \mathbb{Z}_q$ and computes the *vote-code commitment* for $C_{\ell,j'}^{(a)}$:

   $$\boxed{U_{\ell,j'}^{(a)} = \mathsf{CS.Com}_{\mathsf{ck}}\big(C_{\ell,j'}^{(a)}; t_{\ell,j'}^{(a)}\big)}\,.$$

   (ii). For $a \in \{0,1\}$, it chooses randomness $r_{\ell,j'}^{(a)} \xleftarrow{\$} \mathbb{Z}_q$ and computes the *encoded option commitment* for $\mathsf{opt}_{j'}$:

   $$\boxed{E_{\ell,j'}^{(a)} = \mathsf{CS.Com}_{\mathsf{ck}}\big((n+1)^{j'-1}; r_{\ell,j'}^{(a)}\big)}\,,$$

   where $(n+1)^{j'-1}$ is the *encoding* of option $\mathsf{opt}_{j'}$. This encoding is selected to ensure the correctness of our system, as we show in Theorem 4.3.

   (iii). For $a \in \{0,1\}$, EA prepares *pre-audit data* $\phi_{1,\ell,j'}^{(a)}$ to be used for verifying that $E_{\ell,j'}^{(a)}$ is a commitment to a valid encoding from the set $\{(n+1)^0, \ldots, (n+1)^{m-1}\}$ at the verification phase. In addition, it maintains *prover state* $\mathsf{state}_{\phi,\ell,j'}^{(a)}$. Both $\phi_{1,\ell,j'}^{(a)}$ and $\mathsf{state}_{\phi,\ell,j'}^{(a)}$ are described in the Σ-protocol shown in Figure 4.1 (1st round) of Subsection 4.2.4.

Pre-election BB data:

6. The public information w.r.t. $\mathbf{B}_\ell$ is $\mathsf{Pub}_\ell \triangleq \left(\mathsf{tag}_\ell, \left\{(U_{\ell,j'}^{(a)}, E_{\ell,j'}^{(a)}, \phi_{1,\ell,j'}^{(a)})\right\}_{j\in[m]}^{a\in\{0,1\}}\right)$. It is indexed by tag_ℓ and contains the ballot information for both parts in committed form, as well as the respective pre-audit data. The information that refers to the (vote-code, option) pair $(C_{\ell,j'}^{(a)}, \mathsf{opt}_{j'})$ is tabulated in the j-th location of the part that is associated with $\mathbf{B}_\ell^{(a)}$.

7. The public information that EA generates and posts in the BB is

   $$\boxed{\mathsf{Pub} \triangleq \big(\mathsf{ck}, \mathcal{P}, \mathcal{U}, \{\mathsf{Pub}_\ell\}_{\ell\in[n]}\big)}\,.$$

Trustees' private inputs:

8. The state of EA is

   $$\boxed{\mathsf{st}_{\mathsf{ea}} \triangleq \{\mathsf{Pub}_\ell, \mathbf{B}_\ell, msk_\ell, \mathsf{state}_{\phi,\ell}\}_{\ell\in[n]}}\,,$$

   where we denote $msk_\ell \triangleq \left\{\left(C_{\ell,j}^{(a)}, t_{\ell,j}^{(a)}, \pi_\ell^{(a)}(j), r_{\ell,j}^{(a)}\right)\right\}_{j\in[m]}^{a\in\{0,1\}}$

   and $\mathsf{state}_{\phi,\ell} \triangleq \left\{\mathsf{state}_{\phi,\ell,j'}^{(a)}\right\}_{j\in[m]}^{a\in\{0,1\}}$.

9. The EA uses (k, k)-Shamir secret sharing as described in Subsection 4.2.2 to split its state into the shares $\|\mathsf{st}_{\mathsf{ea}}\|_1, \ldots, \|\mathsf{st}_{\mathsf{ea}}\|_k$, each consisting of shares of every value in $\mathsf{st}_{\mathsf{ea}}$ tabulated consistently. Namely, for $i \in [k]$

$$\boxed{\|\mathsf{st}_{\mathsf{ea}}\|_u \triangleq \{\|\mathsf{Pub}_\ell\|_u, \|\mathbf{B}_\ell\|_u, \|msk_\ell\|_u, \|\mathsf{state}_{\phi,\ell}\|_u\}_{\ell\in[n]}},$$

where $\mathsf{Pub}_\ell\|_u, \|\mathbf{B}_\ell\|_u, \|msk_\ell\|_u, \|\mathsf{state}_{\phi,\ell}\|_u$ are vectors that contain the shares of values tabulated consistently with $\mathsf{Pub}_\ell, \mathbf{B}_\ell, msk_\ell, \mathsf{state}_{\phi,\ell}$. Then, EA provides each trustee T_u, $u \in [k]$ with the share $\|\mathsf{st}_{\mathsf{ea}}\|_u$.

At the end of the setup phase, the working tape of the EA is destroyed, thus its state is erased.

The **Cast** *protocol* :

On input $(\mathsf{Pub}, \mathbf{B}_\ell, \mathcal{U}_\ell)$, voter V_ℓ flips a coin $a_\ell \leftarrow \{0, 1\}$ and picks part $\mathbf{B}_\ell^{(a_\ell)}$ to vote and part $\mathbf{B}_\ell^{(a_\ell)}$ for audit. Let opt_{j_ℓ} be the option that V_ℓ is going to vote for, i.e., $\mathcal{U}_\ell = \{\mathsf{opt}_{j_\ell}\}$. Then, V_ℓ selects to submit $C_{\ell,j_\ell}^{(a_\ell)}$, which is the vote-code that corresponds to opt_{j_ℓ} in part $\mathbf{B}_\ell^{(a_\ell)}$. Next, V_ℓ casts the vote

$$\boxed{\psi_\ell \triangleq \left(\mathsf{tag}_\ell, a_\ell, C_{\ell,j_\ell}^{(a_\ell)}\right)}.$$

The VC receives the vote and updates its state $\mathsf{st}_{\mathsf{vc}}$ by appending ψ_ℓ. The individual audit data audit_ℓ of V_ℓ is

$$\boxed{\mathsf{audit}_\ell \triangleq \left(\psi_\ell, a_\ell, \mathbf{B}_\ell^{(1-a_\ell)}\right)}.$$

The **Tally** *protocol* :

Let $\mathcal{V}_{\mathsf{succ}}$ be the set of the voters that have voted successfully. The tally phase proceeds as follows.

1. The VC posts the set of submitted votes $\mathsf{V}_{\mathsf{tally}} = \{\psi_\ell\}_{V_\ell\in\mathcal{V}_{\mathsf{succ}}}$ in the BB.

2. For each $\psi_\ell \in \mathsf{V}_{\mathsf{tally}}$, every trustee T_u reads $\left(\mathsf{tag}_\ell, C_{\ell,j_\ell}^{(a_\ell)}\right)$ from the BB. Then, it provides its share of openings to all the vote-code commitments, $\left\{U_{\ell,j}^{(a)}\right\}_{\ell\in[n],j\in[m]}^{a\in\{0,1\}}$, by posting the list $\left\{\left(\|C_{\ell,j}^{(a)}\|_u, \|t_{\ell,j}^{(a)}\|_u\right)\right\}_{\ell\in[n],j\in[m]}^{a\in\{0,1\}}$ in the BB. After all shares have been published, every party can recover the decommitted vote-codes and the decommitted ballot parts used for auditing.

3. The VC, for every ψ_ℓ corresponding to a $V_\ell \in \mathcal{V}_{\mathsf{succ}}$:

(i). Locates the decommitted vote-code C_ℓ that matches the cast vote-code $C^{(a_\ell)}_{\ell,j_\ell}$. Then, it marks the vote-code C_ℓ as 'voted' and adds the corresponding commitment $E^{(a_\ell)}_{\ell,j'_\ell}$ into the set $\mathbf{E}_{\text{tally}}$ (initially empty). Recall that $j'_\ell = \pi^{(a_\ell)}_\ell(j_\ell)$. The set of marked vote-codes is denoted by $\mathbf{C}_{\text{cast}}$.

(ii). Adds all the commitments $\{E^{(1-a_\ell)}_{\ell,j}\}_{j\in[m]}$ that correspond to the vote-codes in $\mathbf{B}^{(1-a_\ell)}_\ell$ into the set $\mathbf{E}_{\text{open}}$ (initially empty).

When finalised, $\mathbf{E}_{\text{tally}}$ includes the collection of votes that will be counted (homomorphically) and $\mathbf{E}_{\text{open}}$ includes the information that will be used for verifying ballot correctness. After this happens, VC posts in the BB the list of marked vote-codes along with $\mathbf{E}_{\text{tally}}$ and $\mathbf{E}_{\text{open}}$.

4. The VC produces and posts in the BB all the *verifier's challenges* $\{\rho_E\}_{E\in\mathbf{E}_{\text{tally}}}$ of the Σ-protocols for the validity of the commitments in $\mathbf{E}_{\text{tally}}$, as determined in Figure 4.1 (2nd round). The extraction of the challenges is done via the randomness contributed by the voters' coin-flips according to the method that is described in Subsection 4.2.5. We denote by ρ the extracted challenge.

5. Every trustee T_u prepares and posts in the BB its share of *post-audit data* $\{\phi_{2,E}\}_{E\in\mathbf{E}_{\text{tally}}}$ of the Σ-protocols for verifying the validity of the commitments in $\mathbf{E}_{\text{tally}}$, as determined in Figure 4.1 (3rd round). In particular, by parsing $\|\text{st}_{\text{ea}}\|_u$ it obtains $\|t_{i,E}\cdot r_{i,E}\|_u, \|z_{i,E}\|_u, \|y_{i,E}\|_u, \|r_{i,E}\|_u, \|b_{i,E}\|_u, \|w_{i,E}\|_u, \|f_{i,E}\|_u$, for $i = 0,\ldots,\log m - 1$ and $E \in \mathbf{E}_{\text{tally}}$. Then, for $i = 0,\ldots,\log m - 1$ and $E \in \mathbf{E}_{\text{tally}}$, it computes the shares for the responses $t'_{i,E}, z'_{i,E}, w'_{i,E}f'_{i,E}$ as

$$\|y'_{i,E}\|_u = -\|y_{i,E}\|_u - \|t'_{i,E}\cdot r_{i,E}\|_u, \quad \|z'_{i,E}\|\|_u = \rho\|r_{i,E}\|_u + \|z_{i,E}\|_u,$$
$$\|w'_{i,E}\|_u = \rho\|a_{i,E}\|_u + \|w_{i,E}\|_u, \qquad \|f'_{i,E}\|_u = \rho\|r'_{i,E}\|_u + \|f_{i,E}\|_u,$$

applying the linearity of the Shamir SSS. Upon the end of this step, all the 3rd round responses constituting $\{\phi_{2,E}\}_{E\in\mathbf{E}_{\text{tally}}}$ can be recovered normally from the k respective shares. Thus, for each commitment in $\mathbf{E}_{\text{tally}}$ there is a triple of pre-audit data, challenge and post-audit data that forms a *complete* Σ *proof* of a valid commitment to some encoded option.

6. Every trustee T_u performs homomorphic tally by computing $E_{\text{sum}} = \prod_{E\in\mathbf{E}_{\text{tally}}} E$ and prepares its share of the opening $\left(\sum_{E^{(a_\ell)}_{\ell,j'_\ell}\in\mathbf{E}_{\text{tally}}}(n+1)^{j_\ell}, \sum_{E^{(a_\ell)}_{\ell,j'_\ell}\in\mathbf{E}_{\text{tally}}} r^{(a_\ell)}_{\ell,j_\ell}\right)$ to E_{sum}, denoted by (T,R), as follows:

(i). It computes $T \stackrel{\Delta}{=} \sum_{E^{(a_\ell)}_{\ell,j'_\ell}\in\mathbf{E}_{\text{tally}}}(n+1)^{j_\ell}$.

(ii). It computes $\|R\|_u \stackrel{\Delta}{=} \sum_{E^{(a_\ell)}_{\ell,j'_\ell}\in\mathbf{E}_{\text{tally}}} \|r^{(a_\ell)}_{\ell,j_\ell}\|_u$.

(iii). It posts $(E_{\text{sum}}, T, \|R\|_u)$ in the BB.

After all $\{(E_{\mathsf{sum}}, T, \|R\|_u)\}_{u\in[k]}$ have been posted, every party can recover (T, R). The additive homomorphic property implies that T is the election result encoded in the number system with base $n+1$ and it is committed under randomness R, which is the sum of all the randomness used for the commitments in $\mathbf{E}_{\mathsf{tally}}$.

7. Next, every trustee T_u provides its share of openings to all the commitments in $\mathbf{E}_{\mathsf{open}}$; that is, for each $\psi_\ell \in \mathsf{V}_{\mathsf{tally}}$, every trustee T_u reads $(\mathsf{tag}_\ell, a_\ell)$ from the BB to recover its share of respective audit ballot part $\|\mathbf{B}_\ell^{(1-a_\ell)}\|_u$ from $\|\mathsf{st}_{\mathsf{ea}}\|_u$. Then, it sends to BB the list $\left\{\|\mathbf{B}_\ell^{(1-a_\ell)}\|_u\right\}_{V_\ell\in\mathcal{V}_{\mathsf{succ}}}$. After all trustees have posted their shares, every party can recover the set of openings to all the commitments in $\mathbf{E}_{\mathsf{open}}$, denoted by Open.

8. After the end of the tally phase, the following election transcript τ can be read from the BB:

$$\mathsf{Pub}, \left\{(C_{\ell,j}^{(a)}, t_{\ell,j}^{(a)})\right\}_{\ell\in[n],j\in[m]}^{a\in\{0,1\}}, \left(\mathbf{E}_{\mathsf{tally}}, \mathbf{C}_{\mathsf{cast}}\right), \left(E_{\mathsf{sum}}, (T,R)\right), (\mathbf{E}_{\mathsf{open}}, \mathsf{Open},), \{\rho_E, \phi_{2,E}\}_{E\in\mathbf{E}_{\mathsf{tally}}}$$

The **Result**(τ) *algorithm* :

The election result R_τ is derived by the following decoding algorithm:

Set $X \leftarrow T$;
- For $j = 1, \ldots, m$:
  1. $x_j \leftarrow X \bmod (n+1)$;
  2. $X \leftarrow (X - x_j)/(n+1)$;
- Return $\langle x_1, \ldots, x_m\rangle$;

The correctness of the algorithm (and the e-voting system) is shown in Theorem 4.3.

The **Verify**(τ, audit) *algorithm* :

Initially, audit is parsed as $\left(\mathsf{tag}, a, C, \mathbf{B}^{(1-a)}\right)$. The algorithm returns 1 only if the following checks are valid:

1. All committed information in τ is associated with n ballots indexed under different tags and no two vote-codes under the same tag are marked as 'voted'.
2. Let $\hat{C}$ be a vote-code that appears in part $\hat{\mathbf{B}}^{(\hat{a})}$ of some ballot and has been marked as 'voted' . Then, only the committed information for the other part $\hat{\mathbf{B}}^{(1-\hat{a})}$ of this ballot has been opened.
3. All the complete Σ proofs that are associated with commitments in $\mathbf{E}_{\mathsf{tally}}$ are valid.

4. $E_{\mathsf{sum}} = \prod_{E \in \mathbf{E}_{\mathsf{tally}}} E$.

5. All the openings of the commitments are valid.

6. tag equals some tag_ℓ in τ for some $\ell \in [n]$ and it holds that $a = a_\ell$.

7. The vote-code that is marked as 'voted' and is associated to tag_ℓ is C where ℓ is as in check 6.

8. The correspondence of option encodings to vote-codes revealed in the opening of the commitments $\{(U_{\ell,j}^{(1-a_\ell)}, E_{\ell,j}^{(1-a_\ell)})\}_{j \in [m]}$ where ℓ is as in check 6, is equal to the one defined in $\mathbf{B}^{(1-a)}$.

4.3.1 Correctness of DEMOS-A

We prove the correctness of DEMOS-A in the following theorem. In the remaining of the chapter, we assume that $q > n \cdot (n+1)^{m-1}$.

Theorem 4.3. *Let q be the size of the group for the ElGamal commitment scheme described in Section 4.2.1. If $q > n \cdot (n+1)^{m-1}$, then DEMOS-A has perfect correctness.*

Proof. It is straightforward that in a honest execution where the information is consistently tabulated. In addition, by the correctness of the building blocks that are used all verifications are successful. Thus, it suffices to show the correctness of the **Result**$(\cdot)$ algorithm.

We denote by opt_{j_ℓ} the option that the voter V_ℓ has selected, i.e. $\mathcal{U}_\ell = \{\mathsf{opt}_{j_\ell}\}$. The encoding of opt_{j_ℓ} is $(n+1)^{j_\ell - 1}$, therefore we have that

$$E_{\mathsf{sum}} = \prod_{\ell \in [n]} \mathsf{CS.Com}_{\mathsf{ck}}\big((n+1)^{j_\ell - 1}; r_{\ell,j_\ell}^{(a_\ell)}\big) = \mathsf{CS.Com}_{\mathsf{ck}}(T; R) \ .$$

Due to the binding and homomorphic properties, the above equation implies that if the options $\mathsf{opt}_1, \ldots, \mathsf{opt}_m$ have been voted $t_1, \ldots, t_m$ times respectively, then E_{sum} is opened to

$$T = \sum_{j=1}^{m} t_j \cdot (n+1)^{j-1} \bmod q \ .$$

We observe that $T \leqslant n \cdot (n+1)^{m-1} < q$ (the equality holds when all n voters vote for option opt_m). Therefore, $T \bmod q = T$, i.e. E_{sum} is opened to the actual result, $\langle t_1, \ldots, t_m \rangle$. Moreover, since $0 \leqslant t_1, \ldots, t_m \leqslant n$, we have that $t_i = t_i \bmod (n+1)$, for all i. By induction, we will show that the output $\langle x_1, \ldots, x_m \rangle$ of the **Result** algorithm equals to $\langle t_1, \ldots, t_m \rangle$, which completes the proof.

- For $j = 1$, we have that

$$x_1 = T \bmod (n+1) = \sum_{i=1}^{m} t_i \cdot (n+1)^{i-1} \bmod (n+1) =$$
$$= t_1 \bmod (n+1) + (n+1) \cdot \sum_{i=2}^{m} t_i \cdot (n+1)^{i-2} \bmod (n+1) = t_1.$$

- For $2 \leqslant j \leqslant m$, if $x_i = t_i$ for every $i < j$, then, by the description of the decoding algorithm

$$x_j = \frac{T - \sum_{1 \leqslant i < j} x_i \cdot (n+1)^{i-1}}{(n+1)^{j-1}} \bmod (n+1) =$$
$$= \frac{\sum_{i=1}^{m} t_i \cdot (n+1)^{i-1} - \sum_{1 \leqslant i < j} t_i \cdot (n+1)^{i-1}}{(n+1)^{j-1}} \bmod (n+1) =$$
$$= \frac{\sum_{j \leqslant i \leqslant m} t_i \cdot (n+1)^{i-1}}{(n+1)^{j-1}} \bmod (n+1) =$$
$$= t_j \bmod (n+1) + \sum_{0 < s \leqslant m-j} t_{j+s} \cdot (n+1)^s \bmod (n+1) = t_j.$$

■

4.4 A Toy Example

For the better understanding of DEMOS-A, we provide a toy example of a referendum where $P_1 =$ YES, $P_2 =$ NO are the candidates and $\mathcal{V}$ consists of three voters V_1, V_2, V_3. Our goal is to familiarise the reader with the functionality of our system so, for simplicity, we deviate from the description in Section 4.3 by not including Σ-protocol proofs.

Setup phase :

EA generates the vote-codes for the double ballots $\mathbf{B}_1$,$\mathbf{B}_2$ and $\mathbf{B}_3$ of V_1, V_2 and V_3 as

$$\Big(C^{(0)}_{1,1} = 27935, C^{(0)}_{1,2} = 75218, C^{(1)}_{1,1} = 84439, C^{(1)}_{1,2} = 77396\Big),$$
$$\Big(C^{(0)}_{2,1} = 58729, C^{(0)}_{2,2} = 45343, C^{(1)}_{2,1} = 14582, C^{(1)}_{2,2} = 93484\Big),$$
$$\Big(C^{(0)}_{3,1} = 52658, C^{(0)}_{3,2} = 65864, C^{(1)}_{3,1} = 84373, C^{(1)}_{3,2} = 49251\Big)$$

respectively. The double ballots $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ are labelled by the tags $101, 102, 103$ respectively and are formed as follows:

101	
Vote-code	**Option**
27935	YES
75218	NO
84439	YES
77396	NO

102	
Vote-code	**Option**
58729	YES
45343	NO
14582	YES
93484	NO

103	
Vote-code	**Option**
52658	YES
65864	NO
84373	YES
49251	NO

101	
Vote-code commitment	**Option encoding commitment**
$\mathsf{CS.Com}(\mathsf{ck}, 27935; t_{1,1}^{(0)})$	$\mathsf{CS.Com}(\mathsf{ck}, 1; r_{1,1}^{(0)})$
$\mathsf{CS.Com}(\mathsf{ck}, 75218; t_{1,2}^{(0)})$	$\mathsf{CS.Com}(\mathsf{ck}, 4; r_{1,2}^{(0)})$
$\mathsf{CS.Com}(\mathsf{ck}, 77396; t_{1,2}^{(1)})$	$\mathsf{CS.Com}(\mathsf{ck}, 4; r_{1,2}^{(1)})$
$\mathsf{CS.Com}(\mathsf{ck}, 84439; t_{1,1}^{(1)})$	$\mathsf{CS.Com}(\mathsf{ck}, 1; r_{1,1}^{(1)})$

102	
Vote-code commitment	**Option encoding commitment**
$\mathsf{CS.Com}(\mathsf{ck}, 45343; t_{2,2}^{(0)})$	$\mathsf{CS.Com}(\mathsf{ck}, 4; r_{2,2}^{(0)})$
$\mathsf{CS.Com}(\mathsf{ck}, 58729; t_{2,1}^{(0)})$	$\mathsf{CS.Com}(\mathsf{ck}, 1; r_{2,1}^{(0)})$
$\mathsf{CS.Com}(\mathsf{ck}, 14582; t_{2,1}^{(1)})$	$\mathsf{CS.Com}(\mathsf{ck}, 1; r_{2,1}^{(1)})$
$\mathsf{CS.Com}(\mathsf{ck}, 93484; t_{2,2}^{(1)})$	$\mathsf{CS.Com}(\mathsf{ck}, 1; r_{2,2}^{(1)})$

103	
Vote-code commitment	**Option encoding commitment**
$\mathsf{CS.Com}(\mathsf{ck}, 52658; t_{3,1}^{(0)})$	$\mathsf{CS.Com}(\mathsf{ck}, 1; r_{3,1}^{(0)})$
$\mathsf{CS.Com}(\mathsf{ck}, 65864; t_{3,2}^{(0)})$	$\mathsf{CS.Com}(\mathsf{ck}, 4; r_{3,2}^{(0)})$
$\mathsf{CS.Com}(\mathsf{ck}, 49251; t_{3,2}^{(1)})$	$\mathsf{CS.Com}(\mathsf{ck}, 4; r_{3,2}^{(1)})$
$\mathsf{CS.Com}(\mathsf{ck}, 84373; t_{3,1}^{(1)})$	$\mathsf{CS.Com}(\mathsf{ck}, 1; r_{3,1}^{(1)})$

Figure 4.2: Ballot tabulation in the BB at setup phase.

EA prepares the commitments to each vote-code and the encoding of the candidate that they correspond. The commitment for YES (resp. NO) is a commitment to $(3+1)^0 = 1$ (resp. $(3+1)^1 = 4$). Next, it chooses whether the commitments of the vote-code and candidate pairs are going to be ordered in the BB as they are in the ballot part, or swapped. For example, assume that for the ballot s_1 (resp. s_2) (resp. s_3), EA chooses to leave the order in ballot part (0) (resp. (1)) (resp. (0)) intact and to swap the pairs in ballot part (1) (resp. (0)) (resp. (1)) . Then, the information posted in the BB for s_1 would have the following form in Figure 4.2.

Voting phase :

Suppose that V_1 votes for NO using ballot part (1), V_2 votes for YES using ballot part (1) and V_3 votes for YES using ballot part (0).Then, the votes cast by V_1, V_2 and V_3 are $(101, 1, 77396)$, $(102, 1, 14582)$ and $(103, 0, 52568)$ respectively.

The coins that V_1, V_2 and V_3 have flipped, are $a_1 = 1, a_2 = 1$ and $a_3 = 0$ respectively. Hence, we get internal randomness, $(1, 1, 0)$, of 3 bits (which would be the "weak source" of randomness used for the extraction of the challenge of the Σ protocols).

The individual audit information that the voters receive are

(101,1,77396)	
27935	YES
75218	NO

$(102, 1, 14582)$	
58729	YES
45343	NO

$(103, 0, 52568)$	
84373	YES
49251	NO

Tally phase :

After the voting ends, and the trustees provide their vote-code decommitment shares, VC opens the vote-code commitments, marks the cast vote-codes $77396, 14582$ and 52658 and includes the respective option encoding commitments CS.Com$(\mathsf{ck}, 4; r^{(1)}_{1,2})$, CS.Com$(\mathsf{ck}, 1; r^{(1)}_{2,1})$ and CS.Com$(\mathsf{ck}, 1; r^{(0)}_{3,1})$ in the tally set.

Next, the trustees collectively perform homomorphic tally, by computing the product of the above encoded candidate commitments as

$$E_{\mathsf{sum}} = \mathsf{CS.Com}(\mathsf{ck}, 4; r^{(1)}_{1,2}) \cdot \mathsf{CS.Com}(\mathsf{ck}, 1; r^{(1)}_{2,1}) \cdot \mathsf{CS.Com}(\mathsf{ck}, 1; r^{(0)}_{3,1}) = \\ = \mathsf{CS.Com}(\mathsf{ck}, 6; r^{(1)}_{1,2} + r^{(1)}_{2,1} + r^{(0)}_{3,1}) \ .$$

At the end of the tally phase, E_{sum} along with its opening at value $(6; r^{(1)}_{1,2} + r^{(1)}_{2,1} + r^{(0)}_{3,1})$, are posted in the BB. The result is derived by computing $x_1 = 6 \bmod 4 = 2$ and $x_2 = ((6 - x_1)/4) \bmod 4 = 1$, which is interpreted as two votes for YES and one for NO.

Verification :

In the verification phase, the EA opens the commitments in the ballot parts that the voters selected for auditing. For example, V_1 would check the consistency of her individual audit information with the election audit data in the BB, as illustrated in Figure 4.3.

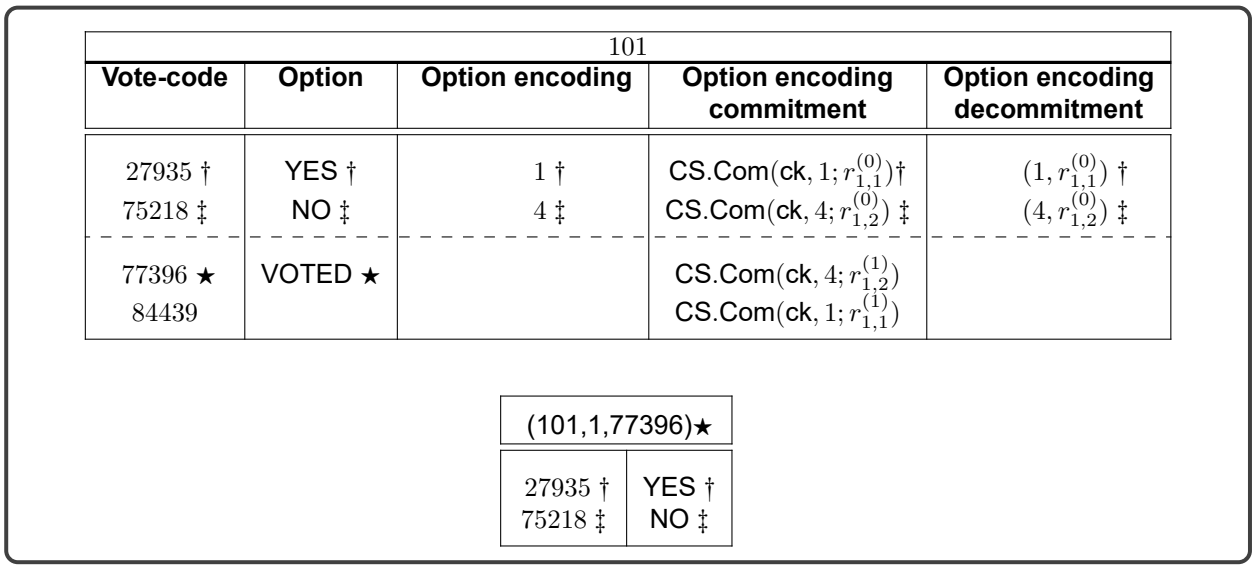


Figure 4.3: Ballot tabulation in the BB and verification via individual audit information after election end. The symbols $\star, \dagger, \ddagger$ indicate the data grouping w.r.t. auditing.

Observe that, as we will prove shortly, the cut-and-choose verification that V_1 performs, does not reveal her vote even to a party that obtains her individual audit information. This is because the cast vote-code (77396) alone does not leak any information about the associated candidate (NO), while the entirely opened auditing part only serves as a check that the correspondence of the vote-codes and candidates in this part has not been tampered with. Therefore, V_1 can delegate the task of verification to a third party, without compromising her privacy.

4.5 End-to-end Verifiability of DEMOS-A

In this section, we prove the E2E verifiability tht DEMOS-A achieves under the security model in Section 3.3. We stress that our E2E verifiability theorem holds information theoretically in the standard model. Before stating the theorem (cf. Subsection 4.5.2), we list the plausible attacks against the verifiability of DEMOS-A, describing their effectiveness and detection probability at a high level (cf. Subsection 4.5.1).

4.5.1 Attacks on verifiability

For simplicity, we exclude all the trivial attacks that the adversary may follow, i.e. the ones that will be detected with certainty (e.g. malformed or unreadable election transcript). Therefore, the meaningful types of attack that an adversary may launch against DEMOS-A are the following:

- *Invalid encoding attack*: the adversary creates an option-encoding commitment to some invalid value, i.e. a vector that does not encode a candidate selection (e.g., multiple votes for some specific candidate). This attack can be prevented by the soundness of the Σ protocol in Subsection 4.2.4, except from the negligible soundness error ϵ. The proof verification is done via a trusted *auditing supporting device (ASD)*.

- *Modification attack*: the information in an honest voter's ballot part is inconsistent with the respective audit data committed in the BB as compared with the one (e.g. the vote-code and candidate correspondence is altered). The deviation achieved by this type of attack is at most 1, whereas the probability of detection is 1/2 (the voter chooses to audit using the inconsistent ballot part).

- *Clash attack* [73]: the adversary instructs y honest voters whose ballots are indexed under the same tag to vote so that the votes of any $y-1$ out of these y voters are all different than some fixed $y-1$ committed votes (either cast by corrupted voters or initially injected in τ by the adversary). All y voters verify the correct counting of their votes by auditing the same information on the BB and hence miss the injected votes that produce the tally deviation. The deviation achieved by this type of attack is $y-1$, whereas the probability of detection is $1-2^{y-1}$ (at least two out of the y voters choose a different ballot part to vote).

Remark 4.1 (***Completeness of the attack list***)**.** It can be easily shown that the above list exhausts all possible attack strategies against DEMOS-A in our threat model. In the case where all ballot information is tabulated using all valid commitments in the BB without being deleted or replaced, the adversary can only perform a combination of modification and clash attacks on the honest votes. If no such combination occurs, then all honestly cast votes are in correct (yet unknown) one-to-one correspondence with the BB audit data, hence by the perfect binding property of the commitment scheme, the opening of the homomorphic tally matches the intended result.

4.5.2 End-to-end verifiability theorem

Having described all possible attack scenarios, we prove the E2E verifiability of DEMOS-A in the following theorem.

Theorem 4.4. *Assume an election run of DEMOS-A with n voters, m candidates and k trustees. Let q be the size of the group for the ElGamal commitment scheme described in Section 4.2.1. Then, DEMOS-A achieves E2E verifiability information theoretically for at least θ honest successful voters and tally deviation δ with error*

$$2^{-\delta} + \epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta)) \ ,$$

where $\epsilon(\cdot,\cdot,\cdot,\cdot)$ is the soundness error of the Σ protocol given in Theorem 4.2.

Proof. W.l.o.g., we assume that in any adversarial execution as described in the E2E verifiability game $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$, exactly n ballots are tabulated on τ under n different tags and all vote-codes are marked as 'voted' correspond to different tags (if such deviations happen the transcript is immediately rejected). In the same spirit, we assume there is no double ballot that both parts have been opened and that all double ballots for honest voters in $\mathcal{V}_{\text{succ}}$ are well-formed, otherwise they would not engage in the **Cast** protocol. Finally, we recall that the adversary cannot modify the history of the transcript since it does not have control over the BB.

As a first step, we construct a vote extractor $\mathcal{E}$ for DEMOS-A. Then, we will prove that every adversary can win the E2E verifiability game w.r.t. $\mathcal{E}$ with probability bounded by the error in the theorem's statement.

Construction of the vote extractor for DEMOS-A :

$\mathcal{E}$ on input τ and the set of individual audit information $\{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}$, where $\mathcal{V}_{\text{succ}}$ is the set of the honest voters that voted successfully, operates as follows:

The vote extractor $\mathcal{E}\left(\tau, \{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}\right)$ for DEMOS-A

1. Let $t \leqslant |\mathcal{V}_{\text{succ}}|$ be the number of different tags that appear in $\{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}$. This implies that the ballot audit for all voters in $\mathcal{V}_{\text{succ}}$ focuses on a list of t tabulated ballots on the BB (thus, an adversary may inject $|\mathcal{V}_{\text{succ}}| - t$ ballots for candidate selections of its choice that will be counted in the final tally as if they were honest).
2. If $\textbf{Result}(\tau) = \perp$ (i.e., the transcript is not meaningful), then $\mathcal{E}$ outputs $\perp$. Otherwise, $\mathcal{E}$ (arbitrarily) arranges the voters in $\mathcal{V} \backslash \mathcal{V}_{\text{succ}}$ and the tags not included in $\{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}$ as $\langle V_\ell^{\mathcal{E}} \rangle_{\ell \in [n - |\mathcal{V}_{\text{succ}}|]}$ and $\langle \text{tag}_\ell^{\mathcal{E}} \rangle_{\ell \in [n-t]}$ respectively. Next, for every $\ell \in [n - |\mathcal{V}_{\text{succ}}|]$:
   (a) If there is no marked as 'voted' vote-code that is associated with $\text{tag}_\ell^{\mathcal{E}}$, then $\mathcal{E}$ sets $\mathcal{U}_\ell^{\mathcal{E}} = \varnothing$ (encoded as the zero vector) which is interpreted as an abort for voter $V_\ell^{\mathcal{E}}$.
   (b) If there is a 'voted' vote-code $C_{\ell,j}^{(a)}$ that is associated with $\text{tag}_\ell^{\mathcal{E}}$, then $\mathcal{E}$ brute-force opens the respective encoded candidate commitment $E_{\ell,j}^{(a)}$ to a value Open_ℓ (recall the commitment is perfectly binding). If Open_ℓ is a valid encoding (i.e. $\text{Open}_\ell \in \{(n+1)^0, (n+1)^1, \ldots, (n+1)^{m-1}\}$) of a candidate $\mathcal{P}_\ell^{\mathcal{E}}$, then $\mathcal{E}$ sets $\mathcal{U}_\ell^{\mathcal{E}} = \{\mathcal{P}_\ell^{\mathcal{E}}\}$. Otherwise, it outputs $\perp$.
3. Finally, $\mathcal{E}$ outputs $\langle \mathcal{U}_\ell^{\mathcal{E}} \rangle_{V_\ell^{\mathcal{E}} \in \mathcal{V} \backslash \mathcal{V}_{\text{succ}}}$. Note that if $t < |\mathcal{V}_{\text{succ}}|$, then the remaining tags $\text{tag}_{n-|\mathcal{V}_{\text{succ}}|+1}^{\mathcal{E}}, \ldots, \text{tag}_{n-t}^{\mathcal{E}}$ are ignored by $\mathcal{E}$.

Based on the above vote extractor, we will prove the E2E verifiability of our scheme. Assume an adversary $\mathcal{A}$ that wins the game $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$ described in Figure 3.1. Namely, $\mathcal{A}$ breaks E2E verifiability by allowing at least θ honest successful voters and achieving tally deviation d. Since there is at least one honest voter that performs verification ($\theta > 0$), w.l.o.g. we assume that $\mathcal{A}$ always outputs meaningful transcripts.

Let F be the event that $\mathcal{A}$ performs at least one invalid encoding attack (cf. Subsection 4.5.1). Namely, there exists a committed value in τ which is marked to be counted and invalid (i.e., it is in $\mathbf{E}_{\text{tally}}$ but it is not a commitment to some candidate encoding). Since condition (1) of $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$ holds, we have that there are at least θ honest voters. Therefore, by applying Theorem 4.2 for min entropy equal to θ, we have that each Σ protocol has soundness error $\epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta)$. Hence, the probability that a committed value is invalid while verification accepts is no more than $\epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta)$. Since there is at least one honest voter that verifies, we conclude that

$$\Pr\left[G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k) = 1 \mid F\right] \leqslant \epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta). \tag{4.1}$$

Assume that F does not occur. Thus, all marked committed values in $\mathbf{E}_{\text{tally}}$ correspond to a valid candidate encoding. This implies that (i) the maximum deviation per marked commitment that $\mathcal{A}$ may achieve is 1 (the vote is counted for a candidate other than the intended one) and (ii) $\mathcal{E}$ does not output $\perp$ (it returns a vector $\langle \mathcal{U}_\ell^{\mathcal{E}} \rangle_{V_\ell^{\mathcal{E}} \in \mathcal{V} \setminus \mathcal{V}_{\text{succ}}}$), so $\mathcal{A}$ wins because conditions (1),(2) and (3-a) hold. The auditor can verify that E_{sum} is equal to the homomorphic commitment $\prod_{E \in \mathbf{E}_{\text{tally}}} E$. Due to the perfect binding of the commitment scheme, the tally $f(\langle \mathcal{U}_\ell^{\mathcal{E}} \rangle_{V_\ell^{\mathcal{E}} \in \mathcal{V} \setminus \mathcal{V}_{\text{succ}}})$ that $\mathcal{E}$ estimates as non-honest votes, is correctly included in the adversarial result that derives from the opening (T, R) of E_{sum}. Thus, the deviation from the intended result that $\mathcal{A}$ achieves, derives only by miscounting the honest votes. This may be achieved if $\mathcal{A}$ performs combination of successful modification and clash attacks (cf. Subsection 4.5.1.

Let $\mathcal{V}_{\text{succ}}^1, \ldots, \mathcal{V}_{\text{succ}}^d$ be the partition of $\mathcal{V}_{\text{succ}}$ s.t. each of these subsets consists of honest voters that their individual audit information (hence their ballots) are indexed under the same tag. These subsets are created adaptively, according to the strategy of $\mathcal{A}$, under the constraint that $|\mathcal{V}_{\text{succ}}| \geqslant \theta$. Note that there are $|\mathcal{V}_{\text{succ}}| - d$ ignored tags in vote extraction, while $\sum_{i \in [d]} (|\mathcal{V}_{\text{succ}}^i| - 1) = |\mathcal{V}_{\text{succ}}| - d$. This implies that the adversary can perform clash attacks in all these subsets, with maximum possible deviation. We will prove that given that F does not occur, the success probability of $\mathcal{A}$ is no more than $2^{-\delta}$, whatever its strategy might be.

We observe that in order for $\mathcal{A}$ to win, all voters in $\mathcal{V}_{\text{succ}}^i$ must have the same receipt, or else inconsistencies will cause verification to fail. To achieve this, $\mathcal{A}$ must instruct the voters from the same subset to vote so that they all cast the same vote-code (otherwise two marked vote-codes under the same tag should appear) and create the corresponding audit ballot part identically for each auditing voter. In detail, in order for $\mathcal{A}$ to win, the following must hold for each $\mathcal{V}_{\text{succ}}^i$, $i \in [d]$:

1. There is a representative vote-code C_i that appears in the same ballot part of all the double ballots of the voters in $\mathcal{V}_{\text{succ}}^i$. The voters must select this part to vote by casting C_i. Therefore, the coin-flippings of the auditing voters must be consistent, in the sense that they correspond to ballot parts that contain a consistent vote-code. There can be at most 2 consistent coin-flips (i.e., either all coins are flipped to 0 or all coins are flipped to 1). Thus, the probability of consistent coin-flipping in $\mathcal{V}_{\text{succ}}^i$

is at most $2/2^{|\mathcal{V}^i_{\text{succ}}|} = 2^{-(|\mathcal{V}^i_{\text{succ}}|-1)}$. In addition, the ballot parts that will be used for auditing must contain the same information, up to a permutation of the vote-code and candidate pairs.

2. If $\mathcal{A}$ wants to achieve $|\mathcal{V}^i_{\text{succ}}|$ deviation exploiting the voters in $\mathcal{V}^i_{\text{succ}}$, then it must perform a modification attack in at least one voter V in $\mathcal{V}^i_{\text{succ}}$. This is because if all voters' ballots are consistent to the corresponding committed information in τ, then by performing only a clash attack in $\mathcal{V}^i_{\text{succ}}$, $\mathcal{A}$ can achieve deviation by at most $|\mathcal{V}^i_{\text{succ}}| - 1$, as described above. However, the modification comes with a loss of 1/2 success probability, since $\mathcal{A}$ must also guess which is the part that V is going to use for voting. Indeed, if V chooses to audit the modified part of the ballot, then she will detect the attack. Therefore, all voters in $\mathcal{V}^i_{\text{succ}}$ must perform a consistent coin-flip that agrees with the coin-flip of V. It is straightforward that in case of a single modification attack this event happens with $1/2 \cdot (1/2)^{|\mathcal{V}^i_{\text{succ}}|-1} = (1/2)^{|\mathcal{V}^i_{\text{succ}}|}$ probability. Moreover, in case $\mathcal{V}^i_{\text{succ}} \geqslant 2$, performing two modification attacks does not lead to any improvement in terms of probability or maximum deviation.

We note that the above arguments hold trivially, if $\mathcal{V}^i_{\text{succ}}$ is a singleton. Let **X** be the set of subsets from $\{\mathcal{V}^1_{\text{succ}}, \ldots, \mathcal{V}^d_{\text{succ}}\}$ that $\mathcal{A}$ performs clash attacks and **Y** the collection that $\mathcal{A}$ performs a modification attack on at least one voter in each of the subsets. According to the previous arguments, we have the following cases:

(i). for each $\mathcal{V}^i_{\text{succ}} \in \mathbf{X} \backslash \mathbf{Y}$ the maximum deviation is $|\mathcal{V}^i_{\text{succ}}| - 1$,

(ii). for each $\mathcal{V}^i_{\text{succ}} \in \mathbf{Y} \backslash \mathbf{X}$ the maximum deviation is 1,

(iii). for each $\mathcal{V}^i_{\text{succ}} \in \mathbf{X} \cap \mathbf{Y}$ the maximum deviation is $|\mathcal{V}^i_{\text{succ}}|$, and

(iv). for each $\mathcal{V}^i_{\text{succ}} \in \{\mathcal{V}^1_{\text{succ}}, \ldots, \mathcal{V}^d_{\text{succ}}\} \backslash (\mathbf{X} \cup \mathbf{Y})$ the maximum deviation is 0.

For brevity, let $x = |\mathbf{X}|$ and $y = |\mathbf{Y}|$. By the above, the tally deviation from the intended result that $\mathcal{A}$ achieves is at most

$$\sum_{\mathcal{V}^i_{\text{succ}} \in \mathbf{X} \backslash \mathbf{Y}} (|\mathcal{V}^i_{\text{succ}}| - 1) + \sum_{\mathcal{V}^i_{\text{succ}} \in \mathbf{Y} \backslash \mathbf{X}} 1 + \sum_{\mathcal{V}^i_{\text{succ}} \in \mathbf{X} \cap \mathbf{Y}} |\mathcal{V}^i_{\text{succ}}| = \sum_{\mathcal{V}^i_{\text{succ}} \in \mathbf{X}} |\mathcal{V}^i_{\text{succ}}| - x + y \leqslant |\mathcal{V}_{\text{succ}}| - x + y.$$

In order for $\mathcal{A}$ to win, condition (3-a) must hold, so $|\mathcal{V}_{\text{succ}}| - x + y \geqslant \delta$.

We will now upper bound the success probability of $\mathcal{A}$. Since $\{\mathcal{V}^1_{\text{succ}}, \ldots, \mathcal{V}^d_{\text{succ}}\}$ is a partition of $\mathcal{V}_{\text{succ}}$, we have that $\mathcal{A}$ must not be detected by all the voters in all these subsets. Thus,

$$\begin{aligned} \Pr[G^{\mathcal{A},\mathcal{E},\delta,\theta}_{\text{E2E}}(1^\lambda, m, n, k) = 1 | \neg F] &\leqslant \prod_{\mathcal{V}^i_{\text{succ}} \in \mathbf{Y}} 2^{-|\mathcal{V}^i_{\text{succ}}|} \cdot \prod_{\mathcal{V}^i_{\text{succ}} \in \{\mathcal{V}^1_{\text{succ}}, \ldots, \mathcal{V}^d_{\text{succ}}\} \backslash \mathbf{Y}} 2^{-(|\mathcal{V}^i_{\text{succ}}|-1)} = \\ &= 2^{-\sum_{\mathcal{V}^i_{\text{succ}} \in \mathbf{Y}} |\mathcal{V}^i_{\text{succ}}| - \sum_{\mathcal{V}^i_{\text{succ}} \in \{\mathcal{V}^1_{\text{succ}}, \ldots, \mathcal{V}^d_{\text{succ}}\} \backslash \mathbf{Y}} (|\mathcal{V}^i_{\text{succ}}|-1)} = \\ &= 2^{-\left(|\mathcal{V}_{\text{succ}}| - (d-y)\right)} \leqslant 2^{-(|\mathcal{V}_{\text{succ}}| - x + y)} \leqslant 2^{-\delta}, \end{aligned} \tag{4.2}$$

where we used the fact that $x \leqslant d$. By adding (4.1),(4.2) we conclude that

$$\mathsf{Pr}[G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k) = 1] \leqslant 2^{-\delta} + \epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta) \ .$$

■

Remark 4.2 (*Strenth of Theorem 4.4*)**.** Note that if the number of honest voters satisfies the bound $\theta = \Omega(n \log\log m/\log q + \lambda)$, then the overall soundness error of the repeated Σ protocol will be sufficiently small. For instance, in an election where there are $n = 1000$ voters and $m = 40$ candidates we can use a group with at least 500 bit prime order q. Assuming a number of $\theta = 50$ honest voters (5% of total) we can divide the 1000 voter's coins into two challenges with 500 bits each (i.e. $k = 2$). With these parameters the above theorem will have a verifiability error that is at most $2^{-43} + (1/2)^\delta$ where δ is the tally deviation. We remark that in this setting no deterministic extractor would be able to provide sufficient entropy and hence our ZK amplification technique is crucial.

4.6 Game-based Voter Privacy/PCR of DEMOS-A

In order to show that DEMOS-A satisfies privacy according to the game-based Definition 3.3, we utilise complexity leveraging. Specifically, the system security parameter is configured such that breaking the hiding property of the underlying commitment scheme is much harder than guessing the challenge of the Σ protocol; therefore, we can simulate the protocol's view by guessing the proof challenges without breaking the hiding property. Due to this proof technique, the number of corrupted voters t should be polynomially related to the security parameter λ in a certain way; while the total number of voters n can be any function that is poly(λ) (as long as the correctness requirement is fulfilled, cf. theorem 4.3.1). We emphasise that given a specific n, our system can support privacy for any desired number of adversarial voters $t < n$ (as long as a suitably large security parameter λ is used).

Theorem 4.5. *Assume an election run of DEMOS-A with n voters, m candidates and k trustees. Assume there exists a constant $c, 0 < c < 1$ such that for any 2^{λ^c}-time adversary $\mathcal{A}$, the advantage of breaking the hiding property of the commitment scheme is* $\mathsf{Adv}_{\mathsf{hide}}(\mathcal{A}) = \mathsf{negl}(\lambda)$. *Let $t = \lambda^{c'}$ for any constant $c' < c$. Then, for any constant m and n, k polynomial in the security parameter λ, DEMOS-A achieves voter privacy/PCR for at most t corrupted voters.*

Proof. By the information theoretic security of the (k, k)-linear SSS and the non-interaction between the trustees during the tally phase, an adversary that corrupts $k-1$ trustees does not learn anything regarding the values that are in committed form (vote-codes or encoded options) unless these are opened after election end. As a result, for the rest of the proof, we restrict to adversaries that corrupt only theVC, i.e. the election setup and tally phases are executed by the challenger Ch in the voter privacy/PCR game $G_{t\text{-}\mathsf{priv}}^{\mathcal{A},\mathcal{S}}(1^\lambda, n, m, k)$.

To prove our claim, we will explicitly construct a view simulator $\mathcal{S}$ such that we can convert any adversary $\mathcal{A}$ who can win the game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m, k)$ a non-negligible probability to an adversary $\mathcal{B}$ who can break the commitment hiding assumption within poly$(\lambda) \cdot 2^t = o(2^{\lambda^c})$ time.

The construction of view simulator $\mathcal{S}$:

Recall that in the privacy game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m, k)$ the challenger Ch is maintaining a coin $b \in \{0,1\}$ and always uses the candidate selection $\mathcal{U}^b_\ell$ in the **Cast** protocol. Note when $n - t < 2$ (i.e. the number of honest voters is strictly less than 2), the simulator $\mathcal{S}$ simply outputs the view of the real **Cast** protocol. It is easy to see that, by definition, the adversary $\mathcal{A}$ loses the voter privacy game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$ unconditionally. When $n - t \geqslant 2$, consider the following simulator $\mathcal{S}$. At the beginning of the experiment, $\mathcal{S}$ flips a coin $b' \in \{0,1\}$. For each honest voter V_ℓ, $\mathcal{S}$ receives $view_\ell = (\mathsf{Pub}, s_\ell, \mathcal{U}^b_\ell, \alpha_\ell)$ and the candidate selections $\langle \mathcal{U}^0_\ell, \mathcal{U}^1_\ell \rangle$. If $\mathcal{U}^b_\ell = \mathcal{U}^{b'}_\ell$, $\mathcal{S}$ outputs the simulated view $\mathsf{view}'_\ell = \mathsf{view}_\ell$. If $\mathcal{U}^b_\ell \neq \mathcal{U}^{b'}_\ell$, $\mathcal{S}$ produces a fake s'_ℓ by switching the vote-codes for candidate selections $\mathcal{U}^b_\ell$ and $\mathcal{U}^{b'}_\ell$, i.e. replacing $\left(C^{(a_\ell)}_{\ell,j_1}, \mathcal{U}^b_\ell\right)$ and $\left(C^{(a_\ell)}_{\ell,j_2}, \mathcal{U}^{b'}_\ell\right)$ with $\left(C^{(a_\ell)}_{\ell,j_2}, \mathcal{U}^b_\ell\right)$ and $\left(C^{(a_\ell)}_{\ell,j_1}, \mathcal{U}^{b'}_\ell\right)$. $\mathcal{S}$ then outputs $\mathsf{view}'_\ell = (\mathsf{Pub}, s'_\ell, \mathcal{U}^{b'}_\ell, \alpha_\ell)$.

Define $\mathsf{Adv}_{G_i,G_j}(\mathcal{A}) := \frac{1}{2}\,|\mathsf{Pr}[\mathcal{A} = 1|G_i] - \mathsf{Pr}[\mathcal{A} = 1|G_j]|$. Consider the following sequence of games from G_0 to G_5.

Game G_0: The actual game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$, where the challenger uses $\mathcal{U}^b_\ell$ in the **Cast** protocol and the above simulator $\mathcal{S}$ is invoked when $b = 1$. By definition,

$$\mathsf{Adv}_{G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda,n,m),G_0}(\mathcal{A}) = 0\ . \tag{4.3}$$

Game G_1: Game G_1 is the same as Game G_0 except the following. At the beginning of the experiment, the challenger Ch generates a set of coins $\{a_\ell\}^n_{\ell=1}$ uniformly at random. During the experiment, for each voter $V_\ell \in \mathcal{V}$, the adversary $\mathcal{A}$ first chooses whether V_ℓ is corrupted. If V_ℓ is not corrupted, Ch uses a_ℓ in the **Cast** protocol to vote on behave of V_ℓ according to $\mathcal{U}^b_\ell$; otherwise, Ch sends s_ℓ to the adversary $\mathcal{A}$ and interacts with $\mathcal{A}$ in the **Cast** protocol, playing the role of EA and BB. Let $\hat{a}_\ell$ be the coin used by the corrupted voter $\hat{V}_\ell \in \mathcal{V}_{\mathsf{corr}}$ in the **Cast** protocol execution. The experiment aborts and starts over if there exists one corrupted voter's coin $\hat{a}_\ell \neq a_\ell$.

It is easy to see that, no matter how $\mathcal{V}_{\mathsf{corr}}$ is chosen, it requires (expected) at most 2^t attempts to guess all the $\hat{a}_\ell$ correctly given $|\mathcal{V}_{\mathsf{corr}}| \leqslant t$. On the other hand, when Ch does not abort, the view of Game G_1 is identical to that of Game G_0. Hence,

$$\mathsf{Adv}_{G_0,G_1}(\mathcal{A}) = 0\ . \tag{4.4}$$

Game G_2: Game G_2 is the same as Game G_1 except the following. The challenger Ch computes a set of commitments $\{E_j\}_{j\in[m]}$, where $E_j = \mathsf{Com}_{\mathsf{ck}}((n+1)^{j-1}; r_j)$ with fresh randomizer $r_j \in \mathbb{Z}_q$. For each ballot, for $a_\ell \in \{0,1\}$, Ch permutes and re-randomizes

$\{E_j\}_{j\in[m]}$ to produce the commitments $\left\{E_{\ell,j}^{(a_\ell)}\right\}_{j\in[m]}$ instead of committing them from scratch as follows.

- Pick a random permutation $\pi_\ell^{(a_\ell)}$ over $[m]$.
- For $j \in [1, m]$,
  - Pick a random $r_{\ell,j}^{(a_\ell)} \leftarrow \mathbb{Z}_q$;
  - Set $E_{\ell,j}^{(a_\ell)} = E_{\pi_\ell^{(a_\ell)}(j)} \cdot \mathsf{Com}_{\mathsf{ck}}(0; r_{\ell,j}^{(a_\ell)})$;

It is straightforward that the view of Game G_2 is identical to that of Game G_1, as the distributions of all the commitments are the same. Hence,

$$\mathsf{Adv}_{G_1,G_2}(\mathcal{A}) = 0\ . \tag{4.5}$$

Game G_3: Game G_3 is the same as Game G_2 except the following. Ch randomly selects $\ell^* \in [n]$ and guesses the tally vector $(t_1, \ldots, t_m)$. Denote $T = \sum_{i=1}^{m} t_i \cdot (n+1)^{i-1}$. Ch aborts if either of the following two events occur: (i) $\mathcal{A}$ corrupts $V_{\ell*}$ and then does not let $V_{\ell*}$ submit a vote; (ii) the guessed T is wrong. When Ch does not abort, it generates the challenge(s) ρ using the guessed voters' coins $\{a_\ell\}_{\ell\in[n]}$ in Game G_1, and then replaces all the real Σ protocols with their simulated transcripts.

The probability $\mathcal{A}$ corrupts $V_{\ell*}$ and then does not let $V_{\ell*}$ submit a vote is at most $\frac{t}{n}$ (Namely all the corrupted voters abort). Besides, the probability the Ch guesses T correctly is at least $\frac{1}{n(n+1)^{m-1}}$. Hence, it requires (expected) at most

$$\frac{n}{n-t} \cdot \left(n(n+1)^{m-1}\right) = \frac{n^2(n+1)^{m-1}}{n-t}$$

attempts to get both events occur. On the other hand, when Ch does not abort, according to Lemma 4.1, for each Σ protocol, the adversary can distinguish the simulated transcript from a real one with advantage at most $\log m \cdot \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$, where $\mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$ is the distinguishing advantage of $\mathcal{A}$ for breaking the hiding property of the ElGamal commitment scheme. There are $2nm$ simulated Σ protocols, so by union bound we have

$$\mathsf{Adv}_{G_2,G_3}(\mathcal{A}) \leqslant 2nm \log m \cdot \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})\ . \tag{4.6}$$

Game G_4: Game G_4 is the same as Game G_3 except the following. At the beginning of the experiment, Ch replaces the set of commitments $\{E_j\}_{j=0}^{m-1}$ in Game G_3 with commitments of 0. Let T and ℓ^* be the ones guessed in Game G_3. For all $\ell \in [n] \wedge \ell \neq \ell^*$, Ch produces $\left\{E_{\ell,j}^{(a_\ell)}\right\}_{j\in[m]}$ by re-randomizing $\{E_j\}_{j\in[m]}$ as in Game G_3; Ch replaces all the commitments $\left\{E_{\ell^*,j}^{(a_{\ell^*})}\right\}_{j\in[m]}$ with fresh commitments of T, i.e. $\mathsf{Com}_{\mathsf{ck}}(T; R_j)$, where $R_j \leftarrow \mathbb{Z}_q$ are chosen uniformly at random.

The reduction to the commitment hiding property :

We now show that the view of Game G_4 is indistinguishable to that of Game G_3 by reduction to the hiding property of the ElGamal commitment scheme. Suppose the adversary $\mathcal{B}$ is playing the hiding game of the underlying commitment scheme. On receiving ck, $\mathcal{B}$ queries two messages $m_0 = 0$ and $m_1 = 1$. Given $E = \mathsf{Com}_{\mathsf{ck}}(m_b; *)$, $\mathcal{B}$ needs to guess b. $\mathcal{B}$ plays as role of the challenger in the game where $\mathcal{A}$ is trying to distinguish between G_3 and G_4. (Assume that $\mathcal{A}$ outputs 1 if she thinks she is in G_3 and outputs 0 if she thinks she is in G_4.) For $j \in [m]$, $\mathcal{B}$ sets $E_j = E^{(n+1)^{j-1}}$. For $\ell \in [n] \wedge \ell \neq \ell^*$, $\mathcal{B}$ produces $\left\{E_{\ell,j}^{(a_\ell)}\right\}_{j\in[m]}$ by re-randomizing $\{E_j\}_{j\in[m]}$ as in Game G_3. For $j \in [m]$, Ch picks $R_j \leftarrow \mathbb{Z}_q$ at random and sets

$$E_{\ell^*,j}^{(a_{\ell^*})} = \mathsf{Com}_{\mathsf{ck}}(T, R_j) / \left(E_j^{t_j - 1} \cdot \prod_{i=1, i\neq j}^{m} E_i^{t_i}\right) .$$

Denote as abort the event that either: (i) $\mathcal{A}$ corrupts V_{ℓ^*} and then does not let V_{ℓ^*} submit a vote; (ii) the guessed T is wrong. Clearly, if E commits to 1, then the produced view is identical to G_3; if E commits to 0, then the produced view is identical to G_4. Except in Game G_4, there are commitments of T in Pub, so the adversary $\mathcal{A}$ might be able to intentionally make the abort abort event occurs with a higher probability. To address this, Ch maintains a counter, and Ch increases the counter by 1 each time abort occurs. Denote halt as the event where abort continuously occurs $\frac{n^3(n+1)^{m-1}}{n-t}$ times, and when halt occurs, Ch outputs 0 in the commitment hiding game; otherwise, $\mathcal{B}$ forwards the bit that $\mathcal{A}$ outputs.

The probability $\mathcal{B}$ wins the hiding game is

$$\begin{aligned}
\mathsf{Pr}[\mathcal{B} = b] &= \mathsf{Pr}[\mathcal{B} = b|\mathsf{halt}] \cdot \mathsf{Pr}[\mathsf{halt}] + \mathsf{Pr}[\mathcal{B} = b|\neg\mathsf{halt}] \cdot \mathsf{Pr}[\neg\mathsf{halt}] \\
&= (1 - \mathsf{Pr}[b = 1|\mathsf{halt}]) \cdot \mathsf{Pr}[\mathsf{halt}] + \frac{1}{2}\mathsf{Pr}[\mathcal{A} = 1|G_3] \cdot \mathsf{Pr}[\neg\mathsf{halt}] \\
&\quad + \frac{1}{2}\mathsf{Pr}[\mathcal{A} = 0|G_4] \cdot \mathsf{Pr}[\neg\mathsf{halt}] \\
&\geqslant (1 - \frac{n-t}{n^2(n+1)^{m-1}})^{\frac{n^3(n+1)^{m-1}}{n-t}} \cdot \mathsf{Pr}[\mathsf{halt}] \\
&\quad + (\frac{1}{2} + \frac{1}{2}\mathsf{Pr}[\mathcal{A} = 1|G_3] - \frac{1}{2}\mathsf{Pr}[\mathcal{A} = 1|G_4]) \cdot \mathsf{Pr}[\neg\mathsf{halt}] \\
&\geqslant (1 - e^{-n}) \cdot \mathsf{Pr}[\mathsf{halt}] + (\frac{1}{2} + \mathsf{Adv}_{G_3,G_4}(\mathcal{A})) \cdot \mathsf{Pr}[\neg\mathsf{halt}] \\
&\geqslant \min\left(\frac{1}{2} + \mathsf{Adv}_{G_3,G_4}(\mathcal{A}), 1 - e^{-n}\right) .
\end{aligned}$$

Since we assume that no poly(λ)-time adversary $\mathcal{A}$ can win the hiding game with non-negligible advantage, we have $\frac{1}{2} + \mathsf{Adv}_{G_3,G_4}(\mathcal{A}) \leqslant 1 - e^{-n}$; hence,

$$\mathsf{Adv}_{G_3,G_4}(\mathcal{A}) \leqslant \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A}) . \tag{4.7}$$

Game G_5: Game G_5 is the same as Game G_4 except the following. For each honest voter $V_\ell \in \tilde{\mathcal{V}}$, Ch picks $\tilde{\mathcal{U}}_\ell$ at random, and uses $\tilde{\mathcal{U}}_\ell$ in the **Cast** protocol, ignoring the adversary's $\langle \mathcal{U}_\ell^0, \mathcal{U}_\ell^1 \rangle$. Regardless the coin b, Ch always uses the simulator $\mathcal{S}$ to transform the view $\mathsf{view}_\ell = (\mathsf{Pub}, s_\ell, \tilde{\mathcal{U}}_\ell, \alpha_\ell)$ to $\mathsf{view}'_\ell = (\mathsf{Pub}, s'_\ell, \mathcal{U}_\ell^{b'}, \alpha_\ell)$.

It is obvious that the view of G_5 is identical to the view of G_4, so

$$\mathsf{Adv}_{G_4,G_5}(\mathcal{A}) = 0 \; . \tag{4.8}$$

Notice that all the vote-codes are generated at random and all the commitments in each ballots commit to the same value (0 or T). Moerover, since the view of G_5 does not depend on the challenger's coin b, we have the probability that $\mathcal{A}$ guess b correctly is

$$\mathsf{Pr}[\mathcal{A} = 1 | G_5] = \frac{1}{2} \; . \tag{4.9}$$

To sum up, the total running time of our reduction is $\mathsf{poly}(\lambda) \cdot 2^t$.
By Eq. (4.3),(4.4),(4.5),(4.6),(4.7),(4.8),(4.9), we have that

$$\begin{aligned} \mathsf{Pr}[G^{\mathcal{A},\mathcal{S}}_{t\text{-}\mathsf{priv}}(1^\lambda, n, m) = 1] &= \mathsf{Pr}[\mathcal{A} = 1 | G_5] + \sum_{i=1}^{5} \mathsf{Adv}_{G_{i-1},G_i}(\mathcal{A}) \\ &\leqslant \frac{1}{2} + (2nm \log m + 1) \cdot \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A}) = \frac{1}{2} + \mathsf{negl}(\lambda). \end{aligned}$$

■

4.7 Discussion

The DEMOS-A e-voting system significantly advances e-voting security allowing for the first time, the voters to verify the election procedure without trusting any external assumptions. Besides its theoretic contribution, DEMOS-A is an e-voting system designed to be applied in the real world, thus usability has been an important parameter in all its current implementations (European and National Election pilot experiments, e-voting services for the University of Athens, e-voting services for the Greek Workers Union). There, the voter enjoys the flexibility of choosing to cast her vote either (i) with a simple button click after being prompted to user-friendly web interface or (ii) by directly typing the vote-code that corresponds to the option of her preference, in case she does not trust her client. In addition, as any vote-code based e-voting, DEMOS-A has the advantage of requiring voter clients of minimum computational power since the voting procedure is just vote-code submission from the voter side.

Human behaviour and DEMOS-A security :

An intriguing observation regarding E2E verifiability of DEMOS-A (cf. Theorem 4.4) is its direct dependence on the number of voters that not only they are honest, but also actively participate in the verifications process. Moving a step forward, one could study

the security of DEMOS-A according to the ceremony aspect modelled in Section 3.6 by properly customising human behaviour e.g., the coin flip entropy, or the auditing probability of an honest voter. In this thesis, we will not proceed to a formal security approach of DEMOS-A modelled as an e-voting ceremony, as the design of Helios e-voting system is a more interesting case of study from this aspect (cf. Chapter 6). However, we provide some intuition on the effect of the human factor in DEMOS-A by introducing the simple generalisation on voters' behaviour below:

- For every honest voter V, the probability that V chooses any part in her ballot is in $[\alpha, 1-\alpha]$, where $\alpha \in [0, 1/2]$.
- For every honest voter V, the probability that V performs verification is at least β.

By the above, an analogue statement of Theorem 4.4 would be expressed as follows:

Theorem. *Assume an election run of DEMOS-A with n voters, m candidates and k trustees. Let q be the size of the group for the commitment scheme described in Section 4.2.1. Then, DEMOS-A achieves E2E verifiability information theoretically for θ honest successful voters and tally deviation δ with error*

$$\big((1-\alpha)\beta\big)^{\delta} + \epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta \log \alpha) \ , \tag{4.10}$$

where $\epsilon(\cdot,\cdot,\cdot,\cdot)$ is the soundness error of the Σ protocol given in Theorem 4.2.
Roughly, the error bound derives from the fact that the entropy of each honest voter is now $\log \alpha$ instead of 1 bit. Therefore, Theorem 4.2 is applied for min entropy $\theta \log \alpha$, resulting in the soundness error bound $\epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta \log \alpha)$. If no invalid attacks happen, then at least δ honest voters must verify correctly while under modification or clash attack of their vote. The probability that the adversary guesses a voter's coin flip is no more than $1-\alpha$ while every single honest voter has β probability of auditing.

Observe that the error bound of Theorem 4.4 is a special case of Eq. (4.10) for $\alpha = 1/2$ and $\beta = 1$.

Limitations of DEMOS-A :

As any vote=code e-voting system, DEMOS-A comes with a performance and storage penalty for the EA compared to client-side encryption systems such as Helios [2]. The main reason is that due to the way the EA forms the proof of the tally result, it is required to precompute a number of ciphertexts for each voter and each possible choice of the voter. This approach clearly does not scale to elections that have a complex ballot and voters have an exponential number of ways to vote in the number of candidates. As an indication, we provide benchmark results for election preparation step in Table 4.1. The result are obtained by the original DEMOS-A implementation, designed by Bingsheng Zhang on a Debian server with Intel i7-4700HQ 2.4 GHz, 16GB RAM, when run for small to medium scale elections. We can expect that DEMOS-A preparation time will reach the order of hours for much smaller scales than elections at a national level.

Table 4.1: Election preparation time benchmarks for DEMOS-A.

n	m	**Curve**	**Preparation Time**
1000	2	p192	21 seconds
1000	5	p192	91 seconds
1000	10	p192	220 seconds
10000	2	p192	3.5 minutes
10000	5	p192	15 minutes
10000	10	p192	36 minutes

In order to overcome the implementation limitations that appear in DEMOS-A, this thesis introduces an alternative e-voting system that uses client-side encryption like Helios, thus it transfers the computational overhead to the voters' clients. The system, named *DEMOS-2*, is equipped with a novel mechanism for constructing NIZK proofs which preserves the E2E verifibiality level of DEMOS-A in the standard model. The following chapter is dedicated to the detailed description and security analysis of DEMOS-2.

5. THE DEMOS-2 E-VOTING SYSTEM

The design of DEMOS-2 is motivated by the scalability problems that stem from the implementation restrictions of DEMOS-A (cf. Section 4.7). DEMOS-2 turns to client-side encryption to obviate the need for a demanding precomputation step from the EA at the setup phase, thus it allows for deployment at a large scale. As a result, from a functionality aspect, the system shares the characteristics of the well-known web-based Helios e-voting system [2] (cf. Chapter 6 for an extended study of Helios in the ceremony model introduced in Section 3.6).

The major contribution of DEMOS-2 is that it encompasses the scalability advantages of e-voting via client-side encryption, while retaining the strong E2E verifability characteristics of DEMOS-2, i.e., it removes the reliance to the RO model for security that Helios and other similar systems assume [41, 67, 36, 96, 12]. This is achieved by introducing a new technique for proving the validity of ciphertexts that are submitted by the voters during ballot casting (that may have applications beyond the e-voting domain).

In few words, DEMOS-2 utlises a type of NIZK where there are two possible ways to generate the CRS; one that makes every NIZK *perfectly sound* and another that makes every NIZK *simulatable* using the trapdoor information associated with the CRS. The EA uses this *dual mode* feature, publishing a master CRS of the first type, i.e., one that makes all NIZKs perfectly sound. In order to prove the validity of the CRS, EA engages in a Σ protocol along the lines of DEMOS-A, where the challenge is extracted from the voters' coins. During the voting phase, the master CRS will function as a public-key of an additively homomorphic encryption scheme that will be used for sound ballot generation by the voters' VSDs.

CHAPTER ROADMAP. At first, we provide a high-level overview of DEMOS-2 (cf. Section 5.1) and all the underlying cryptographic tools (cf. Section 5.2). Then, we present DEMOS-2 in at length and prove its correctness (cf. Section 5.3). We prove the E2E veribiability and the simulation-based voter privacy/PCR presented in Sections 3.3 and 3.5 respectively (cf. Sections 5.4 and 5.5). Conclusively, we provide evidence of the efficiency of DEMOS-2 and discuss its complementary relation with DEMOS-A (cf. Section 5.6).

5.1 Overview of DEMOS-2

DEMOS-2 is a client-side encryption web-based system. We assume there is a secure channel between the EA, VC election servers and each trustee, say, realized by HTTPS. The system uses homomorphic tally and currently supports x-out-of-m type of option selection. Let m_{min} and m_{max} be the minimum and maximum number of options that is allowed to choose to vote.

At the setup phase, the EA needs to login to the election server and provides the election definition. An election definition consists of question, options, (m_{min}, m_{max}), start/end time,

trustee list (including their email addresses), and voter list (including their voter IDs and email addresses). The election server then creates a unique election ID, eid, selects a bilinear group parameter for the election, $\sigma_{\mathsf{bp}} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathsf{Gen}_{\mathsf{bp}}(1^\lambda)$, and posts σ_{bp} on BB. (Note that σ_{bp} is hard-coded in our prototype.) The election server then generates and sends a random 128-bit credential to each trustee by email, inviting them to setup the election parameters. Upon receiving the credential, each trustee authenticates himself to the server and executes the election parameters setup process. Once all the trustees jointly setup the election parameters, the EA triggers the server to send an invitation email (with the voter ID, $\mathsf{vid}_\ell \in \mathbb{G}_2$ and a freshly generated random 128-bit credential s_ℓ) to each voter V_ℓ, where vid_ℓ is a random group element in $\mathbb{G}_2$ generated by the election server. At the end of the setup phase, each trustee T_i is able to check the consistency between the posted election parameters on the BB and its private state st_i.

At the ballot-casting phase, each voter V_ℓ uses (vid_i, s_ℓ) to authenticate herself to the VC election server. Next, she prepares and casts her vote using the voter supporting device $\mathsf{VSD}_e ll$. The voters' ballots are prepared locally in the VSD_ℓ and are posted to the in the BB by the VC.

When voting is finished, the tally phase is initiated. The VC computes the tally ciphertexts by multiplying all the valid submitted ciphertexts for each option on the BB Note that, during this step, any invalid ciphertexts and duplicated ciphertexts are removed. The voters' coins are used to produce a *Sigma protocol challenge*, as in the challenge extraction mechanism of DEMOS-A, hence no trust in an external source of randomness (oracle or beacon) is required. The trustees are then invited by the VC to complete their Sigma protocols and decrypt the tally ciphertexts . Note that each trustee should respond to this invitation using a secure channel such as HTTPS. Upon receiving such a message from a trustee, the election server checks the validity of all the Σ proofs and NIZK proofs, and rejects it in case some of the proofs are invalid. The election server posts all the received trustees' messages to the BB only after all the trustees have successfully completed their **Tally** protocols.

After the election end, the tally result can be computed according to standard threshold ElGamal decryption (cf. Subsection 5.2.4) using the partial decryption of the tally ciphertexts from each trustees. Each voter V_ℓ is able to fetch the election transcript, **info** from BB and verify the integrity of the election with its individual audit information; the voter checks if the data in her ballot hashes to the rec_ℓ and the validity of all the Sigma proofs and NIZK proofs.

5.2 Building Blocks of DEMOS-2

We introduce the cryptographic tools required for the construction of DEMOS-2. Throughout the section, we consider

- Cyclic group parameters that consist of the description $\langle \mathbb{G} \rangle$, the prime order q and

the generator g of some multiplicative cyclic group $\mathbb{G}$, output by a group generator $\mathsf{GGen}(1^\lambda)$.

- The bilinear groups $\mathbb{G}_1, \mathbb{G}_2$ and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ as in Definition 2.6. The pairing parameters $\sigma_{\mathsf{bp}} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where p is the prime order and g_1, g_2 are the generators of $\mathbb{G}_1, \mathbb{G}_2$, are produced by an execution of the bilinear group generator $\mathsf{BGGen}(\cdot)$.

5.2.1 Cryptographic hash function

DEMOS-2's design considers an arbitrary cryptographic hash function (cf. Subsection 2.3.1), assuming that no known algorithm can find a collision within $2^{2\kappa}$ expected steps. In the following lemma, we show that such a hash function preserves the min entropy H_∞ of its inputs.

Lemma 5.1. *Let X be an n-bit efficiently samplable distribution with $H_\infty(X) \geqslant \kappa$. Let* $\mathsf{hash} : \{0,1\}^* \mapsto \{0,1\}^\lambda$ *be a cryptographic hash function, where $\lambda > 2\kappa$ is a security parameter. If there is no algorithm that can find a collision for* hash *within $2^{2\kappa}$ expected number of steps with more than* $\mathsf{negl}(\lambda)$ *probability, then it holds that $H_\infty(\mathsf{hash}(X)) \geqslant \kappa$.*

Proof. Since $H_\infty(\mathsf{hash}(X)) < \kappa \leqslant H_\infty(X)$, we have that

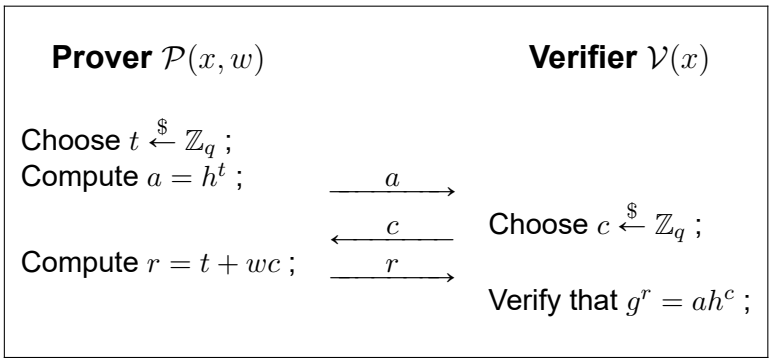
$$\exists \sigma : \mathsf{Pr}[x \leftarrow X : \mathsf{hash}(x) = \sigma] > 2^{-\kappa} \quad \text{and} \quad \forall x : \mathsf{Pr}[x \leftarrow X] \leqslant 2^{-\kappa}.$$

Therefore, σ must have collisions.

Consider the algorithm $\mathcal{A}_\sigma$ that repetitively samples x from X at random and stores $\mathsf{hash}(x)$, trying to find a collision for σ. Given that $\mathsf{Pr}[x \leftarrow X : \mathsf{hash}(X) = \sigma] > 2^{-\kappa}$, the expected running time for $\mathcal{A}_\sigma$ to find a collision for the hash image σ is less than $2^{2\kappa}$. ■

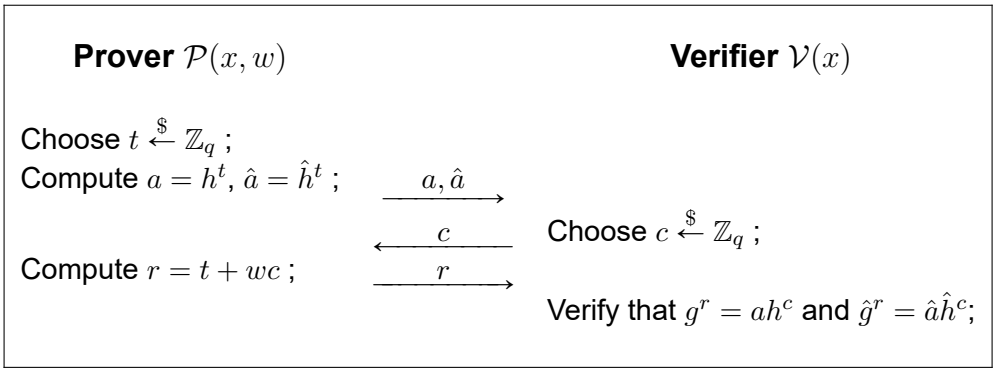
5.2.2 Schnorr proof of knowledge of a DLOG

The Schnorr protocol for proving the knowledge of a discrete logarithm [90], is perhaps the best-known Σ protocol, widely used in cryptography as an identification scheme. Given as parameters the description $\langle \mathbb{G} \rangle$ of some multiplicative cyclic group $\mathbb{G}$ of prime order q and for some statement $x = (g, h) \in \mathbb{G}^2$, the prover $\mathcal{P}$ on private input w, convinces the verifier $\mathcal{V}$ of knowledge of $\log_g h = w$ via the following interaction:



5.2.3 Chaum-Pedersen proof of DLOG equality

A Chaum-Pedersen (CP) proof [31], is a Σ protocol for proving the equality of discrete logarithms. Given as parameters the description $\langle \mathbb{G} \rangle$ of some multiplicative cyclic group $\mathbb{G}$ of prime order q and for some statement $x = (g, h, \hat{g}, \hat{h}) \in \mathbb{G}^4$, the prover $\mathcal{P}$ on private input w, convinces the verifier $\mathcal{V}$ of the equality $\log_g h = \log_{\hat{g}} \hat{h} = w$ via the following interaction:



Alternatively, CP protocol can be seen as a proof that $(g, h, \hat{g}, \hat{h})$ *forms a DDH tuple*. Indeed, $\mathbb{G}$ is cyclic, so if $\log_{g_1} h_1 = \log_{g_2} h_2 = w$, then for some a it holds that

$$(g, h, \hat{g}, \hat{h}) = (g, g^w, g^a, g^{aw}) \ .$$

A CP proof can be applied for proving the correct encryption of a standard (lifted) *ElGamal ciphertext* $(C_1, C_2) = (g^t, g^M h^t)$ over the group $\mathbb{G}$, where g is a generator of $\mathbb{G}$, h is the random element in $\mathbb{G}$ used as public key, M is the plaintext and $t \in \mathbb{Z}_q$ is the randomness used in the encryption process. Namely, the encryptor provides a CP proof of the equality

$$\log_g C_1 = \log_h (C_2 / g^M) \ .$$

5.2.4 Threshold ElGamal encryption

The threshold ElGamal cryptosystem, which DEMOS-2 deploys in its simplest (k, k) version, is the most common instantiation of a (t, k)-TPKE scheme, used in various client-side encryption e-voting systems (e.g. [41, 67, 2, 36, 96, 12]) for ballot encryption. The scheme consists of the following algorithms, following the syntax in Subsection 2.3.6 :

- TPKE.Gen(1^λ):
  - The servers (trustees) agree on scheme parameters that consist of the description $\langle \mathbb{G} \rangle$, the prime order q and the generator g of some multiplicative cyclic group $\mathbb{G}$, output by the group generator GGen(1^λ) ;
  - Every server Ser$_i$, $i \in [k]$ chooses $x_i \xleftarrow{\$} \mathbb{Z}_q$ and computes $h_i = g^{x_i}$;
  - The partial secret key/public key pair for Ser$_i$ is $(\mathsf{sk}_i, \mathsf{pk}_i) \triangleq \big(x_i, (g, q, h_i)\big)$;
- TPKE.Combine$(\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$:
  - Compute $h = \prod_{i \in [k]} h_i = g^{\sum_{i \in [k]} x_i}$;
  - Set the public key pk $\triangleq (g, q, h)$;
- TPKE.Enc(pk, M), where M is selected from a small message space $\mathcal{M}$:
  - Choose $r \xleftarrow{\$} \mathbb{Z}_q$;
  - Generate a standard lifted ElGamal ciphertext $C = (C_1, C_2) = \big(g^r, g^M h^r\big)$;
- TPKE.Dec(sk_i, C):
  - Compute $M_i = C_1^{x_i} = g^{r x_i}$;
  - Generate a CP *proof of valid partial decryption* π_i for the DLOG equality
    $$\log_g(h_i) = \log_{C_1}(M_i) \ .$$
  - Output $D_i = (C_2, M_i, \pi_i)$;
- TPKE.Recon$(D_1, \ldots, D_k)$:
  - Verify all CP proofs $\pi_1, \ldots, \pi_k$. If some proof is invalid, then output $\perp$;
  - Compute $Y = C_2 \cdot \big(\prod_{i \in [k]} M_i\big)^{-1} = g^M$;
  - Reconstruct the plaintext M as the value y s.t. $g^y = Y$ by performing an exhaustive search in $\mathcal{M}$;

The correctness of the scheme is straightforward. Moreover, the (k, k)-threshold ElGamal cryptosystem is IND-CPA secure as long as the DDH assumption holds for the underlying group generator GGen. We stress that in the special case of DEMOS-2 ballot encryption where M is a bit ($\mathcal{M} = \{0, 1\}$), the plaintext reconstruction is done simply by setting $M = 0$, if $Y = 1$ and 1 otherwise.

5.2.5 Dual ElGamal homomorphic commitment scheme

Similar to [58], we use the lifted ElGamal additively homomorphic public key cryptosystem, to construct a commitment scheme with the following "dual" property: depending on the way the commitment key is generated, the commitment scheme can be either (i) perfectly binding or (ii) perfectly hiding with a *trapdoor* that allows the commitment to be successfully opened at any value. The *dual ElGamal commitment scheme* $\mathcal{EG}^{\vee}$ consists of the following algorithms:

- CS.Gen(1^{λ}):
  - Choose a message m of poly(λ) size ;
  - Generate ElGamal cyclic group $\mathbb{G}$ with prime order p and generator g ;
  - Choose $x \xleftarrow{\$} \mathbb{Z}_p$ and generate public key pk $= (p, g, h = g^x)$;
  - Generate an ElGamal encryption $u = (u_1, u_2) = (g^t, g^d h^t) \in \mathbb{G}^2$ of m under pk with randomness $t \in \mathbb{Z}_p$;
  - Output ck $\triangleq$ (pk, u) ;
  - Set trapdoor td $\triangleq t$;
- CS.Com(ck, $m; r$): Output $c = u^m \cdot \mathsf{Enc}_{\mathsf{pk}}(0; r) = (u_1^m g^r, u_2^m h^r)$;
- CS.Ver$\big($ck, $c, (m, r)\big)$: If $c = (u_1^m g^r, u_2^m h^r)$, then output accept ; else, output reject ;

It is easy to check that $\mathcal{EG}^{\vee}$ is additively homomorphic. In addition, the following properties hold for $\mathcal{EG}^{\vee}$.

Proposition 5.1. *Let* $c =$ CS.Com(ck, $m; r$) *be an* $\mathcal{EG}^{\vee}$ *commitment to some value* m *under the commitment key* ck $=$ (pk, u)*, where* ck *is generated as described in algorithm* CS.Gen(1^{λ})*. Then,*

1. *If* u *is an encryption of a non-zero value* x*, then* c *it is perfectly binding.*
2. *If* u *is an encryption of 0, then* c *is perfectly hiding with trapdoor* td $= t$*.*

Proof. The ElGamal ciphertext u has the form of a pair $(u_1, u_2) = (g^t, g^d h^t)$, i.e., it is an encryption of a value d, with randomness t. Hence, the commitment $c = u^m \cdot \mathsf{Enc}_{\mathsf{pk}}(0; r)$ is an encryption $(g^{mt+r}, g^{dm} h^{mt+r})$ of dm with randomness $mt + r$. If $d \neq 0$, then c is perfectly binding by the correctness of ElGamal encryption. If $d = 0$ is an encryption of 0, then for every m, $c = (g^{mt+r}, h^{mt+r}) = (g^{mt+r}, g^{x(mt+r)})$ follows the uniform distribution in $\mathbb{G}^2$, thus it is perfectly hiding. In addition, given trapdoor td $= t$, c can be successfully opened at any value m' by setting $r' = r - (m' - m)t$. ■

5.2.6 A NIZK for DDH Tuple

We construct a NIZK proof where the prover $\mathcal{P}$ convinces the verifier $\mathcal{V}$ that the statement $(A, B, C, D) \in (\mathbb{G}_i)^4$, $i \in \{1, 2\}$ is a DDH tuple. Namely, $\mathcal{P}$ proves the knowledge of some (witness) $s \in \mathbb{Z}_p$ s.t. $C = A^s \wedge D = B^s$. In the description of the DDH NIZK proof, we consider the case where $\mathbb{G}_i = \mathbb{G}_1$, as it is applied in DEMOS-2 construction, even though the case where $\mathbb{G}_i = \mathbb{G}_2$ is similar.

Our proof can be seen as a simplification of the well-known Groth-Sahai (GS) proof system [59]. The CRS of the DDH NIZK proof consists of the bilinear group parameter, σ_{bp} and the dual ElGamal commitment key, $\mathsf{ck} := (\mathsf{pk}, u)$. The CRS is *perfectly sound* when the perfectly binding commitment key is used, while it is *perfectly simulatable* when the perfectly hiding commitment key is used. Formally, the NIZK proof system Γ^{ddh} consists of the following PPT algorithms:

- The CRS generator $\mathsf{CRS.Gen}^{\mathsf{ddh}}(1^\lambda)$:
  - Execute $\sigma_{\mathsf{bp}} \triangleq (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathsf{BGGen}(1^\lambda)$;
  - Pick $\alpha_1, \alpha_2 \leftarrow \mathbb{Z}_p^*$;
  - Set $h_2 = g_2^{\alpha_1}$ and $u = (u_1, u_2) = (g_2^{\alpha_2}, g_2 h_2^{\alpha_2})$;
  - Set $\mathsf{ck} = (h_2, u)$;
  - Output $\mathsf{crs} \triangleq (\sigma_{\mathsf{bp}}, \mathsf{ck})$;
- The CRS simulator $\mathcal{S}_{\mathsf{crs}}^{\mathsf{ddh}}(\sigma_{\mathsf{bp}})$:
  - Pick $\alpha_1, \alpha_2 \leftarrow \mathbb{Z}_p^*$;
  - Set $h_2 = g_2^{\alpha_1}$ and $u = (u_1, u_2) = (g_2^{\alpha_2}, h_2^{\alpha_2})$;
  - Set $\mathsf{ck} = (h_2, u)$;
  - Output $\mathsf{crs}^* \triangleq ((\sigma_{\mathsf{bp}}, \mathsf{ck})$ and $\mathsf{td} \triangleq \alpha_2$;
- The prover $\mathcal{P}^{\mathsf{ddh}}(\mathsf{crs}, (A, B, C, D), s)$:
  - Pick $r \leftarrow \mathbb{Z}_p$;
  - Set $c = (c_1, c_2) = \mathsf{CS.Com}(\mathsf{ck}, s; r) = (u_1^s g_2^r, u_2^s h_2^r)$, $\pi_1 := A^r$, and $\pi_2 := B^r$;
  - Output $\pi \triangleq (c, \pi_1, \pi_2)$;
- The verifier $\mathcal{V}^{\mathsf{ddh}}(\mathsf{crs}, (A, B, C, D), \pi)$:
  - Output 1 if and only if the following hold:

    $e(C, u_1) \cdot e(\pi_1, g_2) = e(A, c_1)$; $\quad e(C, u_2) \cdot e(\pi_1, h_2) = e(A, c_2)$;

    $e(D, u_1) \cdot e(\pi_2, g_2) = e(B, c_1)$; $\quad e(D, u_2) \cdot e(\pi_2, h_2) = e(B, c_2)$;
- The verifier simulator $\mathcal{S}^{\mathsf{ddh}}(\mathsf{crs}^*; (A, B, C, D); \mathsf{td})$:

- Pick $r \leftarrow \mathbb{Z}_p$;
- Set $c^* = (c_1, c_2) = (g_2^r, h_2^r)$, $\pi_1^* = A^r C^{-\alpha_2}$, and $\pi_2^* = B^r D^{-\alpha_2}$;
- Output $\pi^* \triangleq (c^*, \pi_1^*, \pi_2^*)$;

Clearly, the simulated CRS is computationally indistinguishable from the real CRS based on the IND-CPA security of the underlying ElGamal cryptosystem. We state the following theorem without providing the proof since it can be directly derive from the generic GS proof for the SXDH instantiation in [59] (cf. Definition 2.9).

Theorem 5.1. *The protocol Γ^{ddh} is a NIZK proof system for the language*

$$\mathcal{L}^{\mathsf{ddh}} = \left\{(A, B, C, D) \in (\mathbb{G}_1)^4 \mid \exists s : C = A^s \wedge D = B^s\right\},$$

i.e. (A, B, C, D) is a DDH tuple. The NIZK proof has perfect completeness, perfect soundness and computational zero-knowledge under the SXDH assumption.

5.2.7 NIZK OR Composition

In the ballot generation and tally decryption step of DEMOS-2, OR composition of the NIZK proofs is needed, e.g., to show a lifted ElGamal ciphertext in $(\mathbb{G}_1)^2$ (resp. $(\mathbb{G}_2)^2$) is an encryption of 0 or 1. To achieve this, we adopt the *correlated key generation* technique from [58][1]. The intuition is to use two tiers of NIZK proofs, where the CRS for the first tier NIZK is given as the *master CRS*. To prove an OR composition of statements such as $x_1 \vee \ldots \vee x_n$, the prover first generates n second tier CRS's, $\mathsf{crs}_1, \ldots, \mathsf{crs}_n$ and uses the master CRS to show that at least one of them is a perfectly sound CRS; the prover then uses the second tier CRS crs_i to prove the statement x_i for $i \in [n]$. Since the prover is able to generate $n-1$ perfectly simulatable CRS's with trapdoors, it can simulate any $n-1$ statements. On the other hand, at least one of the crs_i is perfectly sound, so at least one of the statement x_i is valid. The ZK property directly implies the fact that it is computationally hard to distinguish which CRS is perfectly sound.

More specifically, the prover gives n lifted ElGamal ciphertexts as the n second tier CRS, and shows the product of them is an encryption of 1 using the DDH tuple NIZK described in Subsection 5.2.6. Therefore, we can ensure that at least one of the CRS encrypts an non-zero value. In the following, we describe two special cases of OR composition that we apply in DEMOS-2.

5.2.7.1 Proving that a ciphertext encrypts 0 or 1

We describe the NIZK proof system $\Gamma^{0/1}$ for the ciphertext $c = \mathsf{Enc}_{\mathsf{pk}}(b; r) \in (\mathbb{G}_1)^2$ encrypts 0 or 1, i.e., $b \in \{0, 1\}$.

[1] We refer interested readers to [86] for more general NIZK composition via correlated key generation.

- The CRS generator $\mathsf{CRS.Gen}^{0/1}(1^\lambda)$:
  - Use $\mathbb{G}_1$ variant of $\mathsf{Gen}^{\mathsf{ddh}}_{\mathsf{crs}}(\sigma_{\mathsf{bp}})$ to produce a master CRS $\mathsf{crs_m}$ in $\mathbb{G}_1$;
- The CRS simulator $\mathcal{S}^{0/1}_{\mathsf{crs}}(\sigma_{\mathsf{bp}})$:
  - Use $\mathbb{G}_1$ variant of $\mathcal{S}^{\mathsf{ddh}}_{\mathsf{crs}}(\sigma_{\mathsf{bp}})$ to produce a simulated CRS crs^*_m in $\mathbb{G}_1$ and a trapdoor td ;
- The prover $\mathcal{P}^{0/1}(\mathsf{crs_m}; (\mathsf{pk} := (g_1, f_1), c); (b, r))$:
  - Pick $\alpha_1, \alpha_2, \alpha_3 \leftarrow \mathbb{Z}_p$;
  - Set $h_2 := g_2^{\alpha_1}$, $u^{(b)} := (u_1^{(b)}, u_2^{(b)}) = (g_2^{\alpha_2}, g_2 h_2^{\alpha_2})$, and $u^{(1-b)} := (u_1^{(1-b)}, u_2^{(1-b)}) = (g_2^{\alpha_3}, h_2^{\alpha_3})$;
  - Set $\mathsf{ck}^{(b)} := (h_2, u^{(b)})$ and $\mathsf{ck}^{(1-b)} := (h_2, u^{(1-b)})$;
  - Define $\mathsf{crs}^{(b)} := (\sigma_{\mathsf{bp}}, \mathsf{ck}^{(b)})$ and $\mathsf{crs}^{(1-b)} := (\sigma_{\mathsf{bp}}, \mathsf{ck}^{(1-b)})$;
  - Set $(u_1, u_2) = u^{(b)} \cdot u^{(1-b)} \in (\mathbb{G}_2)^2$;
  - Compute $\pi_{\mathsf{crs}} \leftarrow \mathsf{Prov}^{\mathsf{ddh}}(\mathsf{crs_m}; (g_2, h_2, u_1, u_2/g_2); \alpha_2 + \alpha_3)$;
  - Set $\pi^{(b)} \leftarrow \mathsf{Prov}^{\mathsf{ddh}}(\mathsf{crs}^{(b)}; (g_1, f_1, c_1, c_2/g_1^b); r)$; and $\pi^{(1-b)} \leftarrow \mathcal{S}^{\mathsf{ddh}}(\mathsf{crs}^{(1-b)}; (g_1, f_1, c_1, c_2/g_1^{1-b}); \alpha_3)$;
  - Output $\pi := (\mathsf{crs}^{(0)}, \mathsf{crs}^{(1)}, \pi_{\mathsf{crs}}, \pi^{(0)}, \pi^{(1)})$;
- The verifier $\mathcal{V}^{0/1}(\mathsf{crs_m}; (\mathsf{pk} := (g_1, f_1), c); \pi)$:
  - Output 1 if and only if the following verify:
    (i). $\mathsf{Vrfy}^{\mathsf{ddh}}(\mathsf{crs_m}, (g_2, h_2, u_1, u_2/g_2), \pi_{\mathsf{crs}})$=1 ;
    (ii). $\mathsf{Vrfy}^{\mathsf{ddh}}(\mathsf{crs}^{(0)}, (g_1, f_1, c_1, c_2), \pi^{(0)}) = 1$;
    (iii). $\mathsf{Vrfy}^{\mathsf{ddh}}(\mathsf{crs}^{(1)}, (g_1, f_1, c_1, c_2/g_2), \pi^{(1)}) = 1$;
- The voter simulator $\mathcal{S}^{0/1}(\mathsf{crs}^*_m; (\mathsf{pk} := (g_1, f_1), c); \mathsf{td})$:
  - Pick $\alpha_1, \alpha_2, \alpha_3 \leftarrow \mathbb{Z}_p$;
  - Set $h_2 = g_2^{\alpha_1}$, $u^{(0)} = (u_1^{(0)}, u_2^{(0)}) = (g_2^{\alpha_2}, h_2^{\alpha_2})$, and $u^{(1)} := (u_1^{(1)}, u_2^{(1)}) = (g_2^{\alpha_3}, h_2^{\alpha_3})$;
  - Set $\mathsf{ck}^{(0)} := (h_2, u^{(0)})$ and $\mathsf{ck}^{(1)} := (h_2, u^{(1)})$;
  - Define $\mathsf{crs}^{(0)} := (\sigma_{\mathsf{bp}}, \mathsf{ck}^{(0)})$ and $\mathsf{crs}^{(1)} := (\sigma_{\mathsf{bp}}, \mathsf{ck}^{(1)})$;
  - Set $(u_1, u_2) = u^{(0)} \cdot u^{(1)} \in (\mathbb{G}_2)^2$;
  - Compute $\pi_{\mathsf{crs}} \leftarrow \mathcal{S}^{\mathsf{ddh}}(\mathsf{crs}^*_m; (g_2, h_2, u_1, u_2/g_2); \mathsf{td})$;
  - Set $\pi^{(0)} \leftarrow \mathcal{S}^{\mathsf{ddh}}(\mathsf{crs}^{(0)}; (g_1, f_1, c_1, c_2); \alpha_2)$; and $\pi^{(1)} \leftarrow \mathcal{S}^{\mathsf{ddh}}(\mathsf{crs}^{(1)}; (g_1, f_1, c_1, c_2/g_1); \alpha_3)$;

– Output $\pi^* := (\mathsf{crs}^{(0)}, \mathsf{crs}^{(1)}, \pi_{\mathsf{crs}}, \pi^{(0)}, \pi^{(1)})$;

Theorem 5.2. *The protocol $\Gamma^{0/1}$ is a NIZK proof system for c encrypts 0 or 1. The NIZK proof has perfect completeness, perfect soundness and computational zero-knowledge under the SXDH assumption.*

Proof.

1. *Perfect completeness.* It directly follows from the completeness and simulatability of the underlying NIZK proof Γ^{ddh}.

2. *Perfect soundness.* The prover generates two CRSs, $\mathsf{crs}^{(0)}$ and $\mathsf{crs}^{(1)}$, and uses Γ^{ddh} to show that the product of them is lifted ElGamal encryption of 1. Since Γ^{ddh} is perfect sound, it is sure that at least one CRS encrypts to a non-zero value. By simultaneously showing the given ciphertext c is encryption of 0 and 1 with respect to $\mathsf{crs}^{(0)}$ and $\mathsf{crs}^{(1)}$, we guarantee that c encrypts either 0 or 1.

3. *Computational ZK.* It is straightforward that if the SXDH assumption holds, then $\mathsf{crs}^{(0)}$ and $\mathsf{crs}^{(1)}$ are computationally indistinguishable (hence DDH is hard for $\mathbb{G}_2$) and the simulated CRS crs^*_m is computationally indistinguishable from the real one crs_m. Moreover, the Γ^{ddh} is computationally zero-knowledge, so all the simulated sub-proofs are indistinguishable from the real ones. Therefore, π^* is computationally indistinguishable from π. ■

5.2.7.2 Proving that a ciphertext encrypts a value between min and max

Observe that this case is a generalization of $\Gamma^{0/1}$, where we set $\mathsf{min} = 0$ and $\mathsf{max} = 1$. The description follows the lines of $\Gamma^{0/1}$ where now we generate $\mathsf{max} - \mathsf{min} + 1$ CRSs denoted by $\mathsf{crs}^{(\mathsf{min})}, \ldots, \mathsf{crs}^{(\mathsf{max})}$. For $j \in [\mathsf{min}, \mathsf{max}]$, $\mathsf{crs}^{(j)}$ contains σ_{bp} and the commitment key $\mathsf{ck}^{(j)}$, which in turn consists of a random element $h_2 \in \mathbb{G}_2$ and an ElGamal encryption of j, $u^{(j)}$. We denote such NIZK proofs as $\Gamma^{\mathsf{min}\,/\,\mathsf{max}}$.

5.2.8 Lapidot-Shamir Revisited

A critical point in the desing of DEMOS-A is the mecahnism that allows the EA to prove the validity of cryptographic elements on the BB using Σ protocols. In particular, the EA posts the commitment messages of Σ protocols in the BB before the election starts; During the election, the verifier's challenge is jointly contributed by all the voters (1 bit per honest voter); After the election ends, the EA then completes the Σ protocols by posting the corresponding response messages in the BB. However, this technique has its limitations; namely, the statement to be proven must be fixed before the election starts. In order to prove a statement that is generated during or after the election, we need a generic 3-move ZK protocol whose commitment message is independent of the statement, such as the Lapidot-Shamir protocol[2] [74]. Unfortunately, one has to convert the original language

[2]Technically, the size of the commitment message of the Lapidot-Shamir protocol still depends on the statement.

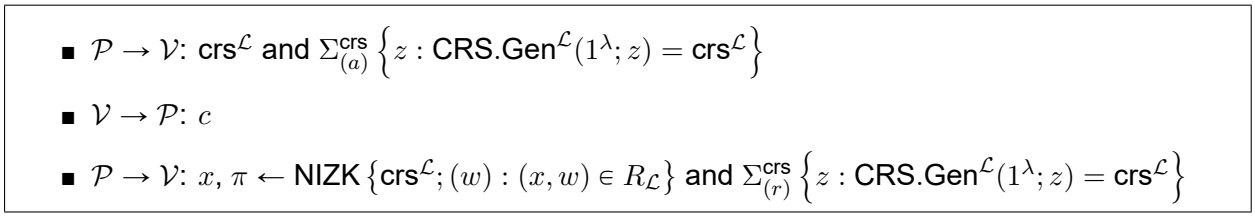


Figure 5.1: The message structure of the composed 3-move ZK for $\mathcal{L}$.

to Hamiltonian cycle in order to use the Lapidot-Shamir protocol, so it is very inefficient in practice.

To resolve this issue, we propose a new Lapidot-Shamir like 3-move ZK framework where the prover's first move does not depend on the statement to be proven. The idea is to combine a 3-move public coin HVZK protocol with a perfectly sound NIZK proof. For notation simplicity, we will use Σ protocol notation for such 3-move public coin HVZK protocols, but we emphasise that the special soundness and special ZK properties are not necessary for our composition.Let $\Gamma^{\mathcal{L}}$ be a perfectly sound NIZK proof system for some NP language $\mathcal{L}$ with witness relation $R_{\mathcal{L}}$, and let

$$\Sigma^{\mathsf{crs}}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}$$

be a Σ protocol to show the given $\mathsf{crs}^{\mathcal{L}}$ is a perfectly sound CRS. The message structure of the composed 3-move ZK protocol between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ is depicted in Figure 5.1. In the first move, the prover $\mathcal{P}$ generates a NIZK CRS $\mathsf{crs}^{\mathcal{L}}$ and sends it to the verifier $\mathcal{V}$ together with the commitment message of $\Sigma^{\mathsf{crs}}_{(a)}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}$. In the second move the verifier $\mathcal{V}$ gives the challenge c. In the third move, the prover $\mathcal{P}$ fixes the statement $x \in \mathcal{L}$ and computes the NIZK proof, $\pi \leftarrow \mathsf{NIZK}\left\{\mathsf{crs}^{\mathcal{L}}; (w) : (x, w) \in \mathcal{R}\right\}$, for x. Subsequently, $\mathcal{P}$ sends to $\mathcal{V}$ the statement x, the NIZK proof π, and the response message of $\Sigma^{\mathsf{crs}}_{(r)}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}$. Finally, $\mathcal{V}$ accepts the proof if

(i). $\left(\Sigma^{\mathsf{crs}}_{(a)}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}, \mathsf{Ch}, \Sigma^{\mathsf{crs}}_{(r)}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}\right)$ is a valid Σ protocol transcript and

(ii). $\mathcal{V}^{\mathcal{L}}(\mathsf{crs}^{\mathcal{L}}, x, \pi) = \mathsf{accept}$.

Theorem 5.3. *Let $\Gamma^{\mathcal{L}}$ be a perfectly complete, perfectly sound, and computationally zero-knowledge NIZK proof system for language $\mathcal{L}$, and let $\Sigma^{\mathsf{crs}}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}$ be a 3-move public coin HVZK protocol with perfectly completeness, statistical soundness, and computational ZK. The composed 3-move public coin HVZK protocol for language $\mathcal{L}$ in Figure 5.1 achieves perfect completeness, statistical soundness, and computational ZK.*

Proof.

1. *Perfect completeness.* It directly follows from the perfect completeness properties of both $\Gamma^{\mathcal{L}}$ and $\Sigma^{\mathsf{crs}}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}$.

2. *Perfect soundness.* Since $\Sigma^{\mathsf{crs}}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}$ is statistically sound, $\mathsf{crs}^{\mathcal{L}}$ is a perfectly sound CRS with overwhelming probability. When $\mathsf{crs}^{\mathcal{L}}$ is a perfectly sound CRS, no adversary can produce a fake π^* to make the verifier accept an invalid $x^* \notin \mathcal{L}$ such that $\mathcal{V}^{\mathcal{L}}(\mathsf{crs}^{\mathcal{L}}, x^*, \pi^*) = \mathsf{accept}$. Hence, the composed ZK is statistically sound.

3. *Computational ZK.* The simulatable CRS crs^* generated by $\mathcal{S}^{\mathcal{L}}_{\mathsf{crs}}(\sigma_{\mathsf{bp}})$ is computationally indistinguishable from a perfectly sound CRS. From the computationally zero-knowledge properties of both $\Gamma^{\mathcal{L}}$ and $\Sigma^{\mathsf{crs}}\left\{z : \mathsf{CRS.Gen}^{\mathcal{L}}(1^{\lambda}; z) = \mathsf{crs}^{\mathcal{L}}\right\}$, it is easy to see that the simulated composed ZK proof is computationally indistinguishable from a real one. ■

5.3 Description of DEMOS-2

According to the syntax in Section 3.1, DEMOS-2, built upon the cryptographic primitives presented in Section 5.2, consists of the following protocols and algorithms. Throughout our description we use $\Sigma^{\mathsf{dlog}}_{(a)}$, $\Sigma^{\mathsf{ddh}}_{(a)}$ to denote the commitment message and $\Sigma^{\mathsf{dlog}}_{(r)}$, $\Sigma^{\mathsf{ddh}}_{(r)}$ to denote the response message of a Schnorr protocol and a CP proof respectively.

The **Setup**$(1^{\lambda}, \mathcal{O}, \mathcal{V}, \mathcal{U}, \mathcal{T})$ *protocol* :

The election parameters generation does not require the interaction between the trustees, and each trustee T_i only needs to interact with the EA. At first, the EA generates and then sends a random 128-bit credential to each trustee inviting them to setup the election parameters. Next, the interaction is completed two rounds. In particular,

Round 1

- Each trustee T_i performs the following:
  - Pick random $\alpha_i, \beta_i \leftarrow \mathbb{Z}_q$;
  - Set $h_{1,i} = g_1^{\alpha_i}$ and $u_{0,i} = g_1^{\beta_i}$;
  - Post/Append $h_{1,i}, u_{0,i}$ to the public election parameters on the BB together with the following Σ commitment messages:

    $$\Sigma^{\mathsf{dlog}}_{(a)}\{(\alpha_i) : h_{1,i} = g_1^{\alpha_i}\}\ ; \qquad \Sigma^{\mathsf{dlog}}_{(a)}\left\{(\beta_i) : u_{0,i} = g_1^{\beta_i}\right\}\ ;$$

- The EA computes and posts in the BB:
  - $\mathsf{pk} \triangleq (g_1, h_1 := \prod_{i=1}^{k} h_{1,i})$;
  - $u_0 = \prod_{i=1}^{k} u_{0,i}$;

Round 2

- Each trustee T_i performs the following:
  - Pick random $\gamma_i \leftarrow \mathbb{Z}_q$;
  - Set $u_{1,i} = g_1^{\gamma_i}$ and $u_{2,i} = u_0^{\gamma_i}$;
  - Post/Append $u_{1,i}, u_{2,i}$ to the public election parameters on the BB together with the following Σ commitment message:
    - $\Sigma_{(a)}^{\mathsf{ddh}}\{(\gamma_i) : u_{1,i} = g_1^{\gamma_i} \wedge u_{2,i} = u_0^{\gamma_i}\}$;

Upon termination, each trustee T_i keep its working tape as its private state st_i. After all the trustees have participated in **Setup**, the EA:

- Computes $\mathsf{ck} = (u_0, (\prod_{i=1}^{k} u_{1,i}, g_1 \cdot \prod_{i=1}^{k} u_{2,i}))$;
- Posts master CRS $\mathsf{crs_m} \stackrel{\Delta}{=} (\sigma_{\mathsf{bp}}, \mathsf{ck})$ on the BB;

Generating Voters' Private Information. For every voter V_ℓ, $i \in [n]$, the election server generates (i) the voter ID, $\mathsf{vid}_\ell \in \mathbb{G}_2$ by selecting a random group element in $\mathbb{G}_2$ and (ii) a freshly generated random 128-bit authentication code s_ℓ. It distributes $\mathsf{cr}_\ell \stackrel{\Delta}{=} (\mathsf{vid}_\ell, s_\ell)$, $\ell \in [n]$ to all the voters and provides VC with $\{\mathsf{cr}_\ell\}_{\ell \in [n]}$.

The **Cast** *protocol* :

The voting supporting device VSD_ℓ of V_ℓ fetches election parameters from the BB and works as a "voting booth". The VSD_ℓ shows the election question and a list of options to the voter V_ℓ. The voter V_ℓ can select $x \in [m_{\mathsf{min}}, m_{\mathsf{max}}]$ options and let the VSD_ℓ prepare the ballots. Let $\mathbf{e} = (e_1, e_2 \ldots, e_m)$ be the characteristic vector corresponding to the voter's selection, where $e_j = 1$ if the option opt_j is selected and $e_j = 0$ otherwise. The VSD_ℓ prepares two versions of the ballot that encrypts the same option selection as follows.

- For $j \in \{1, 2, \ldots, m\}$:
  - Pick random $r_{j,(0)}, r_{j,(1)} \leftarrow \mathbb{Z}_p$;
  - Compute $c_j^{(0)} = (c_{j,1}^{(0)}, c_{j,2}^{(0)}) = (g_1^{r_{j,(0)}}, g_1^{e_j} h_1^{r_{j,(0)}})$;
  - Compute $c_j^{(1)} = (c_{j,1}^{(1)}, c_{j,2}^{(1)}) = (g_1^{r_{j,(1)}}, g_1^{e_j} h_1^{r_{j,(1)}})$;
- Given a cryptographic hash function hash, compute two strings of individual audit information:
  - $\mathsf{audit}_{(0)} = \mathsf{hash}(\mathsf{eid}, \mathsf{vid}_\ell, \text{'A'}, c_1^{(0)}, \ldots, c_m^{(0)})$;
  - $\mathsf{audit}_{(1)} = \mathsf{hash}(\mathsf{eid}, \mathsf{vid}_\ell, \text{'B'}, c_1^{(1)}, \ldots, c_m^{(1)})$;

Namely, VSD_ℓ presents to the voter the individual audit information for both A and B versions of the ballot, $\mathsf{audit}_{(0)}$ and $\mathsf{audit}_{(1)}$; meanwhile, it displays two buttons labelled as 'A' and 'B' respectively. The voter should keep the individual audit information and then randomly choose one of the buttons to proceed. Suppose the voter chooses 'A' (resp. 'B'); then, VSD_ℓ opens the version B (resp. A) of the ballot by revealing the randomness used to create all the ciphertexts in version B (resp. A), $r_{1,(1)}, \ldots, r_{m,(1)}$ (resp. $r_{1,(0)}, \ldots, r_{m,(0)}$). The voter can export the data and use any third-party auditing software to perform the check.

Next, VSD_ℓ prepares the NIZK proofs for the version A of the ballot (the computation for version B is similar); for $j \in [m]$, it computes the $\Gamma^{0/1}$ NIZK proofs (cf. Subsection 5.2.7.1)

$$\pi_j^{(0)} \leftarrow \mathsf{NIZK}\Big\{\mathsf{crs_m}; (e_j, r_{j,(0)}) : c_j^{(0)} = \mathsf{Enc_{pk}}(e_j; r_{j,(0)}) \wedge e_j \in \{0,1\}\Big\}\,.$$

Note that in above $\Gamma^{0/1}$ NIZK proofs, $\mathcal{P}^{0/1}$ uses the vid_ℓ as the h_2 in the description of Section 5.2.7.1 instead of generating a fresh $h_2 = g_2^{\alpha_1}$ for $\mathsf{crs}^{(0)}$ and $\mathsf{crs}^{(1)}$ every time. It then sets $c^{(0)} = \prod_{j=1}^m c_j^{(0)}$, $e = \sum_{j=1}^m e_j$, and $r_{(0)} = \sum_{j=1}^m r_{j,(0)}$, and computes the $\Gamma^{\mathsf{min}\,/\,\mathsf{max}}$ NIZK proof

$$\pi^{(0)} \leftarrow \mathsf{NIZK}\Big\{\mathsf{crs_m}; (e, r_{(0)}) : c^{(0)} = \mathsf{Enc_{pk}}(e; r_{(0)}) \wedge e \in [m_{\mathsf{min}}, m_{\mathsf{max}}]\Big\}\,.$$

VSD_ℓ submits the ballot $\mathbf{B}_\ell \triangleq \Big\langle \mathsf{vid}_\ell, \text{'A'}, \big\{c_j^{(0)}, \pi_j^{(0)}\big\}_{j\in[m]}, \pi^{(0)} \Big\rangle$ along with the authentication code s_ℓ of V_ℓ to the VC. Subsequently, VC verifies the validity of s_ℓ and if so, it posts $\mathbf{B}_\ell$ in the BB. The voter's individual audit information is defined as $\mathsf{audit}_\ell \triangleq (\mathsf{vid}_\ell, \text{'A'}, \mathsf{audit}_{(0)})$ (resp. $\mathsf{audit}_\ell \triangleq (\mathsf{vid}_\ell, \text{'A'}, \mathsf{audit}_{(1)})$) assuming version A (resp. B) of the ballot was selected during the **Cast** protocol.

The **Tally** *protocol* :

Producing the Sigma Protocols Challenge. After the voting phase is finished, the voters' coins are collected to produce the Sigma protocol challenge. On the BB, everyone can identify the version of each submitted ballot. We interpret 'A' as 0, 'B' and 1, and if the voter did not submit a ballot, his coin is fixed as 0. Denote ρ_j as the voter V_j's coin and $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$. As in DEMOS-A, the voters' coins can be modelled as an *adaptive non-oblivious bit fixing source*. Nevertheless, we still want to produce a single challenge if only computationally bounded adversaries are considered. Assume that there is no known algorithm that can find a collision of hash within $2^{2\kappa}$ expected steps. We compute the challenge $\mathsf{Ch} \leftarrow \mathsf{hash}(\rho)$. By Theorem 5.1, if $H_\infty(\rho) \geqslant \kappa$, then $H_\infty(\mathsf{hash}(\rho)) \geqslant \kappa$.

Finalizing the Election. The VC computes the tally ciphertexts by multiplying all the valid submitted ciphertexts for each option on the BB. The tally ciphertexts are denoted by $(E_1, \ldots, E_m)$, where $E_j = (E_{j,1}, E_{j,2})$. Next, each trustee T_i fetches all the posted information from BB and checks its consistency and executes the following steps:

- It computes and posts the following response messages in the BB:

- $\Sigma_{(r)}^{\mathsf{dlog}}\{(\alpha_i) : h_{1,i} = g_1^{\alpha_i}\}$;
- $\Sigma_{(r)}^{\mathsf{dlog}}\left\{(\beta_i) : u_{0,i} = g_1^{\beta_i}\right\}$;
- $\Sigma_{(r)}^{\mathsf{ddh}}\{(\gamma_i) : u_{1,i} = g_1^{\gamma_i} \wedge u_{2,i} = u_0^{\gamma_i}\}$;

▸ For $j \in \{1, \ldots, m\}$:

- It computes and posts the partial decryption $D_{j,i} = E_{j,1}^{\gamma_i}$ together with the Γ^{ddh} NIZK proof

$$\pi_{j,i} \leftarrow \mathsf{NIZK}\left\{\mathsf{crs_m}; (\gamma_i) : h_{1,i} = g_1^{\gamma_i} \wedge D_{j,i} = E_{j,1}^{\gamma_i}\right\} .$$

After all the trustees partial decryption of the tally ciphertexts has been posted, the VC, for $j \in [m]$, computes

$$R_j = \mathsf{log}_{g_1}\left(E_{j,2} / \prod_{i=1}^{k} D_{j,i}\right) .$$

The discrete logarithm can be solved in approximately $\sqrt[2]{n}$ steps, given the knowledge that $R_j \in [0, n]$, as there are maximum n possible votes for each option in total. It then posts the final tally $\mathbf{R} = (R_1, \ldots, R_m)$ in the BB.

The **Result**(τ) *algorithm* :

Given the BB data posted by the trustees, the values $R_1, \ldots, R_m$ can be computed by any party.

The **Verify**$(\tau, \mathsf{audit}_\ell)$ *algorithm* :

After the **Setup** protocol, each trustee T_i is able to check the consistency between the posted election parameters on the BB and its private state st_i. The voter checks the following:

1. There is a unique ballot $\mathbf{B}_\ell$ indexed by vid_ℓ in the election transcript **info**.
2. The data in $\mathbf{B}_\ell$ hashes to the rec_ℓ.
3. There is no duplicated ciphertexts and NIZK proofs across the entire election transcript **info**.
4. All the NIZK proofs in each ballot $\mathbf{B}_\ell$ uses vid_ℓ as a part of the second layer CRS's.
5. All the Σ and NIZK proofs are valid.

5.3.1 Correctness of DEMOS-2

Assuming, that all parties are honest, the correctness of DEMOS-2 follows from the correctness of the dual ElGamal commitment and the threshold ElGamal cryptosystems, as well as the completeness of the underlying Σ protocols and NIZK proofs.

5.4 E2E Verifiability of DEMOS-2

For simplicity, our analysis is for 1-out-of-m elections, and it can be easily extended to x-out-of-m cases. Our proof strategy follows the lines of Theorem 4.4, as DEMOS-2 shares many common elements with DEMOS-A. The main difference is that even though still in the standard model, E2E verifiability in DEMOS-2 holds as long as the applied cryptographic hash function is collision resistant.

5.4.1 Attacks on verifiability

Besides the trivial attacks that the adversary may follow, i.e. the ones that will be detected with certainty (e.g. malformed or unreadable election transcript) and violating the collision resistance of the hash function, the meaningful types of attack that an adversary may launch against DEMOS-2 include attacks on the NIZK proofs and the modification and clash attacks already described in the security analysis of DEMOS-A (cf. Subsection 5.4.1), now adopted to the client-side encryption setting.

- *NIZK attack*: the adversary attempts to generate a malformed ballot that contains invalid ciphertexts (e.g., multiple votes for some specific candidate) or post invalid partial decryption shares. This attack can be prevented by the soundness of the Γ^{ddh}, $\Gamma^{0/1}$, $\Gamma^{\mathsf{min}\,/\,\mathsf{max}}$ NIZK proofs. The proof verification is done via a trusted auditing supporting device (ASD).
- *Modification attack*: the adversary modifies one of the versions of the honest voters' ballots when it was produced on the VSD by encrypting a valid vote but for a different option than the one the voter intended. This attack is successful only if the voter chooses to submit the modified version. The deviation achieved by this type of attack is at most 1, whereas the probability of detection is 1/2.
- *Clash attack*: the adversary assigns the same vid to y honest voters so that the adversary can inject $y-1$ ballots. This attack is successful only if all the y voters verify the same ballot on the BB and hence miss the injected votes that produce the tally deviation. The maximum deviation achieved by this attack is $y-1$, whereas the probability of detection is $1-2^{y-1}$ (at least two out of the y voters choose a different version to vote).

Remark 5.1 (***Completeness of the attack list***)**.** The above list exhausts all possible attack strategies against DEMOS-2. This is because if all ballots and BB data are consistently generated and all ballot information is tabulated in the BB indexed, the adversary can only perform a combination of modification and clash attacks on the honest votes. If no such combination occurs, then all honestly cast votes are in correct (yet unknown) one-to-one correspondence with the BB audit data, hence by the perfect correctness of the ElGamal TPKE scheme, the opening of the homomorphic tally matches the intended result.

5.4.2 End-to-end verifiability theorem

Having described all possible attack scenarios, we prove the E2E verifiability of DEMOS-2 in the following theorem.

Theorem 5.4. *DEMOS-2 run with n voters, m candidates and k trustees achieves E2E verifiability for at least θ honest successful voters and tally deviation δ with error $2^{-\delta}+2^{-\theta}+$ negl(λ) , unless there is an algorithm that can find a collision for* hash $: \{0,1\}^* \mapsto \{0,1\}^\lambda$ *within $2^{2\theta}$ expected number of steps with more than* negl(λ) *probability.*

Proof. Recall that in the E2E verifiability threat model (cf. Section 3.3), only BB is assumed to be honest, while the rest administration entities are controlled by the adversary. Hence, the voter ID, vid$_\ell$, may not necessarily be unique, and the adversary is allowed to change the content on the BB arbitrarily before the **Tally** protocol starts. Nevertheless, we can assume all the Sigma protocols and the NIZK proofs on the BB are valid if there is at least one honest voter that performs verification. We first construct a vote extractor $\mathcal{E}$ for our system as follows:

Construction of the vote extractor for DEMOS-2 :

$\mathcal{E}$ on input τ and the set of individual audit information $\{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}$, where $\mathcal{V}_{\text{succ}}$ is the set of the honest voters that voted successfully, operates as follows:

The vote extractor $\mathcal{E}\left(\tau, \{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}\right)$ for DEMOS-2

1. Let $t \leqslant |\mathcal{V}_{\text{succ}}|$ be the number of different tags that appear in $\{\text{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}$. This implies that the ballot audit for all voters in $\mathcal{V}_{\text{succ}}$ focuses on a list of t tabulated ballots on the BB (thus, an adversary may inject $|\mathcal{V}_{\text{succ}}| - t$ ballots for candidate selections of its choice that will be counted in the final tally as if they were honest).
2. If **Result**$(\tau) = \perp$ (i.e., the transcript is not meaningful), then $\mathcal{E}$ outputs $\perp$.
3. For all the corrupted voters $V_\ell \in \mathcal{V} \backslash \mathcal{V}_{\text{succ}}$, $\mathcal{E}$ extracts $\mathcal{U}_\ell$ by exhaustive search over the ElGamal ciphertexts in the ballot $\mathbf{B}_\ell$.
4. $\mathcal{E}$ outputs $\langle \mathcal{U}_\ell \rangle_{V_\ell \in \mathcal{V} \backslash V_\ell \in \mathcal{V}_{\text{succ}}}$.

Based on the above vote extractor, we now prove the E2E verifiability of our scheme. Assume an adversary $\mathcal{A}$ that wins the game $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^\lambda, m, n, k)$ described in Figure 3.1. Namely, $\mathcal{A}$ breaks E2E verifiability by allowing at least θ honest successful voters and achieving tally deviation δ.

Let F be the event that there exists one tallied ciphertext that encrypts $e^* \notin \mathcal{U}$. By Theorems 5.1 and 5.2, all the NIZK proofs are perfectly sound. Hence, the adversary needs to break the soundness of at least one of the Sigma protocols to make event F occur. By Lemma 5.1 and since the voters' coins have min entropy θ, the Sigma protocols challenge hash(ρ) should also have min entropy θ, unless there is an algorithm that can find

a collision for hash in $2^{2\theta}$ expected number of steps. Hence, each Sigma protocol has soundness error no more than $2^{-\theta}$. Therefore,

$$\mathsf{Pr}\left[G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^{\lambda},m,n,k)=1 \mid F\right] \leqslant 2^{-\theta}\ . \tag{5.1}$$

Now assume that F does not occur. In this case, the deviation from the intended result that $\mathcal{A}$ achieves, derives only by miscounting the honest votes. This may be achieved if $\mathcal{A}$ performs combinations of modification and clash attacks.

Recall that each honest voter should select one of the two versions of the ballot at random, and the other version will be opened for auditing. Hence, the success probability of x deviation via modification attacks is 2^{-x}. With regard to the clash attacks, similarly, it is easy to see that the success probability to clash y honest voters without being detected is $2^{-(y-1)}$ (all y honest voters choose the same version to vote). Given that F does not occur the total tally deviation achieved is $x+y \geqslant \delta$. Therefore, the upper bound of the success probability of $\mathcal{A}$ when F does not occur is

$$\mathsf{Pr}\left[G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^{\lambda},m,n,k)=1 \mid \neg F\right] \leqslant 2^{-(x+y)} \leqslant 2^{-\delta}. \tag{5.2}$$

By Eq. (5.1), (5.2), we have the overall probability

$$\mathsf{Pr}\left[G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\delta,\theta}(1^{\lambda},m,n,k)=1\right] \leqslant 2^{-\delta}+2^{-\theta}\ .$$

■

5.5 Simulation-based Voter Privacy/PCR of DEMOS-2

DEMOS-2 achieves simulation-based voter privacy/PCR according to Definition 3.5. Similarly to DEMOS-A, complexity leveraging is deployed. Specifically, we choose the security parameters such that breaking the SXDH assumption of $\mathsf{Gen}_{\mathsf{bp}}$ and finding a collision for hash is much harder than guessing the challenge of the Σ protocols. Before, proving our simulation-based privacy theorem, we provide formal arguments of the non-malleability of the NIZK proofs used in DEMOS-2, which protects the system form *replay attacks*, analogous to the ones pointed out in [39] for the case of Helios e-voting system [2].

5.5.1 On the non-malleability of the NIZK Proofs

It is well-known that Groth-Sahai proofs [59] are malleable with respect to the same CRS. More specifically, given a GS proof, π for statement x w.r.t. crs, anyone can re-randomise the proof to produce a distinct proof π^* for x respect to crs. To prevent replay attacks [39], all the duplicated ciphertexts shall be removed. However, the adversary can still copy and re-randomise some honest voters' ciphertexts as well as their attached NIZK proofs if the same CRS is used among all the voters. To address this issue, each voter is required to use a distinct vid_{ℓ} as a part of her second layer CRSs.

Regarding privacy, recall that we assume the election servers (EA and BB) are honest; in particular, all the voter ID's $\{\mathsf{vid}_\ell\}_{\ell\in[n]}$ should be generated honestly such that no one knows the discrete logarithms: $\log_{g_2}(\mathsf{vid}_\ell)$ for all $\ell \in [n]$ and $\log_{\mathsf{vid}_{\ell_1}}(\mathsf{vid}_{\ell_2})$ for all $\ell_1 \neq \ell_2 \in [n]$. We show that, given $c = \mathsf{Enc}_{\mathsf{pk}}(b)$ for an unknown $b \in \{0,1\}$ together with a proof π generated by the NIZK proof system $\Gamma^{0/1}$ using vid_1, no PPT adversary can produce $\hat{c} = \mathsf{Enc}_{\mathsf{pk}'}(b)$, where $\mathsf{pk}' \neq \mathsf{pk}$, and $\hat{\pi}$ that includes vid_2 as a part of its second layer CRS's with non-negligible probability.

Recall that in the NIZK proof system $\Gamma^{0/1}$ (cf. Subsection 5.2.7.1), the prover generates $\mathsf{crs}^{(0)}$ and $\mathsf{crs}^{(1)}$ and via a DDH NIZK proof Γ^{ddh} (cf.Subsection 5.2.6), shows that the ciphertext c encrypts 0 using $\mathsf{crs}^{(0)}$ and c encrypts 1 using $\mathsf{crs}^{(1)}$. Since Γ^{ddh} proof is perfectly sound, if c encrypts b, the proof that uses $\mathsf{crs}^{(b)}$ must be perfectly sound and the proof that uses $\mathsf{crs}^{(1-b)}$ must be simulatable. By the description of the NIZK proof system Γ^{ddh}, $\mathsf{crs}^{(b)}$ and $\mathsf{crs}^{(1-b)}$ must be encryptions of 1 and 0 respectively under the "public key", $\mathsf{pk}_1 = \mathsf{vid}_1$. Similarly, in $\hat{\pi}$, $\hat{\mathsf{crs}}^{(b)}$ and $\hat{\mathsf{crs}}^{(1-b)}$ must be encryptions of 1 and 0 respectively under the "public key", $\mathsf{pk}_2 = \mathsf{vid}_2$. Hence, the non-malleability problem is reduced to the following theorem.

Lemma 5.2. *Given randomly chosen* $\mathsf{pk}_1, \mathsf{pk}_2$ *and* $c_0 = \mathsf{Enc}_{\mathsf{pk}_1}(x)$, $c_1 = \mathsf{Enc}_{\mathsf{pk}_1}(1-x)$ *for unknown* $x \in \{0,1\}$*, the probability that a PPT adversary* $\mathcal{A}$ *produces* $\hat{c}_0 = \mathsf{Enc}_{\mathsf{pk}_2}(x)$, $\hat{c}_1 = \mathsf{Enc}_{\mathsf{pk}_2}(1-x)$ *is negligible, if the underlying encryption scheme is IND-CPA secure.*

Proof. The proof is via reduction. Assume there is a PPT adversary $\mathcal{A}$ who can produce $\hat{c}_0 = \mathsf{Enc}_{\mathsf{pk}_2}(x)$, $\hat{c}_1 = \mathsf{Enc}_{\mathsf{pk}_2}(1-x)$. Then, we can construct an adversary $\mathcal{B}$ who can win the IND-CPA game of the underlying encryption scheme as follows:

1. In the IND-CPA game, $\mathcal{B}$ is given pk_1 and it sends $x_0 = 0, x_1 = 1$ to the IND-CPA challenger $\mathsf{Ch}^{\mathsf{ind}}$. $\mathcal{B}$ will receive $c_0 = \mathsf{Enc}_{\mathsf{pk}_1}(x_b)$ from $\mathsf{Ch}^{\mathsf{ind}}$ and will be challenged to guess b.
2. $\mathcal{B}$ computes $c_1 = \mathsf{Enc}_{\mathsf{pk}_1}(1)/c_0 = \mathsf{Enc}_{\mathsf{pk}_1}(1-x_b)$ and generates $(\mathsf{sk}_2, \mathsf{pk}_2)$.
3. Next, $\mathcal{B}$ sends c_0, c_1, pk_1 and pk_2 to $\mathcal{A}$. Upon receiving $\hat{c}_0$ and $\hat{c}_1$ from $\mathcal{A}$, it decrypts $\hat{c}_0$ and $\hat{c}_1$.
4. Finally, $\mathcal{B}$ sends b' to $\mathcal{C}$, if $\hat{c}_0$ and $\hat{c}_1$ are indeed encryptions of b' and $1-b'$, otherwise it sends random $b' \leftarrow \{0,1\}$ to $\mathsf{Ch}^{\mathsf{ind}}$.

Clearly, $\mathcal{B}$ wins when $\mathcal{A}$ succeeds (i.e., $\hat{c}_0$ and $\hat{c}_1$ are indeed encryptions of b' and $1-b'$). Therefore if the probability that $\mathcal{A}$ wins is p, we have that

$$\begin{aligned}\Pr[\mathcal{B}\text{ wins}] &= \Pr[\mathcal{A}\text{ succeeds}]\cdot\Pr[\mathcal{B}\text{ wins}|\mathcal{A}\text{ succeeds}] + \Pr[\mathcal{A}\text{ fails}]\cdot\Pr[\mathcal{B}\text{ wins}|\mathcal{A}\text{ fails}] = \\ &= p\cdot 1 + (1-p)\cdot 1/2 = 1/2 + p/2.\end{aligned}$$

Consequently, if the underlying encryption scheme is IND-CPA secure, then p must be a negligible value. ■

5.5.2 Simulation-based voter privacy/PCR theorem

Theorem 5.5. *Assume an election run of DEMOS-2 with n voters, m candidates and k trustees. Assume there exists a constant κ, $0 < \kappa < 1$ such that for any 2^{λ^κ}-time adversary $\mathcal{A}$ the advantage of breaking the SXDH assumption of the bilinear group generator* BGGen *is* negl(λ). *Then, for every constant κ' s.t. $0 < \kappa' < \kappa$ and every $t \leqslant \lambda^{\kappa'}$, DEMOS-2 achieves $(k-1, t)$-simulation based voter privacy/PCR.*

Proof. Given a 2^{λ^κ}-time adversary $\mathcal{A}$ against the k-privacy of DEMOS-2, we construct a simulator $\mathcal{S}$ s.t. IDEAL$_{\mathcal{F}_{\text{priv}},\mathcal{S},\mathcal{Z}}(\lambda)$ and REAL$_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda)$ are computationally ingistinguishable. Let T_h be the honest trustee and $\mathcal{V}_{\text{corr}}$ be the set of voters that $\mathcal{A}$ corrupts. W.l.o.g., we assume that $\mathcal{A}$ completes the real-world experiment with some non-negligible probability p. Indeed, if $\mathcal{A}$ almost always aborts, then we can construct a simulator for $\mathcal{A}$ that also aborts with overwhelming probability.

The construction of view simulator $\mathcal{S}$:

The simulator $\mathcal{S}$ executes the following stages:

1. *Casting of honest votes in the ideal experiment.*

Upon receiving $(sid, \text{vote}, \mathcal{O}, \mathcal{V}, \mathcal{U})$ from $\mathcal{F}_{\text{priv}}$, $\mathcal{S}$ does not initiate the real world simulation of an execution of DEMOS-2 for $\mathcal{A}$ until all honest votes in the ideal experiment have cast their votes. Namely, when $\mathcal{S}$ receives $(sid, \text{cast}, V_\ell)$ from $\mathcal{F}_{\text{priv}}$, it directly replies with the same message $(sid, \text{cast}, V_\ell)$.

2. *First run: activating $\mathcal{A}$.*

After all ideal honest voters have cast their votes, $\mathcal{S}$ activates $\mathcal{A}$ in the first run of a real world simulation. If $\mathcal{A}$ aborts at any moment of the first run, then $\mathcal{S}$ stops the simulation and provides the environment with $\mathcal{A}$'s output.
Specifically, $\mathcal{S}$ simulates the **Setup** protocol in DEMOS-2 playing the role of the BB. It generates $\mathcal{T} = \{T_1, \ldots, T_t\}$ and allows $\mathcal{A}$ to corrupt all the trustees except from T_h. When simulating the steps of T_h, $\mathcal{S}$ performs the following modifications: it sets $u_{2,h} = u_0^{\gamma_h}/g_1$ and simulates a proof for the fake DDH relation of $(g_1, u_0, u_{1,h}, u_{2,h})$. As a result, the master CRS crs$_{\text{m}}$ consists of the bilinear group parameters σ_{bp} and the commitment key:

$$\text{ck} \triangleq \Big(u_0, \big(\prod_{i=1}^{k} u_{1,i}\ , g_1 \cdot \prod_{i=1}^{k} u_{2,i}\big)\Big) = \Big(u_0, \big(\prod_{i=1}^{k} g_1^{\gamma_i}, g_1 \cdot \big(\prod_{i \in k \setminus \{w\}} u_0^{\gamma_i}\big) \cdot (u_0^{\gamma_h}/g_1)\big)\Big) =$$

$$= \big(u_0, (g_1^{\gamma}, u_0^{\gamma})\big), \quad \text{where } \gamma = \sum_{i=1}^{k} \gamma_i\ .$$

Therefore, ck contains an encryption of 0 under u_0 (instead of an encryption of 1, as in the normal execution), thus the master CRS crs$_m$ is perfectly simulatable. Note that the witnesses γ_i, $i \in [k] \setminus \{w\}$ are not yet known to $\mathcal{S}$. Once all the trustees have completed

their **Setup**, $\mathcal{S}$ generates vid_ℓ such that $d_\ell = \log_{g_2}(\text{vid}_\ell)$ is known to $\mathcal{S}$. Then, it sends the credentials to all the voters.

Next, $\mathcal{S}$ selects all the voters' coins (including both honest and corrupted voters) at random, denoted as $\rho = (\rho_1, \ldots, \rho_n) \in \{0,1\}^n$ and produces the challenge of the Sigma protocols using ρ.
During the **Cast** protocol, $\mathcal{S}$ plays the role of the EA and BB. For every honest voter $V_\ell \notin \mathcal{V}_{\text{corr}}$, $\mathcal{S}$ executes a **Cast** protocol on behalf of V_ℓ for some fixed and invalid input (i.e. not a candidate selection) $\hat{\mathcal{U}} \notin \mathcal{U}$. In the **Cast** protocol, $\mathcal{S}$ uses the pre-generated coins $\rho_1, \ldots, \rho_n$ to simulate the NIZK proofs for $\hat{\mathcal{U}} \notin \mathcal{U}$. In case the corrupted voters' coins do not match the pre-generated (guessed) coins, $\mathcal{S}$ rewinds the state of $\mathcal{A}$ at the beginning of the **Cast** protocol and starts over the voting phase. After all the voters have cast their ballots, $\mathcal{S}$ plays the role of the EA and T_h interacting with the corrupted trustees in the **Tally** protocol. Importantly, $\mathcal{S}$ sends suitably long messages to EA to fake the **Tally** interaction for T_h. Due to the secure channel between T_h and the EA, $\mathcal{A}$ cannot tell whether T_h's **Tally** protocol is fake.

After all the corrupted trustees finish the **Tally** protocol, $\mathcal{S}$ does not post their tally messages to the BB. Instead, it operates as follows.

3. *Second run: extracting the adversarial witnesses in the real world experiment.*

$\mathcal{S}$ stores the set of the transcripts of all the Σ protocols . Then, it rewinds the state of $\mathcal{A}$ at the beginning of the **Cast** protocol and begins a second run of the simulation from that start of the voting phase. If $\mathcal{A}$ aborts at any moment of the second run, then $\mathcal{S}$ stops the simulation and starts a fresh real world simulation, thus returning to the beginning of a new first run as before. Otherwise, the second run simulation proceeds as follows:

(i). The voting phase is executed again as in the first run using the pre-generated coins $\rho_1, \ldots, \rho_n$ and rewinded if the corrupted voters' coins do not match the guess. If there is a hash collision of the fresh challenge of the Σ protocol with that of the first run, then $\mathcal{S}$ aborts.

(ii). $\mathcal{S}$ completes normally the rest of the simulation until all the corrupted trustees finish the **Tally** protocol.

As a result, if the second run is completed successfully, $\mathcal{S}$ obtains another set of the transcripts of all the Σ protocols with a different challenge. Subsequently, $\mathcal{S}$ utilises the knowledge extractor of the Σ protocol to extract all the corrupted trustees' witnesses $\alpha_i, \beta_i, \gamma_i$, $i \in [k] \backslash \{w\}$. Upon witness extraction, $\mathcal{S}$ computes $\gamma = \sum_{i=1}^{k} \gamma_i$. Recall that now the master CRS crs_m contains an encryption of 0 , thus it is perfectly simulatable using γ as the trapdoor.

4. *Extracting and forwarding the adversarial votes to* $\mathcal{F}_{\text{priv}}$.

After extracting the adversarial witnesses, $\mathcal{S}$ is able to learn all the adversarial votes. Namely, for every corrupted voter V_ℓ, $\mathcal{S}$ uses d_ℓ to decrypt all the ciphertexts in her ballot

$\mathbf{B}_\ell$ on the BB, and thus determine $\mathcal{U}_\ell$. Note that $\mathcal{U}_\ell \notin \mathcal{U}$, otherwise then $\mathcal{S}$ would have discarded the ballot at the voting phase. Then, $\mathcal{S}$ sends $(sid, \text{cast}, V_\ell, \mathcal{U}_\ell)$ to $\mathcal{F}_{\text{priv}}$.

5. *Finalizing the real world experiment.*

$\mathcal{S}$ sends (sid, tally) to $\mathcal{F}_{\text{priv}}$. Upon receiving the election result $\mathbf{R} = (R_1, \ldots, R_m)$ from $\mathcal{F}_{\text{priv}}$, $\mathcal{S}$ computes $D^*_{j,h} = E_{j,2}/(g_1^{\tau_j} \cdot E_{j,1}^{\sum_{i \neq h} \alpha_i})$ for $j \in [m]$ and simulates the corresponding NIZK proofs $\pi^*_{j,h}$, $j = 1, \ldots, m$. Finally, $\mathcal{S}$ posts $D^*_{j,w}$ on the BB.

To complete the proof, we need the following two claims:

Claim 5.5.1: *With high probability, $\mathcal{S}$ will terminate successfully in $O\big(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda)\big)$ steps.*

Proof of Claim 5.5.1: It suffices to show that (i) the probability $\mathcal{S}$ runs in $\omega\big(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda)\big)$ steps is negligible and (ii) the probability $\mathcal{S}$ will abort in $O\big(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda)\big)$ steps is negligible. We anayse both cases:

(i). The probability $\mathcal{S}$ runs in $\omega\big(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda)\big)$ steps is negligible.

Let p be the npn-negligible probability that $\mathcal{A}$ does not abort the real world experiment. By the computational indistinguishability of the simulated CRS, the probability that $\mathcal{A}$ successfully completes the first run of the simulated experiment is at least $p - \text{negl}(\lambda)$. By a standard application of the Splitting Lemma, the probability that $\mathcal{A}$ will not abort in a rewinded execution is at least $p^2/4 - \text{negl}(\lambda)$. Therefore, the probability that $\mathcal{A}$ will complete at least one-out-of N rewinding attempts of $\mathcal{S}$ is at least

$$1 - \big(1 - (p^2/4 - \text{negl}(\lambda))\big)^N \geqslant 1 - e^{-\frac{Np^2}{4}} + \text{negl}(\lambda).$$

Therefore, for $N = 2^{\lambda^{\kappa'}}$, $\mathcal{S}$ runs in $2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda) + 2^t \cdot \text{poly}(\lambda) = O\big(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda)\big)$ steps with at least $1 - e^{-\frac{\lambda^\kappa p^2}{4}} + \text{negl}(\lambda)$ probability. Since, p is non-negligible, this probability is overwhelming.

(ii). The probability $\mathcal{S}$ will abort in $O\big(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda)\big)$ steps is negligible.

By construction, $\mathcal{S}$ will abort either because $(ii.a)$ some fresh challenge hashes to the same value as the one in the first run, or $(ii.b)$ some extracted adversarial vote is invalid, i.e. not in $\mathcal{U}$. By the statement of the theorem, the expected number of steps for a hash collision is $2^{\lambda^\kappa} = \omega(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda))$, so case $(ii.a)$ happens with negligible probability. Moreover, in order for extracted $\mathcal{U}_\ell \notin \mathcal{U}$ for some corrupted voter V_ℓ to happen, the adversary $\mathcal{A}$ must have managed to either break the soundness of the underlying NIZK proof system or 'copy' one of the honest voter's ciphertexts by re-randomizing them. According to Theorems 5.2, 5.4 and Lemma 5.2, both events happen with negligible probability, hence so does case $(ii.b)$.

(End of Claim) $\dashv$

The reduction to the SXDH assumption :

By Claim 5.5.1, we can assume that $\mathcal{S}$ terminates successfully in $O\big(2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda)\big)$ steps, inserting only a negligible error. Using this as a fact, we show that if the lifted ElGamal is IND-CPA secure, then the protocol view created by $\mathcal{S}$ is indistinguishable from the real execution. Note that IND-CPA security of the ElGamal implies that SXDH assumption holds. Given an adversary $\mathcal{A}$ who can distinguish the protocol view simulated by $\mathcal{S}$, we can construct an adversary $\mathcal{B}$ who can break the IND-CPA game. Indeed, in the reduction, when $\mathcal{B}$ receives the public key, we will post it as $h_{i,(w)}$ in the **Setup** protocol, simulating the Dlog Sigma protocol. $\mathcal{B}$ then sends $m_0 = 0, m_1 = 1$ to the IND-CPA challenger. When receiving a ciphertext $c = (c_1, c_2)$, $\mathcal{B}$ can transfer the ciphertext under public key $h_{1,(w)}$ to be a ciphertext under the public key h_1 and use it in the honest voters' ballots. The transformation: $c' = (c_1, c_2 \cdot g_1^{\sum_{i \neq w} \alpha_i})$. Clearly, c and c' encrypts the same message under different public keys. If the adversary $\mathcal{A}$ can distinguish the honest voters' ballots, then the adversary $\mathcal{B}$ distinguish the IND-CPA challenge with running time $2^{\lambda^{\kappa'}} \cdot \text{poly}(\lambda) < 2^{\lambda^{\kappa}}$, for sufficiently large λ.

■

Remark. As in DEMOS-A, we use complexity leveraging to argue privacy which means $k < \lambda$. But for any desired k we can always choose a suitable security parameter λ such that the system is k-private. In most real world elections (e.g., national elections) privacy is only guaranteed between hundreds or a few thousands voters that belong to a precinct. If one wants to achieve privacy nation-wide as well, it is still possible to use our scheme efficiently with the following modification: the trustees, each one individually, will perform a Sigma OR proof that either their published parameter is properly generated or that they know a preimage of a one-way hash function of the coins of the voters (this should be done using a Lapidot-Shamir like proof since the statement is not determined fully before the first move of the protocol). In the privacy proof the simulator can use complexity leveraging to find such preimage in time independent of the number of corrupted voters and thus complete the simulation in time proportional to breaking the one-way function.

5.6 Discussion

DEMOS-2 promises to tackle the scalability limitations of DEMOS-A, respecting the way that the latter has paved, as far as security is concerned. Namely, DEMOS-2 is able to handle large scale elections since EA is relieved from the heavy computational workload, now transferred to the voters' VSDs independent of n. This is achieved while E2E verifiability remains in the standard model (assuming only collision resistance of the hash function), while voter privacy/PCR is proven in the presumably stronger simulation-based setting. Evidence on the efficiency of DEMOS-2 at the voting phase is provided in the paragraph below.

Benchmark results for DEMOS-2 :

A prototype for DEMOS-2 was designed by Bingsheng Zhang in Django framework. Twitter Bootstrap [17] was adopted for better user interface. All the cryptographic elements are Base64 encoded and interchanged in JSON format. The hash function was instantiated as SHA3 and CryptoJS [79] served as its JavaScript implementation. The Type F pairing groups [8] instantiated the asymmetric bilinear groups via jPBC [22] arithmetic on top of SJCL [93] for basic big number arithmetic. The benchmark results in Table. 5.1 show the time on a Mac Mini with 2.5 GHz Intel Core i5, 4GB RAM that a VSD (client) requires to encrypt a vote and produce a ballot.

Table 5.1: Client-side vote encryption benchmarks for DEMOS-2.

m	**Security**	**Version A&B**	**NIZK proof**	**Ballot Size**
2	80 bits	399.4 ms	2239.2 ms	2.5 KB
10	80 bits	1913.5 ms	8210.4 ms	9.3 KB

The aforementioned argumentation does not imply that DEMOS-2 overshadows DEMOS-A in whole. As any client-side e-voting system, DEMOS-2 inherently appears weaknesses that DEMOS-A avoids by its vote-code based nature. A main feature is that voting in DEMOS-A can be run under minimum computational infrastructure, even at the worst case scenario where no VSD is available [3]. Beside, the VSD of every honest voter is responsible for encrypting her vote, hence it must remain honest for privacy. On the other hand, the ballot encoding in the setup phase DEMOS-A, leaks no information to an attacker that just reads a vote-code. Consequently, as long as the working tape of the EA is destroyed after setup, privacy in DEMOS-A is preserved against an all malicious setting during the online voting phase (the adversary corrupts the VC and all the VSDs). In conclusion, DEMOS-A and DEMOS-2 are complementary systems and together form a complete proposal for secure E2E verifiable e-voting in the standard model. The decision on which of the two systems is preferable is related to the concerned election setting.

[3] A notable example is the elections for the President of the New Democracy Greek party, where a system crash at the online voting phase led to an abort of the first election run. This could be avoided in case a vote-code based system like DEMOS-A was used, as voting could survive having voters submitting their votes (vote-codes) in a paper-based manner.

6. HELIOS AS AN E-VOTING CEREMONY

In this chapter, we present a thorough security analysis of the Helios e-voting system [2], focusing on the importance of the human factor as modelled in the ceremony framework introduced in Section 3.6. Helios is a web-based open-audit voting system deployed extensively in the real world, (e.g. the International Association of Cryptologic Research (IACR), the Catholic University of Louvain, and Princeton University) that gives weight to integrity preservation by incorporating a fusion of machine and human-oriented verification mechanisms. For these reasons, Helios is an ideal case of study for the ceremony security model.

We formally describe Helios e-voting ceremony according to the syntax in Section 3.6.3 (cf. Section 6.2). Our description does not reflect the current implemented version of Helios, as it adopts necessary minimum modifications to make Helios secure. For instance, we ensure that each voter is given a *unique identifier* to prevent Helios from the clash attacks introduced in [73]. In addition, we consider a hash function $H(\cdot)$ that all parties have oracle access to, used for committing to election information and ballot generation, as well as the *Fiat-Shamir transformations* [46] in the NIZK proofs that the system requires. As we state below, in the generation of the NIZK proofs for ballot correctness, the unique identifier is included in the hash to prevent replaying attacks presented in [39]. Moreover, we apply strong Fiat-Shamir transformations, where the statement of the NIZK should also be included in the hash. As shown in [15], strong Fiat-Shamir based NIZKs are *simulation sound extractable*, while weak Fiat-Shamir based NIZKs make Helios vulnerable.

For consistency with our framework's syntax, we assign the election preparation to the EA and the vote collection to the VC. However, we stress that in Helios's architecture both these functionalities, as well as posting data in the BB, are entirely controlled by a single administration entity, hence EA and VC are merged at a physical level. This detail leaves room for a critical implementation weakness, in the setting where the trustees are not instructed to verify the correct posting of their partial public keys in the BB. Namely, in the case where no honest trustee performs such verification, then a malicious EA may act as *man-in-the-middle (MitM)* and replace the trustees' partial public keys with ones it adversarially generates, thus resulting to a total break of voters' privacy. To provide a rich description of the human behaviour in Helios, we model trustees by considering the event that the trustee will or will not the verify the correct posting of its partial public key.

Subsequently, we study the security of Helios, modelled as an e-voting ceremony. Our analysis focuses on the integrity aspect w.r.t. to the voting behaviour of the electorate (cf. Section 6.3). In particular, we describe an adversarial strategy against the verifiabiilty of Helios and prove that is effective against a specified class of *assailable* voter transducer distributions. Then, we prove a feasibility of E2E verifiability of Helios, for another class of *resistant* voter transducer definitions. To strenghten our argumentation, we comment on the logical tightness of the two classes.

We illustrate our theoretic results on the integrity of Helios ceremony, by providing an ex-

perimental evaluation from two different sources of human data where people used Helios (cf. Section 6.4) : (i) the member elections of the Board of Directors of the International Association for Cryptographic Research (IACR) and (ii) a non-binding poll among the students of the Department of Informatics and Telecommunications (DI&T) of the University of Athens. We report on the auditing behaviour of the participants as we measured it and we discuss the effects on the level of certainty that can be given in each of the two elections. The message from our evaluation is a negative one: The behaviour profile of people is not such that it can provide sufficient certainty on the correctness of the election result. Given our negative results for actual human data we turn to simulated results for investigating the case when people are supposedly well trained. Even for a voter behaviour distribution with supposedly relatively well trained voters our simulated experiment show that the validity of the election result is sustained with rather low confidence.

Regarding secrecy, we prove that Helios achieves voter privacy/PCR under Definition 3.8 where the EA is honest (cf. Section 6.5). As a conclusion, we extend our approach to a setting where the EA is malicious and describe our aforementioned MitM attack (cf. Section 6.6). We propose trustee auditing as a countermeasure for this attack, thus inserting human behaviour as a crucial parameter also for privacy.

6.1 Building Blocks of Helios

Before proceeding to our analysis, we recall the cryptographic tools that are applied in the construction of Helios. These tools are also building bocks for DEMOS-2, therefore we refer the reader to Section 6.1 for their detalied description

6.1.1 Cryptographic hash function

Helios uses a cryptographic hash function, modelled as a RO (cf. Subsection 2.3.1). As a result Helios's security is proven *in the RO model*.

6.1.2 Schnorr proof of knowledge of a DLOG

Given as parameters the description $\langle \mathbb{G} \rangle$ of some multiplicative cyclic group $\mathbb{G}$ of prime order q and for some statement $x = (g, h) \in \mathbb{G}^2$, the Schnorr protocol prover $\mathcal{P}$ on private input w, convinces the verifier $\mathcal{V}$ of knowledge of $\log_g h = w$. Helios is applying Fiat-Shamir transformation [46], to transformed the Schnorr protocol into a NIZK proof of knowledge of a DLOG in the RO model (cf. Subsection 2.3.4.3).

6.1.3 Chaum-Pedersen proof of DLOG equality

A Chaum-Pedersen (CP) proof [31], is a Σ protocol for proving the equality of discrete logarithms. Given as parameters the description $\langle \mathbb{G} \rangle$ of some multiplicative cyclic group $\mathbb{G}$ of prime order q and for some statement $x = (g_1, h_1, g_2, h_2) \in \mathbb{G}^4$, the prover $\mathcal{P}$ on private input w, convinces the verifier $\mathcal{V}$ of the equality $\log_{g_1} h_1 = \log_{g_2} h_2 = w$. Alternatively, CP protocol can be seen as a proof that $(g, h, \hat{g}, \hat{h})$ forms a DDH tuple. A CP proof can be applied for proving the correct encryption of a standard (lifted) *ElGamal ciphertext* $(C_1, C_2) = (g^t, g^M h^t)$ over the group $\mathbb{G}$, where g is a generator of $\mathbb{G}$, h is the random element in $\mathbb{G}$ used as public key, M is the plaintext and $t \in \mathbb{Z}_q$ is the randomness used in the encryption process.

Even more so, in the case of proving the correct encryption of some vote, the verifier should be unaware of the encrypted plaintext, which is essentially the vote in encoded form. Hence, ballot generation requires proving that a ciphertext is an encryption of some valid encoding M_j of election option opt_j. *without disclosing the actual encoded option*. For this reason, Helios utilises *disjunctive CP proofs* of correct encryption by instantiating the disjunctive proof transformation technique introduced in [40] (cf. Subsection 2.3.4.3) for the case of CP Σ protocol. Finally, as any Σ protocol, CP proofs can be transformed into a NIZK proof of DLOG equality in the RO model by applying Fiat-Shamir transformation [46].

6.1.4 Threshold ElGamal encryption

As in DEMOS-2, Helios applies (k, k)-threshold ElGamal encryption for encrypted ballot generation.

6.2 Syntax of Helios Ceremony

In this section, we present a formal description of Helios ceremony according to the syntax provided in Section 3.6.3. For simplicity, we consider the case of *1-out-of-m elections*, where the set of allowed selections $\mathcal{U}$ is the collection of singletons, $\{\{\text{opt}_1\}, \ldots, \{\text{opt}_m\}\}$, from the set of options $\mathcal{O}$. We begin by defining the transducers that model the human nodes in Helios.

The Helios's transducers :

We define the collections of transducers $\mathcal{M}^V, \mathcal{M}^T, \mathcal{M}^{\text{CD}}$ that reflect the admissible behaviours of voters, trustees and the credential distributor CD respectively.

- The set of admissible voter transducers is denoted by $\mathcal{M}^V := \{M_{i,c,a}\}_{i \in [0,q]}^{c,a \in \{0,1\}}$, where $q \in \mathbb{N}$; The transducer $M_{i,c,a}$ audits the ballot created by the VSD exactly i times (using its ASD) and then submits the $(i+1)$-th ballot created by the VSD; Upon

successful termination, it outputs a individual audit information audit obtained from the VSD; If the termination is not successful and $c = 1$, $M_{i,c,a}$ outputs a special symbol 'Complain' to complain about its failed engagement in the **Cast** ceremony. In any case of termination, when $a = 1$, $M_{i,c,a}$ also outputs a special symbol 'Audit' and sends audit to the ASD. In order to guarantee termination, we limit the maximum number of ballot audits by threshold q.

- The admissible trustee transducers are two and labelled as M_0^T, M_1^T (so that $\mathcal{M}^T = \{M_0^T, M_1^T\}$). At a high level, both M_0^T and M_1^T will utilize the TSD to generate a partial public/secret key pair in the **Setup** ceremony. However, only M_1^T will verify the correct posting of its partial public key in the BB, whereas M_0^T will have no other interaction with the election.
- The CD is required to check the validity of the credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$ generated by the potentially malicious EA before distributing them. In Helios, we define the credential $\mathsf{cr}_i := (\mathsf{ID}_i, t_i)$, where ID_i is a unique voter identity and t_i is an authentication token. The credential distributor first checks for all $i, j \in [n]$: if $i \neq j$ then $\mathsf{ID}_i \neq \mathsf{ID}_j$, and halts if the verification fails. Upon success, it randomly sends each voter V_ℓ a credential though some human channels. Hence, we define the set of CD transducers as $\mathcal{M}^{\mathsf{CD}} := \{M_\sigma^{\mathsf{CD}}\}_{\sigma \in S_n}$, where S_n stands for all possible permutations $[n] \mapsto [n]$.

Remark 6.1 (***Modelling trustees in Helios***)**.** As described in the previous paragraph, we model trustees' behaviour by considering the event that the trustee will or will not the verify the correct posting of its partial public key. This is done so that we capture the Helios's architecture possible privacy vulnerability, in the case where no honest trustee performs such verification. Then, since there there is no way to authenticate the BB data (e.g. PKI support), a malicious EA may act as man-in-the-middle and replace the trustees' partial public keys with ones it adversarially generates, thus resulting to a total break of voters' privacy. Even though such an attack is not considered in our privacy threat model where the EA is assumed honest, providing a rich description of the human behaviour in Helios, sets a robust background for the study of privacy against a malicious EA. The MitM attack against Helios is presented at length in Section 6.6.

We define the Helios ceremony quintuple $\langle$**Setup**, **Cast**, **Tally**, **Result**, **Verify**$\rangle$, using the hash function $H(\cdot)$ as follows:

The **Setup**$(1^\lambda, \mathcal{O}, \mathcal{V}, \mathcal{U}, \mathcal{T})$ *ceremony* :

Each trustee transducer $M_{b_i}^{T_i} \in \{M_0^T, M_1^T\}$, $i = 1, \ldots, k$ sends signal to its TSD. The TSD generates a pair of threshold ElGamal partial keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ and sends pk_i together with a Schnorr (strong Fiat-Shamir) NIZK proof of knowledge (cf. Section 6.1.2) of sk_i to the EA. In addition, the TSD returns a trustee secret $\bar{s}_i := (H(\mathsf{pk}_i), \mathsf{sk}_i)$ to $M_{b_i}^{T_i}$. If there is a proof that EA does not verify, then EA aborts the protocol. Next, EA computes the election public key $\mathsf{pk} = \prod_{i \in [k]} \mathsf{pk}_i$. The public parameters, **info**, which include the election public key pk and the partial public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ as well as their NIZK proofs of knowledge are posted in the BB by the EA.

Subsequently, for $i = 1, \ldots, k$, if $b_i = 1$, then $M_{b_i}^{T_i}$ sends $H(\mathsf{pk}_i)$ to its ASD, and the ASD will fetch **info** from the BB to verify if there exists a partial public key pk_* such that its hash matches $H(\mathsf{pk}_i)$.

Finally, the EA generates the voter credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$, where $\mathsf{cr}_i := (\mathsf{ID}_i, t_i)$, and t_i is a random authentication code. Then, forwards the credentials to the VC and the CD transducer M^{CD}. The CD transducer M_σ^{CD} checks the uniqueness of each ID_i and distributes them to the voter transducers M_{i_ℓ,c_ℓ,a_ℓ} for $\ell \in [n]$, according to the permutation σ over $[n]$ that specifies its behaviour.

The **Cast** *ceremony* :

For each voter V_ℓ, the corresponding transducer M_{i_ℓ,c_ℓ,a_ℓ} has a pre-defined number of i_ℓ ballot auditing steps, where $i_\ell \in [0, q]$. The input of M_{i_ℓ,c_ℓ,a_ℓ} is $(\mathsf{cr}_\ell, \mathcal{U}_\ell)$. For $u \in [i_\ell]$, the following steps are executed:

1. M_{i_ℓ,c_ℓ,a_ℓ} sends $(\mathsf{ID}_\ell, \mathcal{U}_\ell)$ to its VSD, labelled as VSD_ℓ. Let opt_{j_ℓ} be the option selection of V_ℓ, i.e. $\mathcal{U}_\ell = \{\mathsf{opt}_{j_\ell}\}$.

2. For $j = 1, \ldots, m$, VSD_ℓ creates a ciphertext, $C_{\ell,j}$, that is a lifted ElGamal encryption under pk of 1, if $j = j_\ell$ (the selected option position), or 0 otherwise. In addition, it attaches a NIZK proof $\pi_{\ell,j}$ showing that $C_{\ell,j}$ is an encryption of 1 or 0. Finally, an overall NIZK proof π_ℓ is generated, showing that exactly one of these ciphertexts is an encryption of 1. These proofs are strong Fiat-Shamir transformations of disjunctive CP proofs (cf. Section 6.1.3). To generate the proofs, the unique identifier ID_ℓ is included in the hash. The ballot generated is $\psi_{\ell,u} = \langle \psi_{\ell,u}^0, \psi_{\ell,u}^1 \rangle$, where $\psi_{\ell,u}^0 = \big\langle (C_{\ell,1}, \pi_{\ell,1}), \ldots, (C_{\ell,m}, \pi_{\ell,m}), \pi_\ell \big\rangle$ and $\psi_{\ell,u}^1 = H(\psi_{\ell,u}^0)$. The VSD responds to M_{i_ℓ,c_ℓ,a_ℓ} with the ballot $\psi_{\ell,u}$.

3. Then, M_{i_ℓ,c_ℓ,a_ℓ} sends a *Benaloh audit request* to VSD_ℓ. In turn, VSD_ℓ returns the randomness $r_{\ell,u}$ that was used to create the ballot $\psi_{\ell,u}$. The M_{i_ℓ,c_ℓ,a_ℓ} sends $(\mathsf{ID}_\ell, \psi_{\ell,u}, r_{\ell,u})$ to its ASD, which will audit the validity of the ballot. If the verification fails, M_{i_ℓ,c_ℓ,a_ℓ} halts. If the latter happens and $c_\ell = 1$, M_{i_ℓ,c_ℓ,a_ℓ} outputs a special symbol 'Complain', otherwise it returns no output.

After the i_ℓ-th successfully Benaloh audit, M_{i_ℓ,c_ℓ,a_ℓ} invokes VSD_ℓ to produce a new ballot ψ_ℓ as described in step 2 above; however, upon receiving ψ_ℓ, M_{i_ℓ,c_ℓ,a_ℓ} now sends cr_ℓ to VSD_ℓ, indicating it to submit the ballot to the VC. The M_{i_ℓ,c_ℓ,a_ℓ} then outputs $\mathsf{audit}_\ell := (\mathsf{ID}_\ell, \psi_\ell^1)$. If $a_\ell = 1$, M_{i_ℓ,c_ℓ,a_ℓ} also outputs a special symbol 'Audit' which indicates that it will send audit_ℓ to ASD_ℓ which will audit the BB afterwards, as specified in the **Verify** algorithm below.

When VC receives a cast vote $(\mathsf{cr}_\ell, \psi_\ell)$ from VSD_ℓ, it checks the validity of the credential cr_ℓ and that ψ_ℓ is a well-formed ballot by verifying the NIZK proofs. If the check fails, then it aborts the protocol. After voting ends, VC updates its state with the pairs $\{(\psi_\ell, \mathsf{ID}_\ell)\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}}$ of cast votes and the associated identifiers, where $\mathcal{V}_{\mathsf{succ}}$ is the set of voters that voted successfully.

The **Tally** *ceremony* :

In the **Tally** ceremony, VC sends $\{\psi_\ell\}_{V_\ell \in \mathcal{V}_{\text{succ}}}$ to all trustee transducers $M_{b_i}^{T_i}$'s TSD, $i = 1, \ldots, k$. Next, the TSD of each $M_{b_i}^{T_i}$, $i = 1, \ldots, k$, performs the following computation: it constructs the product ciphertext $\mathbf{C}_j = \prod_{V_\ell \in \mathcal{V}_{\text{succ}}} C_{\ell,j}$ for $j = 1, \ldots, m$. By the additive homomorphic property of (lifted) ElGamal, each $\mathbf{C}_j$ is a valid encryption of the number of votes that the option opt_j received. Then, the TSD uses sk_i to produce the partial decryption of all C_j, denoted by x_j^i, and sends it to the VC along with NIZK proofs of correct partial decryption. The latter are Fiat-Shamir transformations of CP proofs. If there is a proof that VC does not verify, then it aborts the protocol. After all trustees finish their computation, EA updates τ with $\{(x_1^i, \ldots, x_m^i)\}_{i \in [k]}$ and the NIZK proofs.

The **Result**(τ) *algorithm* :

For each option opt_j, the **Result** algorithm computes the number of votes, x_j, that opt_j has received using the partial decryptions $x_j^1, \ldots, x_j^k$. The output of the algorithm is the vector $\langle x_1, \ldots, x_m \rangle$.

The **Verify**$(\tau, \text{audit}_\ell)$ *algorithm* :

The algorithm **Verify**$(\tau, \text{audit}_\ell)$ outputs 1 if the following conditions hold:

1. The structure of τ and all election information is correct (using **info**).
2. There exists a ballot in τ, indexed by ID_ℓ, that contains the hash value ψ_ℓ^1.
3. The NIZK proofs for the correctness of all ballots in τ verify.
4. The NIZK proofs for the correctness of all trustees' partial decryptions verify.
5. For $j = 1, \ldots, m$, x_j is a decryption of $\mathbf{C}'_j$, where $\mathbf{C}'_j$ is the homomorphic ciphertext created by multiplying the respective ciphertexts in the ballots published on the BB (in an honest execution, $\mathbf{C}'_j$ should be equal to $\mathbf{C}_j$).

6.3 E2E Verifiability of Helios e-Voting Ceremony

In a Helios e-voting ceremony, an auditor can check the correct construction of the ballots and the valid decryption of the homomorphic tally by verifying the NIZK proofs. In our analysis, it is sufficient to require that all NIZK proofs have negligible soundness error $\epsilon(\cdot)$ in the RO model. Note that in Section 6.2, we explicitly modify Helios to associate ballots with the voters' identities, otherwise a clash attack [73] would break verifiability. For simplicity in presentation, we assume that the identifiers are created by the adversary, i.e. the set $\{\text{ID}_\ell\}_{\ell \in [n]}$ matches the set of voters $\mathcal{V}$.

Throughout our analysis, we assume the honesty of the CD and thus the distribution of the credentials is considered to be an arbitrary permutation over $[n]$. Since there are only

two admissible trustee transducers M_0^T, M_1^T, the distribution of trustee transducers $\mathbf{D}_p^T$ is set as the p-biased coin-flip below:

$$\Pr_{\mathbf{D}_p^T}[M] = \begin{cases} p, & \text{if } M = M_1^T \\ 1-p, & \text{if } M = M_0^T \end{cases} \tag{6.1}$$

Moreover, in the **Cast** ceremony, the ballots and individual audit information are produced before the voters show their credentials to the system. Since the CD is honest, the adversary is oblivious the the maps between the credentials to the voter transducers. The credentials are only required when the voters want to submit their ballots, hence, according to the discussion in Remark 3.4, we will consider only a universal voter transducer distribution **D** in the case study of Helios. Namely, $\mathbf{D}_1 = \cdots = \mathbf{D}_n = \mathbf{D}$.

6.3.1 Attacks on verifiability

As mentioned in the introduction of this section, we have modified Helios to prevent the system from clash attacks [73]. For simplicity, we exclude all the trivial attacks that the adversary may follow, i.e. the ones that will be detected with certainty (e.g. malformed or unreadable voting interface and public information). Therefore, the meaningful types of attack that an adversary may launch are the following:

- *Collision attack*: the adversary computes two votes which hash to the same value. The collision resistance of the hash function $H(\cdot)$, prevents from these attacks except from some negligible probability ϵ' [1].
- *Invalid vote attack*: the adversary creates a vote for some invalid plaintext, i.e. a vector that does not encode a candidate selection (e.g., multiple votes for some specific candidate). This attack can be prevented by the soundness of the NIZK proofs, except from the negligible soundness error ϵ. The NIZK verification is done via the voter's ASD.
- *VSD attack*: the adversary creates a vote which is valid, but corresponds to different selection than the one that the voter intended. A Benaloh audit at the **Cast** ceremony step can detect such an attack with certainty, as the randomness provided by the VSD perfectly binds the plaintext with the audited ElGamal ciphertext.
- BB *attack:* the adversary deletes/inserts an honest vote from/to the BB, or replaces it with some other vote of its choice, after voting has ended. Assuming no hash collisions, any such modification will be detected if the voter chooses to audit the BB via her ASD.
- *Invalid tally decryption attack:* the adversary provides a decryption which is not the plaintext that the homomorphic tally vector encrypts. The NIZK proofs of correct decryption prevent this attack, except for a negligible soundness error ϵ.

[1] This requires that $H(\cdot)$ has resistance to second preimage attacks.

Remark 6.2 (***Completeness of the attack list***)**.** It can be easily shown that the above list exhausts all possible attack strategies against Helios in our threat model. Namely, in an environment with no clash, collision and invalid encryption attacks, the set of votes is in the correct (yet unknown) one-to-one correspondence with the set of voters, and all votes reflect a valid candidate selection of the unique corresponding voter. As a result, a suitably designed vote extractor will decrypt (in super-polynomial time) and output the actual votes from the non-honest-and-successful voters, up to permutation. Consequently, if no honest vote has been modified during and after voting, and the homomorphic tally of the votes is correctly computed and decrypted, then the perfect binding of the plaintexts and ciphertexts of ElGamal implies that the decryption of the tally is the *intended election result*.

6.3.2 Attacking the verifiability of Helios e-voting ceremony

As explained in the previous subsection, any attempt of collision, invalid vote and invalid tally decryption attacks has negligible probability of success for the adversary due to the collision resistance of the hash function and the soundness of the ZK proofs. Therefore, in a setting where no clash attacks are possible, the adversary's chances to break verifiability rely on combinations of VSD and BB attacks. The probability of these attacks being detected depends on the voter transducer distribution **D** which depicts their auditing behaviour during and after voting. In the following theorem, we prove that the verifiability of Helios is susceptible to VSD or/and BB attacks, when the voters sample from a class of assailable voter transducer distributions.

Theorem 6.1 (***Vulnerability of Helios ceremony***)**.** *Assume an election run of Helios with n voters, m candidates and k trustees. Let $q, \delta, \theta, \phi \in$, where $0 < \theta, \phi \leqslant n$ and q is the maximum number of Benaloh audits. Let* **D** *be a (universal) voter transducer distribution s.t. for some $\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2 \in [0, 1)$ at least one of the two following conditions holds:*

(i). There is an $i^* \in \{0, \ldots, q\}$ that determines "vulnerable VSD auditing behaviour". Namely, (i.a) the probability that a voter executes at least i^* Benaloh audits is $1 - \kappa_1$ AND (i.b) the probability that a voter, given that she has executed at least i^* Benaloh audits, will cast her vote after exactly i^* Benaloh audits is $1 - \kappa_2$ AND (i.c) the probability that a voter, given that she will execute exactly i^* Benaloh audits, will not complain in case of unsuccessful audit is κ_3.

(ii). There is a subset $\mathcal{J}^* \subseteq \{0, \ldots, q\}$ that determines "vulnerable BB auditing behaviour". Namely, (ii.a) the probability that a voter executes j Benaloh audits for some $j \in \mathcal{J}^*$ is $1 - \mu_1$ AND (ii.b) for every $j \in \mathcal{J}^*$, the probability that a voter, given she has executed j Benaloh audits, will not audit the BB is at least $1 - \mu_2$.

Let $\mathcal{D} = \langle \mathbf{D}, \ldots, \mathbf{D}, \mathbf{D}^{T_1}, \ldots, \mathbf{D}^{T_k}, \mathbf{D}^{\mathsf{CD}} \rangle$ be a transducer distribution vector where $\mathbf{D}^{T_i} = \mathbf{D}^{T}_{p_i}$, $i = 1, \ldots, k$, is the p_i-biased coin-flip trustee transducer distribution in Eq. (6.1) *for arbitrary $p_i \in [0, 1]$ and $\mathbf{D}^{\mathsf{CD}}$ is an arbitrary* CD *transducer distribution. Then, there is a*

PPT adversary $\mathcal{A}$ that wins the E2E verifiability ceremony game $G_{\text{E2E}}^{\mathcal{A},\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda,n,m,k)$ in Figure 3.7 for any vote extractor $\mathcal{E}$, any $\Delta \in [0,1)$ as follows:

- *under condition (i), provided the parameters d,θ,ϕ satisfy:*

$$\begin{aligned}\delta &\leqslant \big(1-\Delta\big)^2(1-\kappa_2)(1-\kappa_1)n\\ \theta &\leqslant n-(1+\Delta)(\kappa_2+\Delta-\Delta\kappa_2)(1-\kappa_1)n\\ \phi &\geqslant (1+\Delta)^2\kappa_3(\kappa_2+\Delta-\Delta\kappa_2)(1-\kappa_1)n\end{aligned}$$

  with probability of success at least $\boxed{1-5e^{-\kappa_3\beta_2\beta_1\frac{\Delta^2}{3}}}$

  where $\beta_1=(1-\Delta)(1-\kappa_1)n$ *and* $\beta_2=(\kappa_2-\Delta+\Delta\kappa_2)(1-\kappa_2)$.

- *under condition (ii), provided the parameter δ satisfies $\delta \leqslant (1-\Delta)(1-\mu_1)n$*

  with probability of success at least $\boxed{(1-e^{-(1-\mu_1)n\frac{\Delta^2}{2}})(1-\mu_2)^\delta}$.

Proof. We observe that when an adversary makes no voter corruptions, then the set $\mathcal{V}\backslash\mathcal{V}_{\text{succ}}$ contains only honest voters that did not complete the **Cast** ceremony successfully. Therefore, the election result w.r.t. $\mathcal{V}\backslash\mathcal{V}_{\text{succ}}$ is zero, so in our analysis we can fix the trivial vote extractor $\mathcal{E}$ that outputs the zero vector of length $|\mathcal{V}\backslash\mathcal{V}_{\text{succ}}|$. By definition, if the adversary breaks the E2E verifiability game for $\mathcal{E}$, then it does so for any other vote extractor.

We denote by $E_{i,c,a}$ the event that the honest voter engages in the **Cast** ceremony by running the transducer $M_{i,c,a}$. We study the following two cases:

Case 1. *Condition (i) holds [Breaking verifiabiliy via VSD attacks].* We describe a PPT adversary $\mathcal{A}_1$ against verifiabiilty as follows: $\mathcal{A}_1$ corrupts no voters and observes the number of Benaloh audits that each voter performs. If the voter has executed i^* Benaloh audits, then $\mathcal{A}_1$ performs a VSD attack on the i^*+1-th ballot that the voter requests.

By condition (i.a), the probablity that the voter will perform at least i^* Benaloh audits is

$$\Pr_{\boldsymbol{\mathcal{D}}}\Big[\neg\big(\bigvee_{\substack{0\leqslant i<i^*\\ c,a\in\{0,1\}}} E_{i,c,a}\big)\Big]=1-\kappa_1\ .$$

Let T be the number of VSD attacks that $\mathcal{A}_1$ executes. It is easy to see that T follows the binomial distribution $B(n,1-\kappa_1)$. Therefore, by the Chernoff bounds we have that for any $\Delta\in[0,1)$,

$$\begin{aligned}&\Pr_{\boldsymbol{\mathcal{D}}}[(1-\Delta)(1-\kappa_1)n<T<(1+\Delta)(1-\kappa_1)n]\geqslant\\ &\qquad\geqslant 1-e^{-(1-\kappa_1)n\Delta^2/2}-e^{-(1-\kappa_1)n\frac{\Delta^2}{\min\{2+\Delta,3\}}}\geqslant 1-2e^{-(1-\kappa_1)n\frac{\Delta^2}{3}}.\end{aligned}\tag{6.2}$$

Let X_T be the number of successful VSD attacks out of all the T attempts. Observe that each successful single VSD attack adds 1 to the total tally deviation (the ballot encrypts a candidate vector that is different from the voter's intented selection). Hence, $\mathcal{A}_1$ achieves tally deviation exactly X_T. From condition (i.b), the probablity that a voter, given that she has executed at least i^* Benaloh audits, will execute exactly i^* Benaloh audits is

$$\Pr_{\mathcal{D}}\Big[\bigvee_{c,a\in\{0,1\}} E_{i^*,c,a}\Big|\neg\big(\bigvee_{\substack{0\leqslant i<i^*\\ c,a\in\{0,1\}}} E_{i,c,a}\big)\Big] = 1-\kappa_2\ .$$

By definition, X_T follows the binomial distribution $B(T, 1-\kappa_2)$. Thus, by the Chernoff bounds, we have that for any $\Delta \in [0,1)$,

$$\begin{aligned}&\Pr_{\mathcal{D}}[(1-\Delta)(1-\kappa_2)T < X_T < (1+\Delta)(1-\kappa_2)T] \geqslant \\ &\quad\geqslant 1 - e^{-(1-\kappa_2)T\frac{\Delta^2}{2}} - e^{-(1-\kappa_2)T\frac{\Delta^2}{\min\{2+\Delta,3\}}} \geqslant 1 - 2e^{-(1-\kappa_2)T\frac{\Delta^2}{3}}.\end{aligned} \tag{6.3}$$

According to the description of $\mathcal{A}_1$, the number of honest voters that will not complete the **Cast** ceremony successfully is $T - X_T \geqslant 0$. Therefore, the number of successful honest voters is $|\mathcal{V}_{\mathsf{succ}}| = n - (T - X_T)$. In addition, by condition (i.c), the number of complaining voters $|\mathcal{V}_{\mathsf{comp}}|$ follows the binomial distribution $B(T - X_T, \kappa_3)$. Hence, by the Chernoff bounds, we have that for any $\Delta \in [0,1)$,

$$\Pr_{\mathcal{D}}[|\mathcal{V}_{\mathsf{comp}}| < (1+\Delta)\kappa_3(T-X_T)] \geqslant 1 - e^{-\kappa_3(T-X_T)\frac{\Delta^2}{3}}. \tag{6.4}$$

By description, $\mathcal{A}_1$ will definitely win the game $G_{\mathsf{E2E}}^{\mathcal{A}_1,\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda, n, m, k)$ when

$$\big(X_T \geqslant \delta\big) \wedge \big(n - (T - X_T) \geqslant \theta\big) \wedge \big(|\mathcal{V}_{\mathsf{comp}}| \leqslant \phi\big).$$

Based on the above observation, we provide a lower bound on the probability that $\mathcal{A}_1$ wins the E2E verifiabiilty game $G_{\mathsf{E2E}}^{\mathcal{A}_1,\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda, n, m, k)$ when the parameters δ, θ, ϕ satisfy the following constraints:

$$\delta \leqslant \big(1-\Delta\big)^2(1-\kappa_1)(1-\kappa_2)n \tag{6.5a}$$
$$\theta \leqslant n - (1+\Delta)(\kappa_2 + \Delta - \Delta\kappa_2)(1-\kappa_1)n \tag{6.5b}$$
$$\phi \geqslant (1+\Delta)^2\kappa_3(\kappa_2 + \Delta - \Delta\kappa_2)(1-\kappa_1)n \tag{6.5c}$$

By Eq. (6.2),(6.3) and (6.4), we have that for any $\Delta \in [0,1)$,

$$
\begin{aligned}
&\Pr_{\mathcal{D}}[G_{\mathsf{E2E}}^{\mathcal{A}_1,\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda,n,m,k)=1] \geqslant \\
&\geqslant \Pr_{\mathcal{D}}\Big[\big(X_T \geqslant (1-\Delta)^2(1-\kappa_2)(1-\kappa_1)n\big) \wedge \\
&\quad \wedge \big(|\mathcal{V}_{\mathsf{comp}}| \leqslant (1+\Delta)^2\kappa_3(\kappa_2+\Delta-\Delta\kappa_2)((1-\kappa_1)n\big) \wedge \\
&\quad \wedge \big(T-X_T) \leqslant (\kappa_2+\Delta-\Delta\kappa_2)(1+\Delta)(1-\kappa_1)n\Big] \geqslant \\
&\geqslant \Pr_{\mathcal{D}}\Big[\big(|\mathcal{V}_{\mathsf{comp}}| \leqslant (1+\Delta)\kappa_3(T-X_T)\big) \wedge \\
&\quad \wedge \big((1-\Delta)(1-\kappa_2)T < X_T < (1+\Delta)(1-\kappa_2)T\big) \wedge \\
&\quad \wedge \big((1-\Delta)(1-\kappa_1)n < T < (1+\Delta)(1-\kappa_1)n\big)\Big] = \\
&= \Pr_{\mathcal{D}}\Big[(1-\Delta)(1-\kappa_1)n < T < (1+\Delta)(1-\kappa_1)n] \cdot \\
&\quad \cdot \Pr_{\mathcal{D}}\Big[(1-\Delta)(1-\kappa_2)T < X_T < (1+\Delta)(1-\kappa_2)T\Big| \\
&\qquad \Big|(1-\Delta)(1-\kappa_1)n < T < (1+\Delta)(1-\kappa_1)n\Big] \cdot \\
&\quad \cdot \Pr_{\mathcal{D}}\Big[\big(|\mathcal{V}_{\mathsf{comp}}| \leqslant (1+\Delta)\kappa_3(T-X_T)\big)\Big| \\
&\qquad \Big|\big((1-\Delta)(1-\kappa_2)T < X_T < (1+\Delta)(1-\kappa_2)T\big) \wedge \\
&\qquad\quad \wedge \big((1-\Delta)(1-\kappa_1)n < T < (1+\Delta)(1-\kappa_1)n\big)\Big] \geqslant \\
&\geqslant \Big(1-2e^{-(1-\kappa_1)n\frac{\Delta^2}{3}}\Big) \cdot \Big(1-2e^{-(1-\kappa_2)[(1-\Delta)(1-\kappa_1)n]\frac{\Delta^2}{3}}\Big) \cdot \\
&\quad \cdot \Big(1-e^{-\kappa_3([1-(1+\Delta)(1-\kappa_2)]\cdot[(1-\Delta)(1-\kappa_1)n])\frac{\Delta^2}{\min\{2+\Delta,3\}}}\Big) \geqslant \\
&\geqslant 1-5e^{-\kappa_3(\kappa_2-\Delta+\Delta\kappa_2)(1-\kappa_2)(1-\Delta)(1-\kappa_1)n\frac{\Delta^2}{3}} = 1-5e^{-\kappa_3\beta_2\beta_1\frac{\Delta^2}{3}},
\end{aligned}
\tag{6.6}
$$

where $\beta_1 = (1-\Delta)(1-\kappa_1)n$ and $\beta_2 = (\kappa_2-\Delta+\Delta\kappa_2)(1-\kappa_2)$.

Case 2. *Condition (ii) holds [Breaking verifiabiliy via* BB *attacks].* We describe a PPT adversary $\mathcal{A}_2$ against verifiabiilty as follows: $\mathcal{A}_2$ makes no corruptions and keeps record of the voters that perform j Benaloh audits for some $j \in \mathcal{J}^*$. Let $\mathcal{V}_{\mathcal{J}^*}$ be the set of those voters. After all **Cast** ceremonies have been completed, every voter has terminated successfully, i.e. $\mathcal{V}_{\mathsf{succ}} = \mathcal{V}$ and $\mathcal{V}_{\mathsf{comp}} = \varnothing$. In order to achieve tally deviation δ, $\mathcal{A}_2$ performs a BB attack on the votes of an arbitrary subset of d voters in $\mathcal{V}_{\mathcal{J}^*}$. As in the previous case, each single BB attack adds 1 to the total tally deviation, so $|\mathcal{V}_{\mathcal{J}^*}| \geqslant \delta$ must hold. By condition (ii.a), the probability $\Pr_{\mathcal{D}}\Big[\bigvee_{\substack{j\in\mathcal{J}^* \\ c,a\in\{0,1\}}} E_{j,c,a}\big)\Big]$ that a voter is in $\mathcal{V}_{\mathcal{J}^*}$ is $1-\mu_1$. By definition, $|\mathcal{V}_{\mathcal{J}}|$ follows the binomial distribution $B(n,1-\mu_1)$. Thus, by the Chernoff bound and for any $\Delta \in [0,1)$,

$$
\Pr_{\mathcal{D}}[|\mathcal{V}_{\mathcal{J}^*}| > (1-\Delta)(1-\mu_1)n] \geqslant 1-e^{(1-\mu_1)n\frac{\Delta^2}{2}}. \tag{6.7}
$$

However, $\mathcal{A}_2$ will be successful iff all d voters in the selected subset of $\mathcal{V}_j$ do not audit the BB. By condition (ii.b) and the independency of the voter transducers' sampling, this happens with probability at least $(1-\mu_2)^\delta$. Therefore by Eq. (6.7), we have that for $\delta \leqslant (1-\Delta)(1-\mu_1)n$ and any θ, ϕ, it holds that

$$\begin{aligned}
&\Pr_{\mathcal{D}}[G_{\mathsf{E2E}}^{\mathcal{A}_2,\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda,n,m,k)=1] = \\
&\quad = \Pr_{\mathcal{D}}\Big[\big(G_{\mathsf{E2E}}^{\mathcal{A}_2,\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda,n,m,k)=1\big) \wedge \big(|\mathcal{V}_{\mathcal{J}}| \geqslant (1-\Delta)(1-\mu_1)n\big)\Big] = \\
&\quad = \Pr_{\mathcal{D}}\Big[\big(G_{\mathsf{E2E}}^{\mathcal{A}_2,\mathcal{E},\boldsymbol{\mathcal{D}},\delta,\theta,\phi}(1^\lambda,n,m,k)=1\big) \wedge \big(|\mathcal{V}_{\mathcal{J}}| \geqslant (1-\Delta)(1-\mu_1)n\big)\Big] \geqslant \\
&\quad \geqslant (1-e^{-(1-\mu_1)n\frac{\Delta^2}{2}})(1-\mu_2)^\delta.
\end{aligned} \tag{6.8}$$

Hence, by the lower bounds provided in Eq. (6.6),(6.8) and for $\delta \leqslant (1-\Delta)(1-\mu_1)n$, we get the complete proof of the theorem. ■

6.3.3 End-to-end verifiability theorem Helios e-voting ceremony

In this subsection, we prove the E2E verifiability of Helios e-voting ceremony in the RO model, when the voter transducer distribution satisfies two conditions. As we will explain at length in the next subsection, these conditions are logically complementary to the ones stated in Theorem 6.1, as long as the complaining behaviour of the voters is balanced (i.e. the voters have 1/2 probability of complaining in case of unsuccessful termination).

Theorem 6.2 (***Verifiability of Helios ceremony***)**.** *Assume an election run of Helios with n voters, m candidates and k trustees. Assume that the hash function $H(\cdot)$ considered in Section 6.2 is a random oracle. Let $q, \delta, \theta, \phi \in$, where $0 < \theta, \phi \leqslant n$ and q is the maximum number of Benaloh audits. Let* **D** *be a (universal) transducer distribution and some $\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2 \in [0,1)$ s.t. the two following conditions hold:*

(i) There is an $i^* \in \{0,\ldots,q+1\}$ that guarantees "resistance against VSD attacks". Namely, (i.a) the probability that a voter executes at least i^* Benaloh audits is κ_1 and (i.b) for every $i \in \{0,\ldots,q\}$, if $i < i^*$, then the probability that a voter, given that she will execute at least i Benaloh audits, will cast her vote after exactly i Benaloh audits, is no more than κ_2 AND the probability that a voter, given that she will execute exactly i Benaloh audits, will complain in case of unsuccessful audit is at least $1-\kappa_3$.

(ii) There is a subset $\mathcal{J}^* \subseteq \{0,\ldots,q\}$ that guarantees "resistance against BB attacks". Namely, (ii.a) the probability that a voter executes j Benaloh audits for some $j \in \mathcal{J}^*$ is $1-\mu_1$ AND (ii.b) for every $j \in \mathcal{J}^*$, the probability that a voter, given she has executed j Benaloh audits, will audit the BB is at least $1-\mu_2$.

Let $\mathcal{D} = \big\langle \mathbf{D},\ldots,\mathbf{D},\mathbf{D}^{T_1},\ldots,\mathbf{D}^{T_k},\mathbf{D}^{\mathsf{CD}}\big\rangle$ be a transducer distribution vector where $\mathbf{D}^{T_i} = \mathbf{D}^T_{p_i}$, $i=1,\ldots,k$, is the p_i-biased coin-flip trustee transducer distribution in Eq. (6.1) *for arbitrary*

$p_i \in [0,1]$ *and* $\mathbf{D}^{\mathsf{CD}}$ *is an arbitrary* CD *transducer distribution. Then, for any* $\Delta \in [0,1)$ *for any* δ, θ*, and under the constraint*

$$\phi \leqslant (1-\Delta)(1-\kappa_3)\Big(\frac{1}{(1+\Delta)\kappa_2} - 1\Big)\Big(\frac{\delta}{2} - (1+\Delta)\kappa_1 n\Big) \ ,$$

the Helios e-voting ceremony achieves E2E verifiability for $\mathcal{D}$*, a number of* θ *honest successful voters, a number of* ϕ *honest complaining voters and tally deviation* δ *with error*

$$e^{-\min\left\{\kappa_1 n\frac{\Delta^2}{3}\ ,\ \mu_1 n\frac{\Delta^2}{3}\ ,\ \gamma(\frac{\delta}{2}-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{3}\ ,\ \ln\left(\frac{1}{\mu_2}\right)(\frac{\delta}{2}-(1+\Delta)\mu_1 n)\right\}} + \\ + (\mu_1+\mu_2-\mu_1\mu_2)^{\theta} + \mathsf{negl}(\lambda) \ ,$$

where $\gamma = \min\left\{\kappa_2\ ,\ \frac{3}{2}(1-\kappa_3)\big(\frac{1}{(1+\Delta)\kappa_2}-1\big)\right\}$.

Proof. W.l.o.g., we assume that no trivial attacks are executed. Therefore the adversary's strategy comprises a combination of the attacks listed in Subsection 6.3.1. At first, we construct the vote extractor $\mathcal{E}$ as shown below:

Construction of the vote extractor for Helios :

The vote extractor $\mathcal{E}$ for Helios receives as input τ and the set of receipts (list of IDs paired with hashes) $\{\mathsf{audit}_\ell\}_{\mathcal{V}_{\mathsf{succ}}}$. Then, $\mathcal{E}$ on input $\big(\tau, \{\mathsf{audit}_\ell\}_{\mathcal{V}_{\mathsf{succ}}}\big)$ executes the following steps:

The vote extractor $\mathcal{E}\big(\tau, \{\mathsf{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}}\big)$ for Helios

1. If the result is not meaningful (i.e., **Result**$(\tau) = \perp$), then $\mathcal{E}$ outputs $\perp$. Otherwise, $\mathcal{E}$ arbitrarily arranges the voters in $\mathcal{V}\backslash\mathcal{V}_{\mathsf{succ}}$ as $\langle V_\ell^{\mathcal{E}}\rangle_{n-|\mathcal{V}_{\mathsf{succ}}|}$.
2. For every $\ell \in [n-|\mathcal{V}_{\mathsf{succ}}|]$:
   (a) $\mathcal{E}$ reads the vote list in τ. It locates the first vote, denoted by $\psi_\ell^{\mathcal{E}}$, which neither includes a hash appearing in $\{\mathsf{audit}_\ell\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}}$, nor is associated with some voter in $\mathcal{V}\backslash\mathcal{V}_{\mathsf{succ}}$, and associates this vote with $V_\ell^{\mathcal{E}}$. If no such vote exists, then $\mathcal{E}$ sets $\mathcal{U}_\ell^{\mathcal{E}} = \varnothing$ (encoded as the zero vector).
   (b) $\mathcal{E}$ decrypts the ciphertexts in $\psi_\ell^{\mathcal{E}}$ (in superpolynomial time). If the decrypted messages form a vector in $\{0,1\}^m$ that has 1 in a single position, j_ℓ , then it sets $\mathcal{U}_\ell^{\mathcal{E}} = \{\mathsf{opt}j_\ell\}$. Otherwise, it outputs $\perp$.
3. Finally, $\mathcal{E}$ outputs $\langle \mathcal{U}_\ell^{\mathcal{E}}\rangle_{V_\ell^{\mathcal{E}} \in \mathcal{V}\backslash\mathcal{V}_{\mathsf{succ}}}$.

Assume a PPT adversary $\mathcal{A}$ that wins the game $G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\mathcal{D},\delta,\theta,\phi}(1^\lambda, n, m, k)$, for the above vote extractor $\mathcal{E}$. We denote by i_ℓ the number of Benaloh audits that the honest voter V_ℓ executes. We denote by $E_{i,c,a}$ the event that the voter engages in the **Cast** ceremony by running the transducer $M_{i,c,a}$.

Let A be the event that at least one honest voter will audit the BB after the end of the election, i.e. $\mathcal{V}_{\mathsf{audit}} \neq \varnothing$. By condition *(ii)*, the probability that $V_\ell \notin \mathcal{V}_{\mathsf{audit}}$ is bounded by

$$\begin{aligned}
&\Pr_{\mathcal{D}}[V_\ell \notin \mathcal{V}_{\mathsf{audit}}] = \Pr_{\mathcal{D}}[E_{i_\ell,0,0} \vee E_{i_\ell,1,0}] = \\
&\quad = \Pr_{\mathcal{D}}[(E_{i_\ell,0,0} \vee E_{i_\ell,1,0}) \wedge i_\ell \in \mathcal{J}^*] + \Pr_{\mathcal{D}}[(E_{i_\ell,0,0} \vee E_{i_\ell,1,0}) \wedge i_\ell \notin \mathcal{J}^*] \leqslant \\
&\quad \leqslant \Pr_{\mathcal{D}}[i_\ell \in \mathcal{J}^*] + (1 - \Pr_{\mathcal{D}}[i_\ell \in \mathcal{J}^*]) \cdot \Pr_{\mathcal{D}}[E_{i_\ell,0,0} \vee E_{i_\ell,1,0} \mid i_\ell \notin \mathcal{J}^*] \leqslant \\
&\quad \leqslant \mu_1 + (1-\mu_1)\mu_2 = \mu_1 + \mu_2 - \mu_1\mu_2.
\end{aligned} \tag{6.9}$$

Therefore, by Eq. (6.9), the independence of the transducers' sampling and the fact that there are at least θ honest (and successful) voters, we have that

$$\Pr_{\mathcal{D}}[\neg A] = \Pr_{\mathcal{D}}\Big[\bigwedge_{V_\ell \in \mathcal{V}_{\mathsf{succ}}} (V_\ell \notin \mathcal{V}_{\mathsf{audit}})\Big] \leqslant (\mu_1 + \mu_2 - \mu_1\mu_2)^\theta. \tag{6.10}$$

Let F be the event that $\mathcal{A}$ has performed at least one invalid vote or tally decryption attack. Namely, one of the homomorphic tally ciphertexts $\mathbf{C}_j$, for $j \in [m]$, does not decrypt as x_j, or a ballot of a voter $V_\ell \in \mathcal{V}$ does not correspond to an encryption of a vector in $\{0,1\}^m$ that has 1 in a single position. Assuming that $H(\cdot)$ is a RO, all the NIZK proofs are sound except from a negligible error ϵ. If $\mathcal{V}_{\mathsf{audit}} \neq \varnothing$, there is at least one honest voter who verifies the ZK proofs. Hence, it holds that

$$\Pr_{\mathcal{D}}[\big(G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\mathcal{D},\delta,\theta,\phi}(1^\lambda, n, m, k) = 1\big) \wedge F \mid A] \leqslant \epsilon(\lambda) = \mathsf{negl}(\lambda)\,. \tag{6.11}$$

Suppose that F does not occur. In this case, $\mathcal{E}$ outputs a vector of selections that is a permutation of the adversarial votes and some zero vectors, thus it homomorphically sums to the actual adversarial result. Therefore, $\mathcal{A}$ deviates from the intended result $f(\langle \mathcal{U}_1, \ldots, \mathcal{U}_n \rangle)$ only because it

(i). alters some votes of the voters in $\mathcal{V}_{\mathsf{succ}}$ during voting, or

(ii). replaces, deletes or inserts some of the votes of the (successful or unsuccessful) honest voters in τ (BB).

By Remark 6.2, $\mathcal{A}$ achieves this by performing combinations of collision, VSD and BB attacks. As mentioned in Subsection 6.3.1, the probability of a successful collision attack ($\mathcal{A}$ provides V_ℓ with some individual audit information audit_ℓ that has the same hash value as a another ballot of $\mathcal{A}$'s choice) is no more than a negligible function $\epsilon'(\lambda)$.

We denote by X the set of the honest voters whose votes have been altered during voting (VSD attack) and by Y the set of honest voters whose votes have been replaced/deleted/-inserted in the BB, both determined by $\mathcal{A}$'s adaptive strategy. Each of these attacks adds 1 to the total deviation, so the deviation that $\mathcal{A}$ achieves is $|X \cup Y| = |X \backslash Y| + |Y| \geqslant \delta$.

W.l.o.g., we assume that X and Y are disjoint as any vote under VSD and BB attack only lowers the probability of success of $\mathcal{A}$, while adding no more than 1 to the total tally

deviation. In addition, we assume that $|X|+|Y| = \delta$, as any strategy of $\mathcal{A}$ s.t. $|X|+|Y| > \delta$ has success probablity which is upper bounded by the one of a strategy for some VSD and BB attack sets $X' \subseteq X$ and $Y' \subseteq Y$ s.t.$|X'| + |Y'| = \delta$. We provide upper bounds on the success probability of $\mathcal{A}$ w.r.t. to each of the subsets X and Y for the case they become larger than $\delta/2$. Clearly, either $|X| \geqslant \delta/2$ or $|Y| \geqslant \delta/2$ must hold

Bounding $\mathcal{A}$'s success probability w.r.t. X, when $|X| \geqslant \delta/2$:

Let T the set of voters that $\mathcal{A}$ attempted a VSD attack. We partition T, X into the following sets:

$$T^- = \{V_\ell \in T | i_\ell < i^*\} \quad \text{and} \quad T^+ = \{V_\ell \in T | i_\ell \geqslant i^*\}$$
$$X^- = \{V_\ell \in X | i_\ell < i^*\} \quad \text{and} \quad X^+ = \{V_\ell \in X | i_\ell \geqslant i^*\},$$

where i^* is defined in condition *(i)* of the theorem's statement. Clearly, $X^- \subseteq T^-$ and $X^+ \subseteq T^+$. By condition *(i.a)*, $|T^+|$ is a random variable that follows the binomial distribution $Bin(n, \kappa_1)$. By condition *(i.b)*, for an arbitrary value z, the probability $\Pr_{\mathcal{D}}[|X^-| \geqslant z]$ is no more than $\Pr[|\tilde{X}^-| \geqslant z]$, where $|\tilde{X}^-|$ is a random variable that follows the binomial distribution $Bin(|T^-|, \kappa_2)$.

By the syntax of Helios ceremony, the voters can complain only when they are under under VSD atack, so it holds that $\mathcal{V}_{\text{comp}} \subseteq T$. Thus, we can partition the set of complaining voters $\mathcal{V}_{\text{comp}}$ into the two sets

$$\mathcal{V}^-_{\text{comp}} = \mathcal{V}_{\text{comp}} \cap T^- \quad \text{and} \quad \mathcal{V}^+_{\text{comp}} = \mathcal{V}_{\text{comp}} \cap T^+ .$$

By condition *(i.b)*, for an arbitrary value z, the probability $\Pr_{\mathcal{D}}[|\mathcal{V}^-_{\text{comp}}| \leqslant z]$ is no more than $\Pr[|\tilde{\mathcal{V}}^-_{\text{comp}}| \leqslant z]$, where $|\tilde{\mathcal{V}}^-_{\text{comp}}|$ follows the binomial distribution $Bin(|T^-| - |X^-|, 1 - \kappa_3)$. According to the above observations, for any $\Delta \in [0, 1)$ the following hold:

- $\Pr_{\mathcal{D}}[|X^+| \geqslant (1+\Delta)\kappa_1 n] \leqslant \Pr_{\mathcal{D}}[|T^+| \geqslant (1+\Delta)\kappa_1 n] \leqslant e^{-\kappa_1 n \frac{\Delta^2}{3}}$.
- If $|X^+| < (1+\Delta)\kappa_1 n$, then $|T^-| \geqslant |X^-| > |X| - (1+\Delta)\kappa_1 n$.
- $\Pr_{\mathcal{D}}[|X^-| \geqslant (1+\Delta)\kappa_2 |T^-|] \leqslant e^{-\kappa_2 |T^-| \frac{\Delta^2}{3}}$.
- If $|X^+| < (1+\Delta)\kappa_1 n$ and $|X^-| < (1+\Delta)\kappa_2 |T^-|$, then

$$|T^-| - |X^-| > \Big(\frac{1}{(1+\Delta)\kappa_2} - 1\Big)\big(|X| - (1+\Delta)\kappa_1 n\big) .$$

- $\Pr_{\mathcal{D}}[|\mathcal{V}^-_{\text{comp}}| \leqslant (1-\Delta)(1-\kappa_3)(|T^-| - |X^-|)] \leqslant e^{-(1-\kappa_3)(|T^-|-|X^-|)\frac{\Delta^2}{2}}$.

In order for $\mathcal{A}$ to be successful w.r.t. X it must hold that $|\mathcal{V}^-_{\text{comp}}| \leqslant \phi$. Therefore, since we assumed that $|X| \geqslant \delta/2$ and under the constraint that

$$\phi \leqslant (1-\Delta)(1-\kappa_3)\Big(\frac{1}{(1+\Delta)\kappa_2} - 1\Big)\Big(\frac{\delta}{2} - (1+\Delta)\kappa_1 n\Big) ,$$

we have that

$$
\begin{aligned}
&\Pr_{\mathcal{D}}[(\mathcal{A}\text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2)] = \\
&= \max\Big\{\Pr_{\mathcal{D}}\big[(\mathcal{A}\text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2) \;\Big|\; |X^+| \geqslant (1+\Delta)\kappa_1 n\big], \\
&\qquad \Pr_{\mathcal{D}}\big[(\mathcal{A}\text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2) \;\big|\; |X^+| < (1+\Delta)\kappa_1 n\big]\Big\} \leqslant \\
&\leqslant \max\Big\{\Pr_{\mathcal{D}}\big[|X^+| \geqslant (1+\Delta)\kappa_1 n\big], \\
&\qquad \Pr_{\mathcal{D}}\big[(\mathcal{A}\text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2) \;\big|\; |X^+| < (1+\Delta)\kappa_1 n\big]\Big\} \leqslant \\
&\leqslant \max\Big\{e^{-\kappa_1 n\frac{\Delta^2}{3}}, \\
&\quad \max\Big\{\Pr_{\mathcal{D}}\big[(\mathcal{A}\text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2) \;\big|\; \\
&\qquad \big|\; \big(|X^-| \geqslant (1+\Delta)\kappa_2|T^-|\big) \wedge \big(X^+| < (1+\Delta)\kappa_1 n\big)\big], \\
&\qquad \Pr_{\mathcal{D}}\big[(\mathcal{A}\text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2) \;\big|\; \\
&\qquad\quad \big|\big(|X^-| < (1+\Delta)\kappa_2|T^-|\big) \wedge \big(X^+| < (1+\Delta)\kappa_1 n\big)\big]\big\}\Big\} \leqslant \\
&\leqslant \max\Big\{e^{-\kappa_1 n\frac{\Delta^2}{3}}, e^{-\kappa_2(|X|-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{3}}, \\
&\quad \Pr_{\mathcal{D}}\big[(\mathcal{A}\text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2) \;\big|\; \\
&\qquad\quad \big|\big(|X^+| < (1+\Delta)\kappa_1 n\big) \wedge \big(|X^-| < (1+\Delta)\kappa_2|T^-|\big)\big]\Big\} \leqslant \\
&\leqslant \max\Big\{e^{-\kappa_1 n\frac{\Delta^2}{3}}, e^{-\kappa_2(|X|-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{3}}, \\
&\quad \Pr_{\mathcal{D}}\big[|\mathcal{V}^-_{\mathsf{comp}}| \leqslant \phi \;\big|\; \big(|X^+| < (1+\Delta)\kappa_1 n\big) \wedge \big(|X^-| < (1+\Delta)\kappa_2|T^-|\big)\big]\Big\} \leqslant \\
&\leqslant \max\Big\{e^{-\kappa_1 n\frac{\Delta^2}{3}}, e^{-\kappa_2(|X|-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{3}}, \\
&\quad \Pr_{\mathcal{D}}\big[|\mathcal{V}^-_{\mathsf{comp}}| \leqslant (1-\Delta)(1-\kappa_3)\Big(\frac{1}{(1+\Delta)\kappa_2} - 1\Big)\Big(|X| - (1+\Delta)\kappa_1 n\Big) \;\big|\; \\
&\qquad \big|\; \big(|X^+| < (1+\Delta)\kappa_1 n\big) \wedge \big(|X^-| < (1+\Delta)\kappa_2|T^-|\big)\big]\Big\} \leqslant \\
&\leqslant \max\Big\{e^{-\kappa_1 n\frac{\Delta^2}{3}}, e^{-\kappa_2(|X|-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{3}}, \\
&\quad \Pr_{\mathcal{D}}\big[|\mathcal{V}^-_{\mathsf{comp}}| \leqslant (1-\Delta)(1-\kappa_3)(|T^-| - |X^-|) \;\big|\; \\
&\qquad \big|\; \big(|X^+| < (1+\Delta)\kappa_1 n\big) \wedge \big(|X^-| < (1+\Delta)\kappa_2|T^-|\big)\big] \leqslant \\
&\quad \leqslant \max\Big\{e^{-\kappa_1 n\frac{\Delta^2}{3}}, e^{-\kappa_2(|X|-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{3}}, e^{-(1-\kappa_3)\left(\frac{1}{(1+\Delta)\kappa_2}-1\right)(|X|-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{2}}\Big\} \leqslant \\
&\quad \leqslant e^{-\min\left\{\kappa_1 n\,,\,\gamma(\frac{\delta}{2}-(1+\Delta)\kappa_1 n)\right\}\frac{\Delta^2}{3}}\,,
\end{aligned}
\tag{6.12}
$$

where $\gamma = \min\left\{\kappa_2\,,\,\frac{3}{2}(1-\kappa_3)\left(\frac{1}{(1+\Delta)\kappa_2}-1\right)\right\}$.

Bounding $\mathcal{A}$'s success probability w.r.t. Y when $|Y| \geqslant \delta/2$:

A replacement/deletion/insertion attack may be successful because (a) $\mathcal{A}$ has computed an adversarial ballot with the same hash values ψ_ℓ (collision attack) or (b) V_ℓ is not in $\mathcal{V}_{\mathsf{audit}}$. Given the subset $\mathcal{J}^*$ in condition *(ii)* of the stament, we partition Y into the subsets:

$$Y^{\in} = \{V_\ell \in Y | i_\ell \in \mathcal{J}^*\} \quad \text{and} \quad Y^{\notin} = \{V_\ell \in Y | i_\ell \notin \mathcal{J}^*\} \ .$$

By condition *(ii.a)*, $|Y^{\notin}|$ follows the binomial distribution $Bin(n, \mu_1)$. Moreover, by condition *(ii.b)*, the probability of a successful BB attack against any voter in $Y^{\in}$ is upper bounded by $\mu_2 + \epsilon'(\lambda)$ (the voter does not audit the BB or $\mathcal{A}$ finds a collision). Finally, in the case where $|Y^{\notin}| < (1+\Delta)\mu_1 n$, then $|Y^{\in}| = |Y| - |Y^{\notin}| > |Y| - (1+\Delta)\mu_1 n$. Thus, by the Chernoff bounds and for any $\Delta \in [0,1)$,

$$\begin{aligned}
&\Pr_{\mathcal{D}}[(\mathcal{A} \text{ successful w.r.t. } Y) \wedge (|Y| \geqslant \delta/2)] \leqslant \\
&\leqslant \max\Big\{ \Pr_{\mathcal{D}}\big[(\mathcal{A} \text{ successful w.r.t. } Y) \wedge (|Y| \geqslant \delta/2) \ \big|\ |Y^{\notin}| \geqslant (1+\Delta)\mu_1 n\big], \\
&\qquad \Pr_{\mathcal{D}}\big[(\mathcal{A} \text{ successful w.r.t. } Y) \wedge (|Y| \geqslant \delta/2) \ \big|\ |Y^{\notin}| < (1+\Delta)\mu_1 n\big] \Big\} \leqslant \\
&\leqslant \max\Big\{ \Pr_{\mathcal{D}}[(|Y^{\notin}| \geqslant (1+\Delta)\mu_1 n], \\
&\qquad \Pr_{\mathcal{D}}\big[(\mathcal{A} \text{ successful w.r.t. } Y) \wedge (|Y| \geqslant \delta/2) \ \big|\ |Y^{\notin}| < (1+\Delta)\mu_1 n\big] \Big\} \leqslant \\
&\leqslant \max\Big\{ e^{-\mu_1 n \frac{\Delta^2}{3}}, \big(\mu_2 + \epsilon'(\lambda)\big)^{|Y^{\in}|} \Big\} \leqslant \max\Big\{ e^{-\mu_1 n \frac{\Delta^2}{3}}, \big(\mu_2 + \epsilon'(\lambda)\big)^{|Y| - (1+\Delta)\mu_1 n} \Big\} \leqslant \\
&\leqslant \max\Big\{ e^{-\mu_1 n \frac{\Delta^2}{3}}, \mu_2^{\frac{\delta}{2}|-(1+\Delta)\mu_1 n} \Big\} + \mathsf{negl}(\lambda) = e^{-\min\left\{\mu_1 n \ , \ \ln\left(\frac{1}{\mu_2}\right)\left(\frac{\delta}{2} - (1+\Delta)\mu_1 n\right)\right\}} + \mathsf{negl}(\lambda) \ .
\end{aligned} \tag{6.13}$$

By Eq. (6.10),(6.11),(6.12),(6.13) we conclude that for any $\Delta \in [0,1)$ and for any δ, θ, the probability that $\mathcal{A}$ wins under the constraint

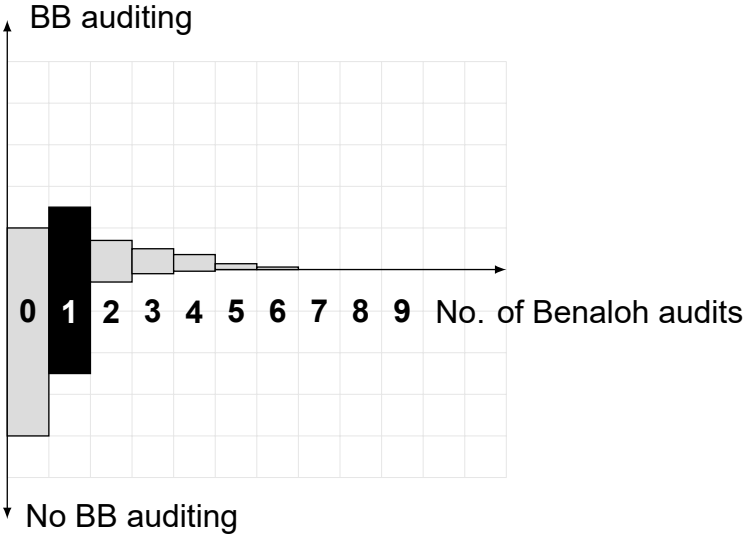
$$\phi \leqslant (1-\Delta)(1-\kappa_3)\Big(\frac{1}{(1+\Delta)\kappa_2} - 1\Big)\Big(\frac{\delta}{2} - (1+\Delta)\kappa_1 n\Big) \ ,$$

is no more than

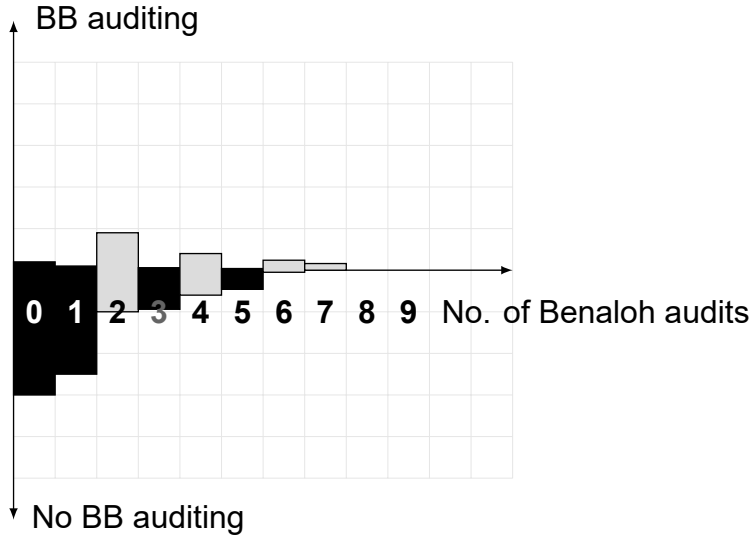
$$\begin{aligned}
&\Pr_{\mathcal{D}}[G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\mathcal{D},\delta,\theta,\phi}(1^\lambda, n, m, k) = 1] = \\
&= \Pr_{\mathcal{D}}\big[\big(G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\mathcal{D},\delta,\theta,\phi}(1^\lambda, n, m, k) = 1\big) \wedge A\big] + \\
&\qquad + \Pr_{\mathcal{D}}\big[\big(G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\mathcal{D},\delta,\theta,\phi}(1^\lambda, n, m, k) = 1\big) \wedge (\neg A)\big] \leqslant \\
&\leqslant \Pr_{\mathcal{D}}\big[\big(G_{\mathsf{E2E}}^{\mathcal{A},\mathcal{E},\mathcal{D},\delta,\theta,\phi}(1^\lambda, n, m, k) = 1\big) \ \big|\ (\neg F) \wedge (\neg A)\big] + (\mu_1 + \mu_2 - \mu_1\mu_2)^\theta + \mathsf{negl}(\lambda) \leqslant \\
&\leqslant \max\Big\{ \Pr_{\mathcal{D}}\big[(\mathcal{A} \text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2)\big] \ , \\
&\qquad \Pr_{\mathcal{D}}\big[(\mathcal{A} \text{ successful w.r.t. } X) \wedge (|X| < \delta/2)\big] \Big\} + (\mu_1 + \mu_2 - \mu_1\mu_2)^\theta + \mathsf{negl}(\lambda) \leqslant \\
&\leqslant \max\Big\{ \Pr_{\mathcal{D}}\big[(\mathcal{A} \text{ successful w.r.t. } X) \wedge (|X| \geqslant \delta/2)\big] \ , \\
&\qquad \Pr_{\mathcal{D}}\big[(\mathcal{A} \text{ successful w.r.t. } Y) \wedge (|Y| \geqslant \delta/2)\big] \Big\} + (\mu_1 + \mu_2 - \mu_1\mu_2)^\theta + \mathsf{negl}(\lambda) \leqslant
\end{aligned}$$

$$\leqslant \max\left\{ e^{-\min\left\{\kappa_1 n\,,\,\gamma(\frac{\delta}{2}-(1+\Delta)\kappa_1 n)\right\}\frac{\Delta^2}{3}}\;,\; e^{-\min\left\{\mu_1 n\,,\,\ln\left(\frac{1}{\mu_2}\right)(\frac{\delta}{2}-(1+\Delta)\mu_1 n)\right\}}\right\}+$$
$$+\,(\mu_1+\mu_2-\mu_1\mu_2)^{\theta}+\mathsf{negl}(\lambda)=$$
$$=e^{-\min\left\{\kappa_1 n\frac{\Delta^2}{3}\,,\,\mu_1 n\frac{\Delta^2}{3}\,,\,\gamma(\frac{\delta}{2}-(1+\Delta)\kappa_1 n)\frac{\Delta^2}{3}\,,\,\ln\left(\frac{1}{\mu_2}\right)(\frac{\delta}{2}-(1+\Delta)\mu_1 n)\right\}}+(\mu_1+\mu_2-\mu_1\mu_2)^{\theta}+\mathsf{negl}(\lambda)\;.$$

■



(a) A voter transducer distribution with vulnerable VSD auditing behaviour ($i^* = 1$).



(b) A voter transducer distribution with vulnerable BB auditing behaviour ($\mathcal{J}^* = \{0, 1, 3, 5\}$).

Figure 6.1: Assailable voter transducer distributions for Helios e-voting ceremony. The length of the bars is proportional to the probability of the corresponding event.

Illustrating Theorem 6.1

In order to provide intuition, we illustrate two representatives from the class of assailable voter transducer distributions that correspond to conditions (i) and (ii) of Theorem 6.1 in Figures 6.1(a) and 6.1(b) respectively.

Illustrating Theorem 6.2

To provide intuition, we illustrate an example of a voter transducer distributions that corresponds to conditions (i) and (ii) of Theorem 6.2 in Figure 6.2.

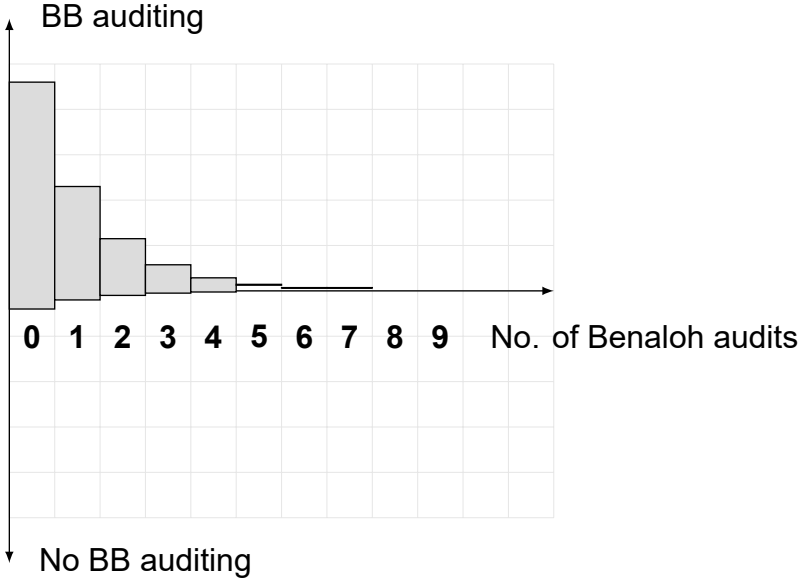


Figure 6.2: A voter transducer distribution with resistance against VSD and BB attacks ($\kappa_1 = \mu_1 = 0.03125, \kappa_2 = 0.5, \mu = 0.08$ w.r.t. $i^* = 5$, $\mathcal{J}^* = \{0, 1, 2, 3, 4\}$). The length of the bars is proportional to the probability of the corresponding event.

6.3.4 On the tightness of the conditions of Theorems 6.1 and 6.2

The conditions stated in Theorems 6.1 and 6.2 determine two classes of voter transducer distributions that correspond to vulnerable and insusceptible settings, respectively. We observe that weakening the condition (i) of Theorem 6.1 (resp. (i) of Theorem 6.2) cannot imply vulnerability (resp. security). Namely, in condition (i) of Theorem 6.1, if one of (1.a),(1.b) or (1.c) does not hold, then the adversary cannot be certain that it will achieve a sufficiently large deviation from VSD attacks without increasing rapidly the number of complaints. On the other hand, if condition (i.a) of Theorem 6.2 does not hold, then E2E verifiability cannot be preserved when (1.b) becomes a disjunction, since a high complaint rate alone is meaningless if the adversary has high success rate of VSD attacks.

Consequently, it is not possible to achieve logical (i.e. probability thresholds are consid-

ered either sufficiently ***high*** or sufficiently ***low***) tightness for interesting sets of parameters d, θ, ϕ only by negating the conditions of each of the two theorems. However, this is possible if we assume that the voter's complaining behaviour is balanced by flipping coins in order to decide whether they will complain in case of unsuccessful termination, i.e. if we set $\kappa_3 = 1 - \kappa_3 = 1/2$. Specifically, given that $\kappa_3 = 1/2$ is a "neutral" value, we can restate the conditions of Theorems 6.1 and 6.2 in their logical form as follows:

THEOREM 6.1 (logical version)

A voter transducer distribution is susceptible to VSD or/and BB attacks *if at least one of the following two conditions holds:*

*(i). There is an $i^* \in \{0, \ldots, q\}$ such that (i.a) the probability that a voter executes at least i^* Benaloh audits is **high** AND (i.b) the probability that a voter, given that she has executed at least i^* Benaloh audits, will cast her vote after exactly i^* Benaloh audits is **high**.*

OR

*(ii). There is a subset $\mathcal{J}^* \subseteq \{0, \ldots, q\}$ such that (ii.a) the probability that a voter executes j Benaloh audits for some $j \in \mathcal{J}^*$ is **high** AND (ii.b) for every $j \in \mathcal{J}^*$, the probability that a voter, given she has executed j Benaloh audits, will not audit the BB **high**.*

THEOREM 6.2 (logical version)

A voter transducer distribution achieves resistance against VSD and BB attacks *if the following two conditions hold:*

*(i) There is an $i^* \in \{0, \ldots, q+1\}$ such that (i.a) the probability that a voter executes at least i^* Benaloh audits is **low** and (i.b) for every $i \in \{0, \ldots, q\}$, if $i < i^*$, then the probability that a voter, given that she will execute at least i Benaloh audits, will cast her vote after exactly i Benaloh audits is **low**.*

AND

*(ii) There is a subset $\mathcal{J}^* \subseteq \{0, \ldots, q\}$ such that (ii.a) the probability that a voter executes j Benaloh audits for some $j \in \mathcal{J}^*$ is **high** AND (ii.b) for every $j \in \mathcal{J}^*$, the probability that a voter, given she has executed j Benaloh audits, will audit the BB is **high**.*

Based on the above statements, we show that the following hold:

1. *If condition (i) of Theorem 6.1 does not hold, then condition (i) of Theorem 6.2 holds*: let $\mathcal{I}_1$ be the set of $i \in \{0, \ldots, q\}$ s.t. the probability that a voter executes at least i Benaloh

audits is ***high***. By the negation of condition (i) of Theorem 6.1, for every $i \in \mathcal{I}_1$, the probability that a voter, given that she will execute at least i Benaloh audits, will cast her vote after exactly i Benaloh audits is ***low***. Observe that $\mathcal{I}_1$ is not empty, as $0 \in \mathcal{I}_1$. Therefore, if we set $i^* = \max\{i \mid i \in \mathcal{I}_1\} + 1$, then, by definition, i^* satisfies the conditions (i.a) and (i.b) of Theorem 6.2.

2. *If condition (i) of Theorem 6.2 does not hold, then condition (i) of Theorem 6.1 holds*: let $\mathcal{I}_2$ be the set of $i \in \{0, \ldots, q+1\}$ s.t. the probability that a voter executes at least i Benaloh audits is ***low***. Clearly, $\mathcal{I}_2$ is non-empty, since $q+1 \in \mathcal{I}_2$). By the negation of condition (i) of Theorem 6.2, for every $i \in \mathcal{I}_2$ there is an $i' < i$ s.t. the probability that a voter, given that she will execute at least i Benaloh audits, will cast her vote after exactly i' Benaloh audits is ***high***. In this case, we set i^* to be this i' that corresponds to the minimum i in $\mathcal{I}_2$ (note that $i^* \geqslant 0$, since $0 \notin \mathcal{I}_2$). In both cases, i^* satisfies the conditions (i.a) and (i.b) of Theorem 6.1.

3. *If condition (ii) of Theorem 6.1 does not hold, then condition (ii) of Theorem 6.2 holds*: by an averaging argument, there is a $j \in \{0, \ldots, q\}$ s.t. the probability that a voter executes j Benaloh audits is at least $1/(q+1)$. Assuming that the maximum number of Benaloh audits q is small (which is meaningful for most interesting cases in practice), we can consider $1/(q+1)$ to be a sufficiently ***high*** probability. By the negation of condition (ii) of Theorem 6.1, for singleton $\{j\}$, the probability that a voter that executes j Benaloh audits wil audit the BB is ***high***. Thus, the set $\mathcal{J}^*$ that contains all j for which the voter executes j Benaloh audits with probability at least $1/(q+1)$ satisfies the conditions (ii.a) and (ii.b) of Theorem 6.2.

4. *The negation of condition (ii) of Theorem 6.2 implies the condition (ii) of Theorem 6.1*: by the negation of condition (ii) of Theorem 6.2, every j for which the voter executes j Benaloh audits with probability at least $1/(q+1)$ (***high***) determines a subset (singleton $\{j\}$) of ***low*** BB auditing probability. Thus, the set $\mathcal{J}^*$ that contains all j for which the voter executes j Benaloh audits with probability at least $1/(q+1)$ satisfies the conditions (ii.a) and (ii.b) of Theorem 6.1.

6.4 Evaluating the E2E verifiability of an e-voting ceremony

In this section, we evaluate our results for the E2E verifiability of Helios, by instantiating the bounds in Theorems 6.1 and 6.2 for various voter transducer distributions. Our evaluations are separated into two categories: (i) evaluations that are based on actual human data that derive from elections using Helios and (ii) evaluations that are based on simulated data for various sets of parameters.

6.4.1 Evaluations based on human data.

Our human data are sampled from two independent surveys: the first sample is from the member elections of the Board of Directors of the International Association for Cryptographic Research (IACR); the second is a non-binding poll among the students of the Department of Informatics and Telecommunications (DI&T) of the University of Athens. In the following subsection, we present at length our methodology for the two surveys.

6.4.1.1 Methodology of our surveys with human subjects

The methodology for IACR elections

We conducted our survey using the SurveyMonkey tool. Specifically, we formed a questionnaire that consisted of three questions, as shown in Figure 6.3.

QUESTIONNAIRE

Q1. In the last IACR election you participated, did you use the *"audit your ballot"* functionality (where you get to see the opening of the ciphertext containing your vote)?

Yes: ☐ **No:** ☐

Q2. If you answered "**Yes**" in the above question, how many times did you audit?

Enter a positive integer: ☐

Q3. Did you verify that the smart ballot tracker (the hash of your submitted ciphertext) was actually posted on the ballot tracking center (the public web-site that lists all encrypted ballots)?

Yes: ☐ **No:** ☐

Figure 6.3: The questionnaire used in the survey on the voter's behaviour at the IACR elections.

The questionnaire was delivered to the IACR board. In turn, the board sent an open call to the IACR members for volunteering to participate in our survey. By the end of the survey, we collected 35 responses, from which we extracted the data presented in Table 6.1.

The methodology for DI&T poll

We conducted a non-binding poll among the students of the DI&T Department of the University of Athens. During a lecture of the Computer Security course, we gave a presentation of Helios, focusing on the importance of auditing their ballots. Then, we asked the

Table 6.1: Distribution of the voters' VSD and BB auditing behaviour in the IACR sample consisting of 35 responders.

	Benaloh audits							
	0		1		2		3	
BB **audit**	Yes	No	Yes	No	Yes	No	Yes	No
	2	22	4	5	1	0	1	0

students to participate in an election run using Helios which concept concerned the improvement of their daily student life. Specifically, the survey consisted of two stages; in the first stage, the students had a period of one week prior to the election to form a proposal that would reply to the following question:

Given a € 10,000 budget, which department facility would you suggest that should be updated or developed?

In the second stage, at the voting phase, all the submitted proposals where considered as options for the above question. In detail, the question as shown in the Helios booth template is depicted in Figure 6.4.

QUESTION

Given a € 10,000 budget, which department facility would you suggest that should be updated or developed?

Select up to 2 options:

1. Improving WiFi coverage in all areas of the department building complex. ☐
2. Extension of night lighting in all external areas of the building complex. ☐
3. Printer room with off-hours student access. ☐
4. Extended access to student reading room via card based gate access control. ☐

Figure 6.4: The question template at the DI&T poll.

A total of 49 students participated in our survey. We modified the Helios codebase so that our server could track the auditing behaviour of the participants. The data extracted from the voting process are presented in Table 6.2.

Table 6.3: The formula and the security significance of parameters $\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2$ used in Theorem 6.1 for given $i \in \{0, \ldots, q\}$ and $\mathcal{J} \subseteq \{0, \ldots, q\}$, where q is the maximum number of Benaloh audits. $E_{i,c,a}$ is the event that voter's behaviour follows the transducer $M_{i,c,a}$.

Parameter	Formula for the parameter	Security Significance
κ_1	$\mathsf{Pr}\left[\bigvee_{\substack{0 \leqslant t < i \\ c,a \in \{0,1\}}} E_{t,c,a}\right]$	As κ_1 decreases, the guarantee that the voter will execute at least i-Benaloh audits increases.
κ_2	$\mathsf{Pr}\left[\bigvee_{c,a \in \{0,1\}} E_{i,c,a} \mid \bigvee_{\substack{0 \leqslant t < i \\ c,a \in \{0,1\}}} E_{t,c,a}\right]$	As κ_2 decreases, the success rate of a VSD attack after the i-Benaloh audit increases.
κ_3	$\mathsf{Pr}\left[E_{i,0,0} \vee E_{i,0,1}\right]$	As κ_3 decreases, the complaint rate due to failed VSD attacks after the i-Benaloh audit increases.
μ_1	$\mathsf{Pr}\left[\bigvee_{\substack{j \notin \mathcal{J} \\ c,a \in \{0,1\}}} E_{j,c,a}\right]$	As μ_1 decreases, the rate of voters that "fall" into the target subset $\mathcal{J}$ increases.
μ_2	$\max_{j \in \mathcal{J}} \left\{\mathsf{Pr}\left[E_{j,0,1} \vee E_{j,1,1}\right]\right\}$	As μ_2 decreases, the success rate of a BB attack against a voter that "falls" into the target subset $\mathcal{J}$ increases.

Table 6.2: Distribution of the voters' VSD auditing behaviour at the DI&T poll. The sample consists of 49 participants.

Benaloh audits		
0	1	2
20	27	2

Parameter computation

The parameters $\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2$ used in Theorem 6.1 express the vulnerability of Helios voting ceremony against verifiability attacks w.r.t. a specific voter transducer distribution. It is easy to see that every $i \in \{0, \ldots, q\}$ and $\mathcal{J} \subseteq \{0, \ldots, q\}$ (where q is the maximum number of Benaloh audits) imply a set of parameters $(\kappa_1, \kappa_2, \kappa_3)$ and (μ_1, μ_2) that determine the success probability of an attacker against the VSD vulnerability and the BB vulnerability when the voter executes i and $j \in \mathcal{J}$ Benaloh audits respectively. The formulas and the security significance of parameters $\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2$ is explained in Table 6.3. There, we can deduce that parameters $\kappa_1, \kappa_3, \mu_1$ determine the ***size*** of the subsets of vulnerable voters, while κ_2, μ_2 can be seen as measures of the ***quality*** of the VSD and BB attacks.

In order to evaluate the vulnerability of the voter behaviour in each survey we performed the following procedure:

- We focused on maximizing the success probability that each type of attack may be mounted leaving the parameters δ, θ, ϕ as free variables[2].

[2]Following a different approach, one could also consider optimizing all parameters simultaneously in-

Table 6.4: Instantiated parameters $\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2$ of Theorem 6.1 for the IACR and the DI&T surveys.

Survey	i^*	$\mathcal{J}^*$	Parameters				
			κ_1	κ_2	κ_3	μ_1	μ_2
IACR elections	0	$\{0\}$	0	0.315	0.5	0.315	0.084
DI&T poll	1	–	0.408	0.069	0.5	–	–

- For both surveys, no complaints or audit failures were reported. Hence, due to lack of data, we choose a "neutral" value for κ_3 equal to 0.5 (see also Subsection 6.3.4). Note that our analysis will hold for any other *not close to* 0 value of κ_3. The case of $\kappa_3 = 0$, i.e., when the voter always complains to the authority when a Benaloh audit goes wrong, would make VSD attacks unattractive in the case that ϕ is small and would suggest that the attacker will opt for BB attacks (if such attacks are feasible which depends on μ_1, μ_2).
- For both surveys, we ran an exhaustive search in all possible numbers of Benaloh audits to locate the index i^* s.t. the parameters κ_1, κ_2 that maximize the probability of success stated in Theorem 6.1:*condition (i)*. Equivalently, we searched for the values κ_1, κ_2 that maximize the function

  $$F_\Delta(\kappa_1, \kappa_2) = (1 - \kappa_1)(\kappa_2 - \Delta + \Delta\kappa_2)(1 - \kappa_2)$$

  for a suitably small value of $\Delta \in [0, 1)$.
- For the IACR survey, we ran an exhaustive search in all subsets of $\{0, 1, 2\}$ to locate the subset $\mathcal{J}^*$ s.t. the parameters μ_1, μ_2 that maximize the probability of success stated in Theorem 6.1:*condition (ii)*, lower bounded by the equation

  $$(1 - e^{-(1-\mu_1)n\frac{\Delta^2}{2}})(1 - \mu_2)^\delta, \quad \text{where } \Delta \in [0, 1).$$

  Since the probability bound drops exponentially as the tally deviation δ increases, the effectiveness of the term $(1-e^{-(1-\mu_1)n\frac{\Delta^2}{2}})$ quickly becomes insignificant as compared with the term $(1-\mu_2)^\delta$. Consequently, we concentrated on the asymptotic behaviour of the equation by searching for the minimum μ_2 that leads to a slower decreasing rate.

Following the above procedure, we computed the optimal (from an adversarial point of view) sets of parameters $\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2$ as shown in Table 6.4.

6.4.2 Analysis of the experiments

Analysis of the IACR survey

cluding δ, θ, ϕ. Performing such analysis could be interesting future work; nevertheless, our analysis already reveals significant security deficiencies in our experiments.

From the first row of Table 6.4, we read that $\mu_2 = 0.084$ which is a very small value as opposed to $\kappa_2 = 0.315$. Thus, we expect that elections where the electorate follows the voter transducer distribution of IACR elections are much more vulnerable to BB attacks rather than VSD attacks. Indeed, this is consistent with the analysis that we describe below.

We computed the percentage of *tally deviation*/*No. of voters* that the adversary can achieve when the success probability is lower bounded by 25%, 10%, 5% and 1% for various electorate scales. Specifically, we observed that the success probability bounds stated in Theorem 6.1 express more accurately the effectiveness of the adversarial strategy for (i) medium to large scale elections when the adversary attacks via the VSD and (ii) for small to medium scale elections when the adversary attacks via the BB. As a consequence, we present our analysis for $n = 100, 500, 1000, 2500$ and 5000 voters w.r.t. BB attack effectiveness and for $n = 5000, 10000$ and 50000 voters w.r.t. VSD attack effectiveness. Our findings are shown in the tables in Tables 6.5 and 6.7.

Table 6.5: Percentage of *tally deviation*/*No. of voters* achieved in elections under BB attack strategies against electorates following the voter transducer distribution of IACR elections. The attack succeeds even when $\theta = n$ and $\phi = 0$.

Voters	**Success probability %**			
	$\geqslant 25$	$\geqslant 10$	$\geqslant 5$	$\geqslant 1$
100	15.92	26.4	34.42	51.42
500	3.18	5.28	6.87	10.56
1000	1.59	2.64	3.42	5.28
2500	0.636	1.05	1.37	2.11
5000	0.31	0.52	0.68	1.05

The data in Table 6.5 illustrate the power of BB attacks against compact bodies of voters (e.g. organizations, unions, board elections, etc.) where BB auditing is rare. We can see that in the order of hundreds, more than 5% of the votes could be swapped with significant probability of no detection. This power deteriorates rapidly as we enter the order of thousands, however, the election result could still be undermined, as deviation between 1%-2%, is possible, without the risk of *any* complaint due to unsuccessful engagement in the **Cast** ceremony (i.e. $\theta = n$ and $\phi = 0$). Therefore, even in a setting of high complaint rate (κ_3 is close to 0), the adversary may turn into a BB attack strategy and still be able to alter radically the election result, as marginal differences are common in all types of elections. We stress that from published data we are aware of, there have been elections for the IACR board where the votes for winning candidates were closer than 3% to the votes of candidates that lost in the election. Therefore, if the voter distribution had been as the one derived by Table 6.4, and 500 members had voted, the result could have been overturned with success probability 25% even if a single complaint was considered to be a "stop election event" (since $\phi = 0$).

Table 6.6: Success probability of a hypothetical BB attack strategy against the IACR elections for the Board of Directors per election year. The success probability is computed given the number of participants and the cutoff between the last elected director and the first candidate that was not elected. The dashed line denotes the actual start of Helios use for IACR elections. Regarding the year 2007, no data were recorded in `https://www.iacr.org/elections/`.

Year	Participants	Cutoff %	Success probability %
2015	437	6.87	7.35
2014	575	5.57	6.17
2013	637	2.99	19.14
2012	518	11.59	0.5
2011	621	4.03	11.35
2010	475	8.64	2.82
2009	325	4.93	24.8
2008	312	0.33	91.66
2007	–	–	–
2006	324	4.33	29.57

To provide more context, in Table 6.6, we provide the cutoff between elected and non-elected candidates for the last 10 years of IACR elections for the Board of Directors, followed by the exact success probability of a hypothetical BB attack strategy to overturn the election result given the actual number of cast ballots per year. We observe that the attacker success probability for many of the elections is considerable.

Table 6.7: Effectiveness of VSD attack strategies against electorates with $n = 5000, 10000$ and 50000 voters following the voter transducer distribution in IACR elections. In the table, δ/n is the percentage of *tally deviation*/*No. of voters*, θ/n is the ratio of honest successful voters in % and ϕ/n is the ratio of honest complaining voters in %.

Success probability %	n=5000			n=10000			n=50000		
	δ/n	θ/n	ϕ/n	δ/n	θ/n	ϕ/n	δ/n	θ/n	ϕ/n
$\geqslant 25$	51.8	54.3	25.7	57.5	59.6	21.9	63.9	65.0	18.1
$\geqslant 10$	52.8	55.3	25.1	58.1	60.2	21.5	64.1	65.2	18.0
$\geqslant 5$	53.2	55.6	24.8	58.3	60.3	21.4	64.2	65.2	17.9
$\geqslant 1$	53.4	55.9	24.7	58.4	60.5	21.3	64.3	65.3	17.8

On the other hand, as already mentioned, the effectiveness of VSD attacks is clear if we scale the electorate in the order of thousands and above. As we see from the results in Table 6.7, a VSD attack strategy against an election that follows the voter distribution in IACR elections would not have a great impact unless an unnatural number of complaints could be tolerated. Indeed, even for the scale of 50000 voters, the rate of complaints that is ignored must be close to 17% which is rather unacceptable in a real world setting (such number of complaints would most definitely lead to a stop election event).

We conclude that the IACR voter behaviour is susceptible to BB attacks with significant probability of success but not VSD attacks unless there is high tolerance in voter complaints.

Analysis of the DI&T poll

From the second row of Table 6.4, we read that $\kappa_2 = 0.069$ which is a very small value. Therefore, we expect that voters' behaviour in DI&T poll will be vulnerable to VSD attacks. Our results are presented in Table 6.8.

Table 6.8: Effectiveness of VSD attack strategies against electorates with $n = 100000$ voters following the voter transducer distribution of elections DI&T poll. The table notation $\delta/n, \theta/n$, ϕ/n is as in Table 6.7.

Success probability %	d/n	θ/n	ϕ/n
$\geqslant 25$	52.87	94.67	27.28
$\geqslant 10$	53.00	94.75	26.76
$\geqslant 5$	53.04	94.77	26.63
$\geqslant 1$	53.07	94.79	26.53

It is easy to see that the data in Table 6.8 add to the intuition on the power of the VSD attacks. One may observe that a very small value of $\kappa_2 = 0.069$ for election DI&T poll leads to efficient attacks while keeping a very high rate of honest voters ($\approx 95\%$), as compared with the cases for elections IACR elections ($\approx 65\%$) where $\kappa_2 = 0.315$.

In the analysis of Table 6.8, we scaled to 100000 voters so that the probability bound in Theorem 6.1 reveals the effectiveness of the VSD attacker. Of course, this does not mean that a medium scale election where the probability of a successful VSD attack is $1 - \kappa_2 = 93.1\%$ is not assailable. For instance, consider an electorate of $n = 500$ voters following the transducer distribution of the DI&T poll and a VSD attacker as the one described in the proof of Theorem 6.1. It easy to show that the attacker can achieve tally deviation $\beta\%$ without *any* complaint (i.e., $\theta = n$ and $\phi = 0$ as in a BB attack strategy) with probability at least

$$(1 - e^{-(1-\kappa_1)n\frac{\delta^2}{2}})(1 - \kappa_2)^{\beta n} = (1 - e^{-148\delta^2})(0.931)^{500\beta}, \tag{6.14}$$

for $d \leqslant (1 - \delta)296$ and any $\delta \in [0, 1)$. In Table 6.9, we present the ratio of tally deviation achieved by the attacker for various success probabilities, as derived from Eq. (6.14). Observe that tally deviation 5% may occur with 16.7% probability, which is certainly significant and reveals VSD vulnerability even at medium scale elections.

We conclude that the DI&T voter behaviour is susceptible to VSD attacks with significant probability. We cannot draw a conclusion for BB attacks since we did not collect auditing data for this case.

Table 6.9: Percentage of *tally deviation*/*No. of voters* achieved in elections under VSD attack strategies against electorates of 500 voters following the voter transducer distribution of DI&T poll. The attack succeeds even when $\theta = n$ and $\phi = 0$.

Success probability %			
$\geqslant 25$	$\geqslant 10$	$\geqslant 5$	$\geqslant 1$
0.013	2.8	16.7	69.9

6.4.3 Evaluations based on simulated data

Our human data analysis is obtained by real bodies of voters that have an imperfect voting behaviour. To understand what would be the security level of a Helios e-voting ceremony when executed by an "ideally trained" electorate, we evaluated the security of simulated elections. Namely, we computed the *detection probability* that Theorem 6.2can guarantee defined as $(1-\epsilon)\cdot 100\%$, where ϵ is the error stated in Theorem 6.2. The voter distributions we considered were chosen from the collection $\{\mathbf{D}_{p,q}\}_{p\in[0,1].q\in\mathbb{N}}$ defined as follows:

$\{\mathbf{D}_{p,q}\}_{p\in[0,1].q\in\mathbb{N}}$: the voter flips a coin b with bias p to perform Benaloh audits when $b = 1$, up to a maximum number of q audits. In any case of termination, she flips a coin b' with bias p to perform BB audit when $b' = 1$.

By choosing as VSD resistance index $i^* = q$ and BB resistance set $\mathcal{J}^* = \{0, \ldots, q-1\}$ we compute the parameters

$$\kappa_1 = \mu_1 = p^q, \quad \kappa_2 = \mu_2 = 1 - p \ ,$$

where we also set κ_3 to the balanced parameter 1/2. Intuitively, this type of voter behaviour should result in a sufficient level of resistance against of VSD and BB attacks, if the values $1-p$ and p^q are small enough. In order for this to hold, the number of maximum allowed Benaloh audits q should be increased when the bias p becomes larger, as otherwise the attacker could wait and attack the VSD when q audits happen (which is likely if the audit rate is high).

By applying the above parameters in Theorem 6.2, we plot the probability error fluctuating $p, q, \Delta)$, the number of all voters n and honest voters θ, expressed by the following function

$$G_\Delta(p,q,n) = e^{-\min\left\{p^q n\frac{\Delta^2}{3}\ ,\ \gamma(\frac{\delta}{2}-(1+\Delta)p^q n)\frac{\Delta^2}{3}\ ,\ \ln\left(\frac{1}{1-p}\right)(\frac{\delta}{2}-(1+\Delta)p^q n)\right\}} \ ,$$

where $\gamma = \min\left\{1-p\ ,\ \frac{3}{4}\left(\frac{1}{(1+\Delta)(1-p)}-1\right)\right\}$. Note that we omit the term $(\mu_1+\mu_2-\mu_1\mu_2)^\theta$ and the negligible term, since they become very small for reasonably large θ, λ.

As an example, we present our findings for $n = 250000$ voters for distributions $\mathbf{D}_{p,q}$, where $p = 0.25, 0.5, 0.75$ and $q = 3, 5, 8, 10$ in Table 6.10. The empty cells appear when no meaningful error can be computed.

Table 6.10: Security w.r.t. detection probability $90\%, 99\%$ and $99,9\%$ of *(tally deviation)/(No. of voters)* percentage for elections with $n = 250000$ voters for distributions $\mathbf{D}_{p,q}$, where $p = 0.25, 0.5, 0.75$ and $q = 3, 5, 8, 10$. The detection probability is defined as $(1 - \epsilon) \cdot 100\%$, where ϵ is the error stated in Theorem 6.2. The table notation δ/n, ϕ/n is as in Table 6.7.

Distribution	**Detection Probability**					
	90%		99%		99,9%	
	δ/n	ϕ/n	δ/n	ϕ/n	δ/n	ϕ/n
$\mathbf{D}_{0.25,3}$	8.8	0.3	14.25	0.6	19.7	0.9
$\mathbf{D}_{0.25,5}$	3.44	0.01	6.63	0.03	9.83	0.04
$\mathbf{D}_{0.25,8}$						
$\mathbf{D}_{0.25,10}$						
$\mathbf{D}_{0.5,3}$						
$\mathbf{D}_{0.5,5}$	7.98	0.2	9.08	0.4	10.19	0.61
$\mathbf{D}_{0.5,8}$	1.21	0.03	1.49	0.07	1.76	0.1
$\mathbf{D}_{0.5,10}$						
$\mathbf{D}_{0.75,3}$						
$\mathbf{D}_{0.75,5}$	54.41	1.32	56.62	2.61	58.8	3.91
$\mathbf{D}_{0.75,8}$	24.23	1.32	26.44	2.61	54.23	3.92
$\mathbf{D}_{0.75,10}$	14.06	0.25	14.62	0.51	15.17	0.77

We observe that when the complaint rate is balanced, acceptable levels of security (e.g., (tally deviation)/(No. of voters) $\leqslant 3\%$ or error probability $\leqslant 1\%$) can be achieved only when a very small rate of complaining voters can be allowed. As a result, the auditing and complaining behaviour of the voters must be almost ideal in order for a high level of security to be achieved.

6.5 Voter Privacy/PCR of Helios e-Voting Ceremony

In this section, we prove the voter privacy/PCR of the Helios e-voting ceremony. The proof is carried out via a reduction. Namely, we show that if there exists a PPT adversary $\mathcal{A}$ that wins the voter privacy/PCR game for Helios with non-negligible distinguishing advantage, then there exists a PPT adversary $\mathcal{B}$ that breaks the IND-CPA security of the ElGamal encryption scheme with blackbox access to $\mathcal{A}$. Throughout the proof, we view $H(\cdot)$ as a RO.

Theorem 6.3. *Assume an election run of Helios with n voters, m candidates and k trustees. Assume that the hash function $H(\cdot)$ considered in Section 6.2 is a random oracle. Let $m, n, k, t \in \mathbb{N}$ be polynomial in λ. If the underlying ElGamal encryption scheme is IND-CPA secure, then there exists a view simulator $\mathcal{S}$ s.t. for all distribution collections $\mathcal{D}$ and for all PPT adversaries $\mathcal{A}$, the distinguishing advantage of the voter privacy/PCR game*

for Helios is

$$\left| \Pr[G^{\mathcal{A},\mathcal{S},\mathcal{D}}_{t\text{-priv}}(1^\lambda, n, m, k) = 1] - 1/2 \right| = \mathsf{negl}(\lambda) \ .$$

Proof. The proof consists of (i) the construction of view simulator $\mathcal{S}$ for the voter privacy/PCR game, and (ii) the reduction showing that any adversary who has non-negligible advantage in the voter privacy/PCR game can be used to break the IND-CPA security of the ElGamal encryption scheme.

The construction of view simulator $\mathcal{S}$:

Recall that in the execution of the **Cast** ceremony, V_ℓ and VSD are controlled by the challenger. V_ℓ behaves according to the sampled transducer $M_{i_\ell,c_\ell,a_\ell} \leftarrow \mathbf{D}_\ell$, which audits the ciphertexts produced by the VSD i_ℓ times before encrypting its real candidate selection. Note that value of c_ℓ, a_ℓ is irrelevant for privacy, as the EA is honest and checks the validity of all the submitted ballots as well as the associated NIZK proofs in the privacy game. For the j-th ciphertext auditing, it sends the VSD the candidate selection $\mathcal{U}^b_\ell$ and obtains the created ballot $\psi_{\ell,j}$ and the corresponding randomness $r_{\ell,j}$ from the VSD. After the j-th auditing, it sends the candidate selection $\mathcal{U}^b_\ell$ to the VSD and casts the created ballot ψ_ℓ together with its identity ID_ℓ. The view of V_ℓ is defined as $view_\ell = \langle(\mathsf{Pub}, s_\ell, \mathcal{U}^b_\ell), (\psi_{\ell,j}, r_{\ell,j})_{j\in[i_\ell]}, \mathsf{audit}_\ell\rangle$, where $\mathsf{audit}_\ell = (\psi^1_\ell, \mathsf{ID}_\ell)$ is the receipt.

The simulator $\mathcal{S}$ randomly picks a coin $b' \leftarrow \{0,1\}$ on its first execution and maintains the coin b' throughout the privacy game. On input $(view_\ell, \mathcal{U}^0_\ell, \mathcal{U}^1_\ell)$, $\mathcal{S}$ for $j \in \{1, \ldots, i_\ell\}$ creates ballot $\psi'_{\ell,j}$ using a fresh randomness $r'_{\ell,j}$ for the candidate selection $\mathcal{U}^{b'}_\ell$, as VSD would. It then outputs the simulated view $view'_\ell = \langle(\mathsf{Pub}, s_\ell, \mathcal{U}^{b'}_\ell), (\psi_{\ell,j}, r_{\ell,j})_{j\in[i_\ell]}, \mathsf{audit}_\ell\rangle$, where $\mathsf{audit}_\ell = (\psi^1_\ell, \mathsf{ID}_\ell)$ remains the same.

The reduction :

Assume that $\mathcal{A}$ is a PPT adversary that wins the voter privacy/PCR game $G^{\mathcal{A},\mathcal{S},\mathcal{D}}_{t\text{-priv}}(1^\lambda, m, n, k)$, for some $m, t, n, k \in$ polynomial in λ, We construct an adversary $\mathcal{B}$ that tries to use $\mathcal{A}$ in a blackbox manner to attack the IND-CPA security of the ElGamal encryption. As shown in [15], strong Fiat-Shamir transformations of Σ protocols are simulation sound extractable. More specifically, for any prover $\mathcal{A}$ who outputs polynomially many statement/proof pairs $(\mathbf{Y}, \square)$, there exists an efficient knowledge extractor $\mathcal{K}$, given black-box access to $\mathcal{A}$ and may invoke further copies of $\mathcal{A}$ using the same randomness as was used in the main run, can extract a vector of witnesses **w** corresponding to the statements **Y**. Consider the following sequence of games from G_0 to G_3.

Game G_0: The actual game $G^{\mathcal{A},\mathcal{S},\mathcal{D}}_{t\text{-priv}}(1^\lambda, n, m, k)$, where the challenger uses $\mathcal{U}^b_\ell$ in the **Cast** ceremony and the above simulator $\mathcal{S}$ is invoked when $b = 1$.

Game G_1: Game G_1 is the same as Game G_0 except the following. The challenger Ch controls the RO $H(\cdot)$. After the **Cast** phase, Ch invokes the knowledge extractor $\mathcal{K}$ to

extract the partial secret keys $\{\mathsf{sk}_i\}_{i\neq w}$ of all the other trustees that $\mathcal{A}$ controls and the candidate selections of all the casted ballots submitted by the corrupted voters. The challenger Ch aborts if the extraction fails; otherwise, Ch completes the experiment.

Game G_2: Game G_2 is the same as Game G_1 except the following. The challenger Ch computes the election result $\langle x_1, \ldots, x_m \rangle$ that corresponds to the ballots that $\mathcal{A}$ posted on the BB according to the candidate selections of the corrupted voters extracted in Game G_1. Denote the final tally ElGamal ciphertext vector as $\langle C_1, \ldots, C_m \rangle$, where $C_j := (C_j^{(0)}, C_j^{(1)}) = (g^{r_j}, g^{x_j} \cdot h^{r_j})$ for some r_j. For $j \in \{1, \ldots, m\}$, the trustee T_w produces its partial decryption of C_j as $D_{w,j} = C_j^{(1)} / (g^{x_j} \cdot (C_j^{(0)})^{\sum_{i\neq w} \mathsf{sk}_i})$ together with simulated NIZK proofs without using its partial secret key.

Game G_3: Game G_3 is the same as Game G_2 except the following. For all the voters $V_\ell \in \tilde{\mathcal{V}}$, the challenger Ch submits a vector of encryptions of 0 together with the simulated NIZK proof instead of the real ciphertexts of the candidate selections. Besides, the challenger Ch always give the adversary $\mathcal{A}$ the simulated **Cast** views, ignoring the bit b.

Define $\mathsf{Adv}_{G_i,G_j}(\mathcal{A}) := \frac{1}{2}\big|\,\mathsf{Pr}[\mathcal{A} = 1 \mid G_i] - \mathsf{Pr}[\mathcal{A} = 1 \mid G_j]\big|$. We complete the proof by showing a sequence of indistinguishability claims for the games G_0, G_1, G_2, G_3.

▸ G_0 *is indistinguishable from* $G_{t\text{-priv}}^{\mathcal{A},\mathcal{S},\mathcal{D}}(1^\lambda, n, m, k)$*:* by definition of the the voter privacy/PCR game,

$$\big|\,\mathsf{Pr}[G_{t\text{-priv}}^{\mathcal{A},\mathcal{S},\mathcal{D}}(1^\lambda, n, m, k) = 1] - \mathsf{Pr}[\mathcal{A} = 1 \mid G_0]\big| = 0.$$

▸ G_1 *is indistinguishable from* G_0*:* the probability that the knowledge extractor fails to extract the witnesses is negligible . Upon successful extraction, the view of $\mathcal{A}$ is identical to G_0. Hence, we have $\mathsf{Adv}_{G_0,G_1}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

▸ G_2 *is indistinguishable from* G_1*:* since the simulated NIZK proofs are identical to the real ones, the view of $\mathcal{A}$ is identical to G_1. Hence, we have $\mathsf{Adv}_{G_1,G_2}(\mathcal{A}) = 0$.

▸ G_3 *is indistinguishable from* G_2*:* it is easy to see that the tally ciphertexts will still be decrypted to the correct election result $\langle x_1, \ldots, x_m \rangle$ due to the fake partial decryptions $D_{w,j}$. The simulated NIZK proofs are indistinguishable from the real ones.

We now show that if the adversary $\mathcal{A}$ can distinguish Game G_3 from G_2 then there exists an adversary $\mathcal{B}$ who can win the IND-CPA game of the ElGamal encryption with the same probability.

In the IND-CPA game, $\mathcal{B}$ first receives a public key denoted as (g, h_w) from the IND-CPA challenger, and $\mathcal{B}$ forwards (g, h_w) together with the simulated NIZK to the EA as the partial public key of the trustee T_w in the **Setup** phase. Then $\mathcal{B}$ submits $m_0 = 0, m_1 = 1$ to the IND-CPA challenger, and $\mathcal{B}$ receives $C := (C^{(0)}, C^{(1)})$ that encrypts m_{b^*}, where $b^* \in \{0, 1\}$

is the IND-CPA challenger bit for $\mathcal{B}$ to guess. $\mathcal{B}$ computes

$$\hat{C} := (\hat{C}^{(0)}, \hat{C}^{(1)}) = (C^{(0)}, C^{(1)} \cdot (C^{(0)})^{\sum_{i \neq w} \mathsf{sk}_i}) \ ,$$

which is encryption of m_{b*} under the election public key (g, h). During the **Cast** ceremony, for each uncorrupted voter V_ℓ, $\mathcal{B}$ sets j^*_ℓ to be the index s.t. $\{P_{j^*_\ell}\} = \mathcal{U}^b_\ell$. Then, it generates $m-1$ encryptions of 0, $\{C_{\ell,i}\}_{i \neq j^*_\ell}$ under the election public key (g, h) together with their NIZK. For j^*_ℓ, $\mathcal{B}$ sets C_{ℓ,i^*_ℓ} to be re-encryption of $\hat{C}$, i.e. $C_{\ell,i^*_\ell} = (\hat{C}^{(0)} \cdot g^{r_j}, \hat{C}^{(1)} \cdot h^{r_j})$ for fresh randomness r_j. $\mathcal{B}$ appends necessary simulated NIZK and submits $\{C_{\ell,i}\}_{i \in [m]}$ as the ballot for V_ℓ. Clearly, if C encrypts 0 then the adversary $\mathcal{A}$'s view is the same as Game G_3; otherwise, if C encrypts 1 then the adversary $\mathcal{A}$'s view is the same as Game G_2. Hence, assume $\mathcal{A}$ outputs 1 if she thinks she is in Game G_2 and outputs 0 if she thinks she is in Game G_3. $\mathcal{B}$ forwards $\mathcal{A}$'s outputs, and $\mathcal{B}$ win the IND-CPA game whenever $\mathcal{A}$ guesses correctly. Thus, we have $\mathsf{Adv}_{G_2,G_3}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ElGamal}}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

▸ $\Pr[\mathcal{A} = 1 \mid G_3] = 1/2$*:* since the view of Game G_3 does not depend on the bit b, the adversary's probability of guessing b correctly in G_3 is exactly $1/2$.

By the above claims, the overall advantage of $\mathcal{A}$ is

$$\begin{aligned}
\left| \Pr[G^{\mathcal{A},\mathcal{S},\mathcal{D}}_{t\text{-priv}}(1^\lambda, n, m, k) = 1] - \frac{1}{2} \right| &= \left| \Pr[\Pr[\mathcal{A} = 1 \mid G_0] - \Pr[\mathcal{A} = 1 \mid G_3] \right| \leqslant \\
&\leqslant \sum_{i=1}^{3} \mathsf{Adv}_{G_{i-1},G_i}(\mathcal{A}) = \mathsf{negl}(\lambda) + 0 + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{ElGamal}}(\mathcal{A}) = \\
&= \mathsf{negl}(\lambda),
\end{aligned}$$

which completes the proof. ■

6.6 A MitM Attack Against Helios's Privacy

In Section 6.2, we described the Helios e-voting ceremony. Recall that in our description there is no way for the BB to authenticate the trustees' data (this typically requires a user-side PKI which is hard to deploy in practice) and the trustees are not required to audit the information posted in the BB.

Unfortunately, this oversight might cause subtle privacy problems when the EA is malicious. In order to achieve voter privacy, it is necessary to ensure that at least one of the trustees participates in the election audit. For instance, this is consistent with claims made in the Helios web server material where it is argued that voter privacy is guaranteed unless all the trustees are corrupted, see [61]. Nevertheless, the trustee auditing step is optional and there are no proper instructions regarding the necessity of the trustee verification process. Moreover, the current Helios implementation (Helios v4 [60]) poses difficulties for someone without technical knowledge to do so.

In Subsection 6.6.1, we introduce a generic MitM attack against the voter privacy of any e-voting systems that like Helios (i) builds upon TPKE encryption and (ii) trustee data is not authenticated and the auditing step is not performed. Next, in Subsection 6.6.2, we demonstrate our attack against Helios as a specific instance of the attack methodology. In Subsection 6.6.3, we propose simple countermeasures that deal with this type of attacks.

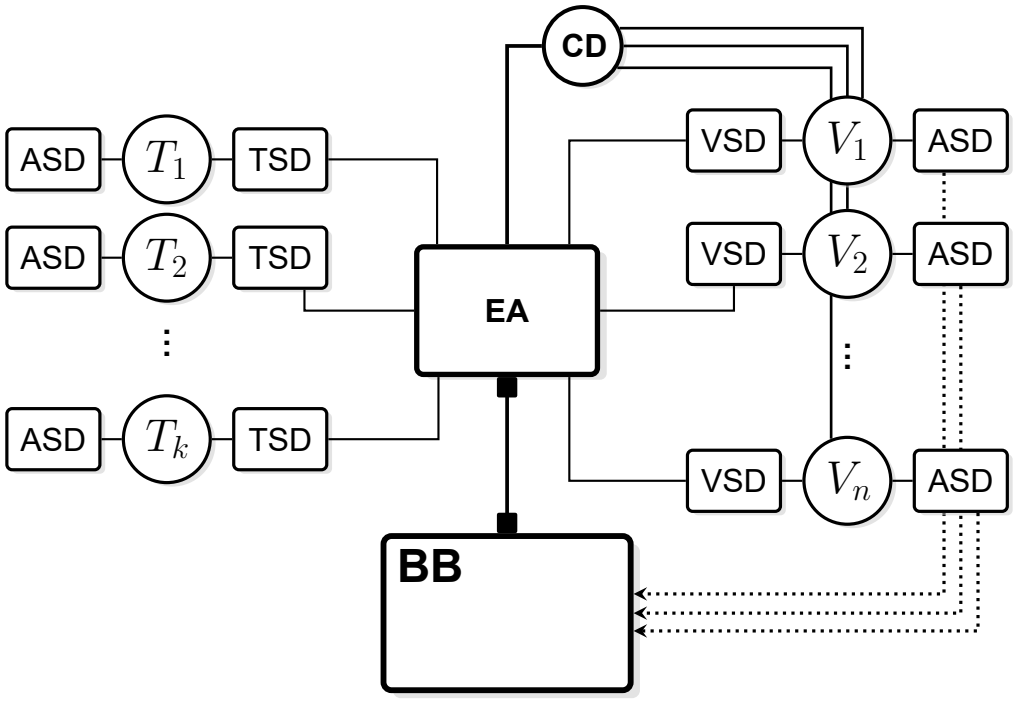


Figure 6.5: The star network topology in the architecture of a typical TPKE-based e-voting system. The dotted lines denote read-only access to the BB.

6.6.1 The system vulnerability and our MitM attacks

Similar to Helios, any typical TPKE-based e-voting system mainly consists of EA, BB, and the supporting devices. The EA additionally handles vote collection, hence for simplicity we absorb EA and VC into a single EA entity. The architecture topology forms a star network as depicted in Figure 6.5; the BB, all the trustees and all the VSDs are connected through the central EA node[3].

Such a network topology is sensible and is followed by systems used in practice since it avoids additional pairwise communication between the entities participating in the election; this has a number of advantages. First of all, it significantly reduces the development and deployment complexity, as the entire e-voting system can be realized by a single

[3]One may think of the EA being part of the BB; we separate the two entities in our work in order to enable the BB to be completely passive; having a passive BB is important in practice since a robust implementation for the BB will distribute the responsibility of maintaining the election transcript to a set of servers which will be required to execute an agreement protocol for each append operation that should be readable by honest parties.

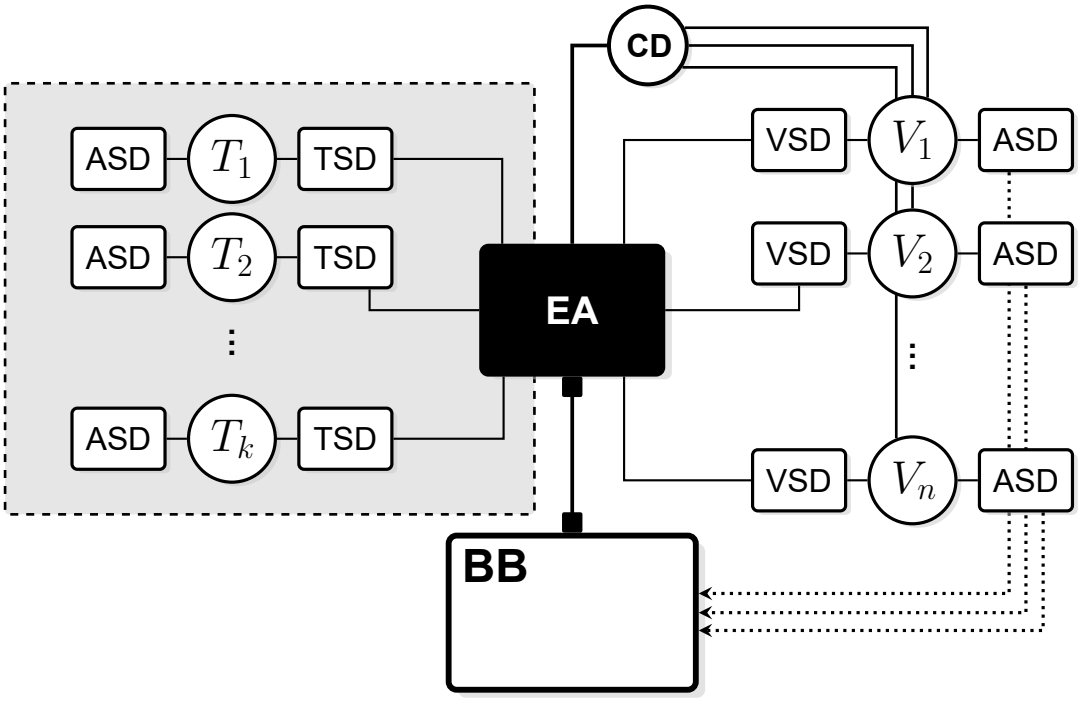


Figure 6.6: A malicious EA acting as MitM against the privacy of a TPKE e-voting system. The trustees $T_1, T_2, \ldots, T_k$ communicate only with the malicious EA, isolated in a fake "gray" environment.

server. Secondly, it is consistent with a reasonable level of usability, as the administrators only need to keep the election server online and all the other election parties are able to participate in the election asynchronously without any coordination.

Unfortunately, the implementation efficiency enjoyed by the aforementioned star network topology does not come without a price; the specific architecture makes the system vulnerable to a class of MitM attacks when the central node (i.e. the election server) is compromised, as depicted in Figure 6.6. Furthermore, the lack of PKI support makes it impossible for a third-party auditor to identify the actual sources of messages that appear in the BB. This problem is recognized in terms of election integrity, and the concept of *individual verifiability* is widely adopted to mitigate this problem by preventing the malicious election server from tampering the submitted ballots. Nevertheless, little attention is given to the contributions of the election trustees even though it is equally important to ensure the integrity of trustees' messages (i.e. the election parameters) in the BB.

Our attack assumes that only the EA is controlled by the adversary, whereas the rest of the TPKE-based e-voting system entities and all the supporting devices remain honest. The steps of the attack are illustrated via corresponding figures.

STEP 1: During the election setup phase, the malicious EA follows the **Setup** protocol description and interacts with the real trustees $T_1, \ldots, T_k$ to jointly generate the real election public parameters **info** and the real voters' credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$. Meanwhile, the malicious EA (conceptually) creates another set of fake trustees, $T_1^*, \ldots, T_k^*$ and generates

fake pairs of keys $(\mathsf{sk}_1^*, \mathsf{pk}_1^*), \ldots, (\mathsf{sk}_k^*, \mathsf{pk}_k^*)$ fake election public parameters **info*** and the fake voters' credentials $\mathsf{cr}_1^*, \ldots, \mathsf{cr}_n^*$ by running the **Setup** protocol with the fake trustees "in its mind", obtaining $\mathsf{sk}_1^*, \ldots, \mathsf{sk}_k^*$. The malicious EA then publishes **info*** (that include $\mathsf{pk}_1^*, \ldots, \mathsf{pk}_k^*$ and $\mathsf{pk}^* \leftarrow \mathsf{TPKE.Combine}(\mathsf{pk}_1, \ldots, \mathsf{pk}_k^*)$) on the BB and thus all the voters will use the fake election parameters during the **Cast** protocol (cf. Figure 6.7).

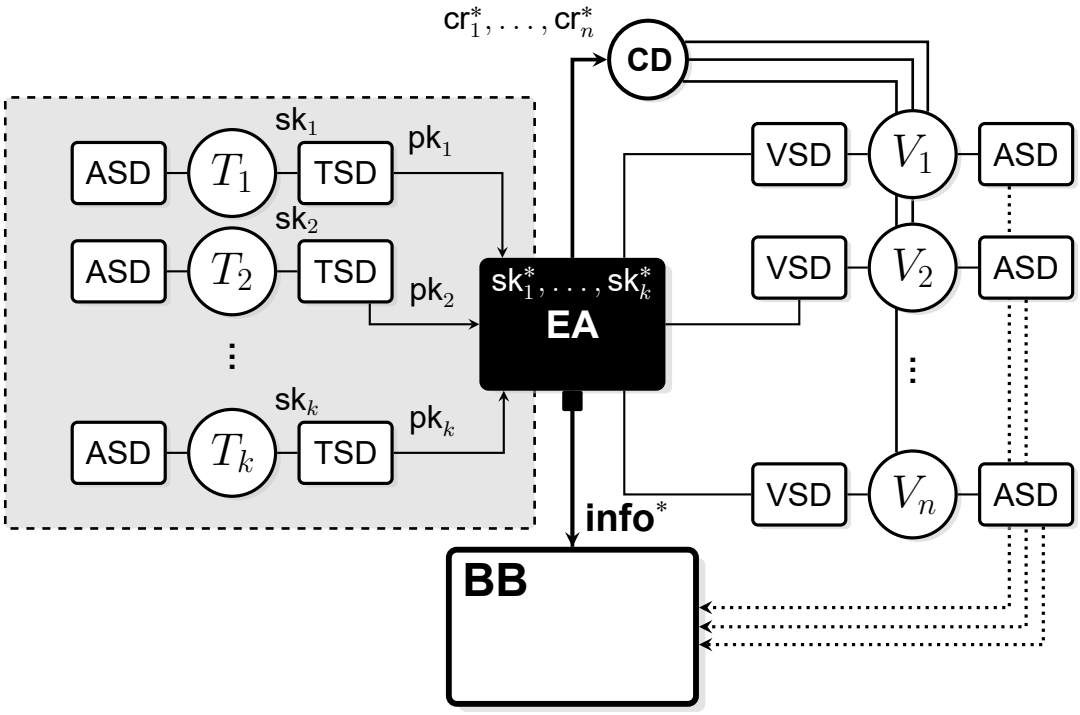


Figure 6.7: MitM attack - STEP 1: Replacement of the trustees' election parameters and setting up of a "fake" election.

STEP 2: The voters $V_1 \ldots, V_n$ read the BB and obtain the fake public key pk^*. They encrypt selections $\mathsf{opt}_{j_1}, \ldots, \mathsf{opt}_{j_n}$ under pk^* and submit their ballots $\mathbf{B}_1^*, \ldots, \mathbf{B}_n^*$ to the malicious EA which, clearly, is able to learn every voter's selection by decrypting the ballots using all fake partial decryption keys $\mathsf{sk}_1^*, \ldots, \mathsf{sk}_k^*$ (cf. Figure 6.8).

STEP 3: The malicious EA can simulate the voting and tally phase of a presumably "real" election run under the real trustees' keys engaging with them in the **Tally** protocol. The purpose of this step is to make the real trustees believe the election tally result is produced by them, whereas EA can simply perform the actual election tally itself and publish the corresponding election result in the BB. It is easy to see that all information in the BB is consistent in the sense that the produced fake election transcript τ^* is publicly verifiable (cf. Figure 6.9).

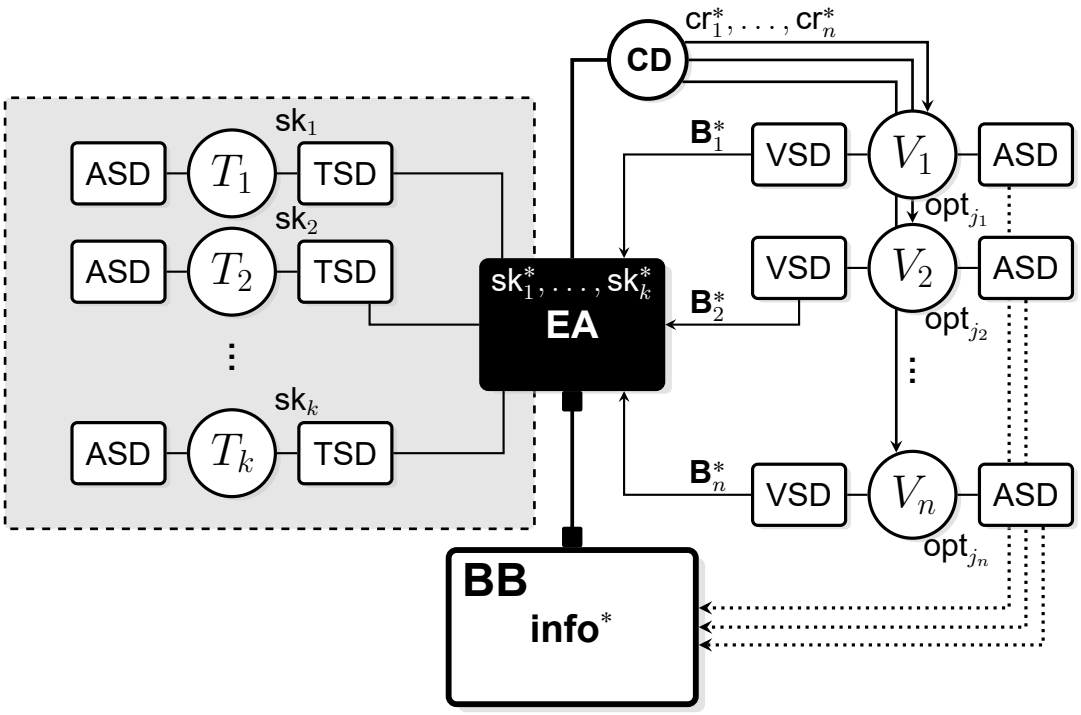


Figure 6.8: MitM attack - STEP 2: Voting under fake election public key and breaching the voters' privacy by decrypting the ballots via $\mathsf{sk}_1^*, \ldots, \mathsf{sk}_k^*$.

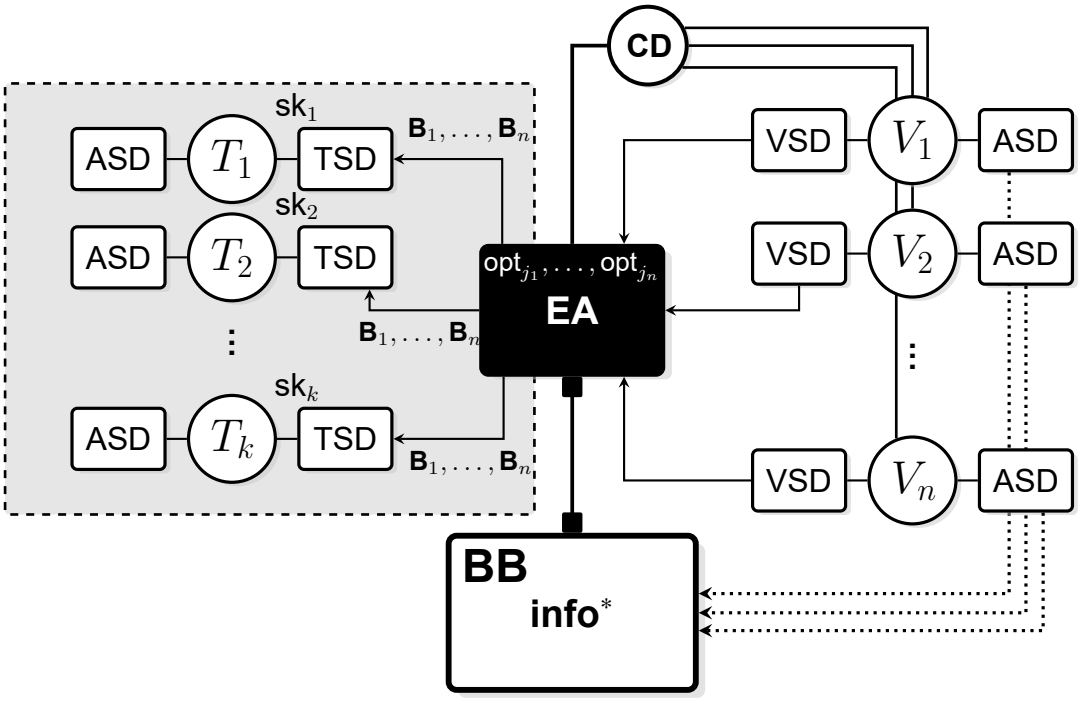


Figure 6.9: MitM attack - STEP 3: Completion of a consistent "real" tally phase under the trustees' election public key.

6.6.2 Instantiation of our MitM attack against Helios

We demonstrate our MitM attack against Helios. However, our attack can apply to all the variants of Helios in the literature with respect to their latest implementations. Our attack does not tamper the javascript code, therefore it is impossible to detect our attack by checking the integrity of the source code as observed at the client side.

Recall that the latest version of Helios v4 uses k out of k threshold (lifted) ElGamal encryption. During the election setup phase, each trustee T_i, $i = 1, \ldots, k$ locally generates a pair of ElGamal partial keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ and sends pk_i to the EA. At this step, a fingerprint (SHA256 hash digest) of the partial public key pk_i is computed and provided to the trustee. It is suggested that the trustees keep the fingerprints of their partial public key and confirm that they are properly stored on the election server. However, there is no further instruction to indicate to the trustees where and how to verify the consistency of this information. Notice that there is no interface for the trustees to verify whether their partial public keys are correctly used to produce the (combined) election public key during the **Setup** ceremony.[4] In fact, only the (combined) election public key is used to generate the election fingerprint (i.e. a SHA256 hash digest of the JSON format of the election definition) after an election is fixed (or "frozen" in Helios terminology). Moreover, the voters are only given the (combined) election public key at the *voting booth* page in the **Cast** ceremony, so it is impossible to check the validity of the partial public keys even if the trustee is also an eligible voter. During the tally phase, the trustees are given their partial public key fingerprints to prevent them from using incorrect partial secret keys. The information displayed on the tally page can never be used for auditing purposes, because every trustee should first identify himself by submitting a unique token to the EA before receiving the content. Hence, the malicious EA can specifically tailor a (inconsistent) view of the tally page for each trustee. Finally, the partial public key information is not even displayed on the bulletin board, which only contains the submitted voters' ciphertexts.

In our MitM attack, the malicious EA receives pk_i from the trustee T_i, $i = 1, \ldots, k$ during the election setup phase. Then, it generates another set of fake partial ElGamal key pairs $(\mathsf{pk}_i^*, \mathsf{sk}_i^*)$ and computes the fake election public key $\mathsf{pk}^* = \prod_{i=1}^{k} \mathsf{pk}_i^*$. When the election is frozen, the malicious EA switches the real election public key with pk^*, so pk^* is used to generate the election fingerprint. In the voting booth, the voters are given pk^* to encrypt their choices, and thus it is consistent with the election fingerprint. In the tally phase, the malicious EA sends the real trustee T_i his partial public key pk_i; therefore, the hash of pk_i matches the fingerprint stored by T_i and the trustee should perform tallying as usual. Once T_i submits the decryption values, the malicious EA mimics the same process with T_i^* (in its "head") and posts the fake decryption factors instead. Clearly, all the information on the BB is publicly verifiable.

Remark 6.3 (*Effectiveness of the MitM attack*)**.** Our MitM attack can be launched against any TPKE-based e-voting system which, like Helios, (i) does not urge that the trustees directly verify that their partial public keys were correctly published and (ii) does not allow

[4] Note that each trustee does not know the other trustees' actual partial public keys.

for any third party to verify the source of the partial public keys (as in [67, 96]). Note that having the EA sign BB data like in [12] does not prevent the attack, since BB consistency is not violated from the EA's point of view (the adversarial public key is posted on the BB by the EA itself). On the contrary, our MitM attack can be prevented when the trustees collaboratively verify their partial public keys either via a verifiable secret sharing scheme [41] or with PKI support [36].

The effectiveness of the attack against various TPKE-based e-voting systems is summarized in Table 6.11.

Table 6.11: Effectiveness of the MitM attack against various TPKE-based e-voting systems.

System	Resistance	Vulnerability
CGS [41]	✓	
Adder [67]		✓
Civitas [36]	✓	
Helios [2]		✓
Zeus [96]		✓
STAR-Vote [12]		✓

6.6.3 Countermeasures

We conclude the presentation of our MitM atack by proposing two countermeasures for TPKE-based e-voting systems without PKI support [67, 2, 96]. Each countermeasure suits a specific threat model for the BB.

1. As shown in Figure 6.5 and commonly used in the e-voting literature [41, 67, 2, 36, 96, 12] as well as in this thesis, the BB is considered to be passive and robust in the sense that posting on the BB is done in an append-only way. In this model, the trustee auditing step should be performed *immediately after* the election setup phase and *before* the voting phase starts. Since the adversary cannot modify the election public key in the BB, it cannot decrypt the encrypted votes without knowing all the partial secret keys. Hence, the voters' privacy is preserved, if at least one of the trustees remains honest.

2. In an alternative threat model, we consider a *covert* adversary [5] (i.e. an adversary that may deviate arbitrarily from the protocol specification, but does not wish to be detected cheating) that may also fully corrupt the BB but cannot link the identity of the auditing party (including both voters and trustees) with the ASD that is used. In this model, the trustee auditing step should be performed *after* the end voting phase and the interaction between the BB and a trustee's ASD should be indistinguishable from an interaction between the BB and any voter's ASD. Observe that the election

public key that has been used for vote encryption is determined w.r.t. a specific voter's ballot tracker. This is because the said public key can be deducted from the statement of any ZK proof of ballot correctness. In order to pass the trustee auditing, the adversary has to post the real public key on the BB whereas in order to pass the voter auditing, it has to post the fake public key. Since the adversary cannot tell whether an auditing party is a trustee or a voter, it will either (i) be discouraged from launching the MitM attack or (ii) eventually be detected .

Future direction: Including MitM in a generalised formal analysis

When an honest EA is not a security requirement, then the MitM attack becomes an immediate threat for any weak auditing behaviour of the trustee nodes. Hence, any typical study of Helios's privacy that extends the threat model in Section 3.6.6 by considering a malicious EA, should incorporate the auditing rate of trustees as expressed via the trustee transducer distribution $\mathbf{D}_p^T$ that we recall here for clarity:

$$\underset{\mathbf{D}_p^T}{\mathsf{Pr}}[M] = \begin{cases} p, & \text{if } M = M_1^T \\ 1-p, & \text{if } M = M_0^T \end{cases}$$

A privacy theorem that generalises the result of Theorem 6.3 is expected to have the following statement.

Theorem. *Assume an election run of Helios with n voters, m candidates and k trustees. Assume that the hash function $H(\cdot)$ considered in Section 6.2 is a random oracle. Let $m, n, k, t \in \mathbb{N}$ be polynomial in λ. Let $\mathcal{D} = \langle \mathbf{D}_1, \ldots, \mathbf{D}_n, \mathbf{D}^{T_1}, \ldots, \mathbf{D}^{T_k}, \mathbf{D}^{\mathsf{CD}} \rangle$ be a transducer distribution vector where $\mathbf{D}^{T_i} = \mathbf{D}_{p_i}^T$, $i = 1, \ldots, k$, is the p_i-biased coin-flip trustee transducer distribution in Eq.* (6.1) *for some $p_i \in [0, 1]$, $\mathbf{D}^{\mathsf{CD}}$ is an arbitrary* CD *transducer distribution and $\mathbf{D}_1, \ldots, \mathbf{D}_n$ are arbitrary voter distributions.*

If the underlying ElGamal encryption scheme is IND-CPA secure, then there exists a view simulator $\mathcal{S}$ s.t. for every PPT adversary $\mathcal{A}$ that controls the EA*, the* VC*, all-but-one trustees and corrupt up to t voters, the distinguishing advantage of $\mathcal{A}$ for voter privacy/passive coercion resistance game for Helios is*

$$\mathsf{max}\{1 - p_i \mid i \in [k]\} + \mathsf{negl}(\lambda) \ .$$

However, such a statement cannot be guaranteed by simply resisting against the MitM attack, as corrupting the EA may yield to other attack opportunities for the adversary that cannot be prevented by trustee auditing alone. We leave the proof of a generalised privacy theorem for future work.

7. SUBSEQUENT WORK

The DEMOS-A and DEMOS-2 e-voting systems enabled for the first time E2E verifiability of the election procedure in the standard model. As a result, the two systems set the bar high for the e-voting security standards. Nonetheless, they do not manage to face all technical challenges that arise in the design of an e-voting constructions. Two noteworthy instances are (i) the proneness of DEMOS-A and DEMOS-2 to single points of failure (the EA, the VC and the BB), a common feature in most existing e-voting systems, and (ii) the vulnerability of both systems against an active coercer. In this chapter, we provide an overview of two follow up e-voting systems, each of them aiming to tackle one of the two main unresolved issues. The first system, named D-DEMOS, is already implemented by Chondros *et al.* [35] and addresses the fault tolerance problem. The second system, designed by Kiayias, Teague, Zacharias and Zikas, is at a near publication status, promising incoercibility and E2E verifiability under minimum assumptions.

7.1 The D-DEMOS e-Voting System

As already mentioned, in the system architecture of DEMOS-A and DEMOS-2, the EA, the VC and the BB are single points of failure. In most real-world cases, system liveness is a top priority, setting election fault tolerance as a critical design issue. Therefore, an e-voting construction applicable in a fully distributed setting is most desirable for spreading the use E2E verifiable e-voting at a national level. This is done in the work of Chondros *et al.* [35] that introduces *D-DEMOS*, a code-voting e-voting system that builds upon DEMOS-A, redesigning VC and BB as distributed subsystems with fault tolerance of less than one third and less than a half of the total VC and BB nodes respectively. In addition, the trustees form a subsystem that operates under a f_k-out-of-k fault threshold. By integrating novel ad-hoc constructions and protocols with DEMOS-A, D-DEMOS achieves the following features:

1. It removes any single point of failure at the on line voting and tally phase, assuming only a concentrated election initialisation authority at setup, that can be removed after completing its role.

2. It supports a distributed fully asynchronous mechanism for the generation of *receipt code* by the VC subsystem that allows the voter to check on-line that their vote was recorded-as-cast (note that in DEMOS-A verification runs after election end, thus voters cannot complain while engaged in the **Cast** protocol in case of malicious behaviour). Every valid vote-code is associated with a unique receipt in every ballot, thus an honest voter can check the consistency of the obtained receipt, simply by comparing it with the respective one in her ballot.

3. It achieves liveness w.r.t. receipt code generation process, along with the safety guarantee that the vote of any honest voter who obtained a valid receipt will be

included in the tally.

4. It preserves the security properties of DEMOS-A, though E2E verifiability is now in the standard model against computationally bounded adversaries.

The contribution of this author to the design of D-DEMOS focuses on the security analysis of the system. In the rest of the section, we provide the proof ideas for each of D-DEMOS's properties (liveness, safety, E2E verifiability, voter privacy/PCR) and, thus, intuition about its security.

Liveness :

Theorem. *Let δ be an upper bound on the communication delay and Δ be an upper bound on the synchronization loss in all node's clocks with respect to a global clock. Let T_{comp} be the worst-case running time of any procedure run by the* VC *nodes and the voters during the voting protocol. Assume that f_{v} out-of the N_{v} total* VC *nodes are honest, where $f_{\text{v}} < N_{\text{v}}/3$. Then, every honest voter that is engaged in the voting protocol at least $(f_{\text{v}} + 1) \cdot \big((2N_{\text{v}} + 5)T_{\text{comp}} + 12\Delta + 6\delta\big)$ clock steps before election end, will obtain a valid receipt.*

Proof strategy overview. If an honest voter submits her vote to an honest VC node, then by the description of the VC nodes consensus protocol and the bounds $\delta, \Delta, T_{\text{comp}}$, we can show that the upper bound on the time required for the honest voter to obtain and verify the validity of her receipt is $T_{\text{wait}} = (2N_{\text{v}}+5)T_{\text{comp}}+12\Delta+6\delta$. Thus, after T_{wait} steps, she will blacklist this VC node and submit the same vote to another randomly selected VC node. By the VC fault tolerance threshold, she will run into a honest VC node after at most $f_{\text{v}}+1$ attempts.

Safety :

The safety theorem is stated in the form of a contract adhered by the VC subsystem.

Theorem. *Assume that f_{v} out-of the N_{v} total* VC *nodes are honest, where $f_{\text{v}} < N_{\text{v}}/3$. Then, any honest voter who receives a valid receipt from a* VC *node, is assured her vote will be published on the honest* BB *nodes and included in the election tally, with probability at least $1 - \text{negl}(\lambda) - \frac{f_{\text{v}}}{2^{64}-f_{\text{v}}}$.*

Proof strategy overview. Assume an adversary that attempts to produce a valid receipt without interacting with the honest VC nodes by either (i) forging digital signatures used in producing certificates for valid vote-codes during vote collection, or (ii) guessing the randomly generated valid 64-bit receipt for some honest voter. By the security of digital signatures, (i) happens only with $\text{negl}(\lambda)$ probability. Further, since there are at most f_{v} malicious VC nodes, the adversary has at most f_{v} attempts (there are $2^{64} - i$ choices left after i attempts) to guess the receipt for each voter, thus (ii) happens with probability no more than

$$\sum_{i=0}^{f_{\text{v}}-1} \frac{1}{2^{64}-i} \leqslant \frac{f_{\text{v}}}{2^{64}-f_{\text{v}}} \ .$$

Now, let V be an honest voter that has obtained a receipt reconstructed from a complete VC interaction. Then, by the security of the underlying digital signature scheme, every honest VC node will submit V's unique cast vote-code to each BB node by including it in the set of voted tuples. Given the fault tolerance thresholds, the majority of honest BB nodes will publish V's vote, while the f_k out-of k honest trustees will read V's vote from the majority of BB nodes and include it in the election tally.

E2E verifiability :

We require fault tolerance only for the BB nodes to guarantee BB consistency and show the E2E verifiability of D-DEMOS in the following theorem.

Theorem. *Let θ be the number of honest voters. Let $\mathcal{A}$ be an adversary that controls the* EA*, all the* VC *nodes, all the trustee nodes and can statically corrupt less than half of the* BB *nodes. Then, the probability that $\mathcal{A}$ causes tally deviation δ from the intended election result without being detected, is no more than $2^{-\theta} + 2^{-\delta}$.*

Proof strategy overview. The proof follows the lines of Theorem 4.4. Specifically, by the number of honest voters, the entropy of the collected voters' coins is at least θ. Similar to Subsections 4.2.4 and 4.2.5, we can show that the verification of the Σ zero-knowledge proofs used in D-DEMOS (Chaum-Pedersen proofs) guarantees the correctness of all the committed ballots in the BB, except some probability error $2^{-\theta}$. In case of all valid zero-knowledge proofs, $\mathcal{A}$ may attack by pointing the honest voter to audit in a BB location where the audit data is inconsistent with the respective information in at least one part of the voter's ballot. As in 4.4, we can show that every such single attack has 1/2 success probability (the voter had chosen to vote with the inconsistent ballot part) and in case of success, adds 1 to the tally deviation. Thus, in this case, the probability that $\mathcal{A}$ causes tally deviation d is no more than $2^{-\delta}$.

Voter privacy/PCR :

Theorem. *Let c, c' be constants s.t. $c' \in (0, c)$ and $n^2(n+1)^m \cdot 2^t = O(2^{\lambda^{c'}})$. Let $\mathcal{A}$ be an adversary that controls all the* VC *nodes, less than half of the* BB *nodes, up to f_k out-of k trustees, and up to t voters, observes the network during election and obtains all the voters' audit information. Then, $\mathcal{A}$ cannot break voter privacy if the underlying commitment scheme is hiding against all 2^{λ^c} adversaries.*

Proof strategy overview. The proof follows the lines of Theorem 4.5. Due to full VC corruption, $\mathcal{A}$ learns all the vote-codes. Even so, the audit information of every voter leaks nothing about her vote, as each ballot part is independently and randomly generated, and the voter could "lie" about her used ballot part (i.e. switch the vote-code and option correspondence in the used ballot part, so that the submitted vote-code appears associated with the option in the alternative the voter did not follow). Moreover, we can show that if $\mathcal{A}$ distinguishes the alternative followed by honest voters, then we can construct an algorithm $\mathcal{B}$ that invokes $\mathcal{A}$ and simulates an election execution where it guesses (i)

the corrupted voters' coins (in 2^t expected attempts) and (ii) the election tally (in $(n+1)^m$ expected attempts). Thus, $\mathcal{B}$ finishes a compete simulation with high probability running in $n^2(n+1)^m \cdot 2^t = O(2^{\lambda^{c'}})$ steps. By exploiting the distinguishing advantage of $\mathcal{A}$, $\mathcal{B}$ can break the hiding property of the option-encoding commitment scheme in $O(2^{\lambda^{c'}}) = o(2^{\lambda^c})$ steps, thus leading to contradiction.

7.2 Towards Fully Coercion Resistant and End-to-end Verifiable e-Voting

In order to design an extension of DEMOS-A that achieves resistance even against an active coercer, Kiayias, Teague, Zacharias, and Zikas deploy the UC incoercibility framework in [4] summarised in Section 3.7. For completeness, we recall the incoercibility conditions for [4] security:

(i). Any election environment $\mathcal{Z}$ cannot distinguish an ideal world coercion settng from a real world coercion setting.

(ii). Any election environment $\mathcal{Z}$ cannot distinguish an ideal world evasion settng from a real world evasion setting.

We are interested in incoercibility without putting trust to supporting devices. This rules out any client-side encryption e-voting system like DEMOS-2, where the clients must remain honest for privacy. Unfortunately, under this model, neither DEMOS-A succeeds in providing security against a coercer that does not aim at an individual level, but at a statistical effect on the will of the electorate. An example of such an attacker was pointed out by Teague, which we cite below.

An active coercer against DEMOS-A:

The idea is that the coercer demands a vote of a certain form and attacks in a statistical rather sn individual manner. Some voters are unable to satisfy the coercer, others are able to satisfy the coercer and still vote any way they choose, but some fraction of voters are able to produce a receipt of the right form only by obeying the coercer. Some are able to do so only by *disobeying* the coercer, but as long as there are fewer of them than those who must obey, the attack is successful.

101	
Vote-code	**Option**
27935	YES
75218	NO
84439	YES
77396	NO

102	
Vote-code	**Option**
58729	YES
45343	NO
14582	YES
93484	NO

103	
Vote-code	**Option**
52658	YES
65864	NO
84373	YES
49251	NO

It's easiest to understand by looking at the toy example in Section 4.4, repeated above. Let's assume that vote-codes codes run from 0 to 99999. Note that the attack does not

depend on the vote-codes having high entropy—it would also be possible for example if the codes were taken from $\{0, 1\}$. It does depend on the vote-codes being ordered independently of the options on the ballot.

Suppose the coercer programme instructs:

> "*Look at the double ballot presented to you. Instead of choosing randomly which one to open, try to produce a receipt and vote as follows:*
>
> - *On the opened ballot part, the* 'NO' *vote-code should be at least 50000.*
> - *Your voting code (from the other ballot part) should be at least 50000.*"

Now consider the 3 examples given there:

- 101 can satisfy the coercer by voting either 'YES' or 'NO'. (Opening the top ballot.)
- 102 can satisfy the coercer only by voting 'YES'. (Opening the bottom ballot.)
- 103 can satisfy the coercer only by voting 'YES'. (Opening the top ballot.)

In general, it holds that

1. If neither 'NO' vote-code is $\geqslant 50000$, then the voter cannot satisfy the coercer. (1/4 of ballot pairs).
2. If exactly one 'NO' vote-code is $\geqslant 50000$, then
   (a) If the other ballot's 'YES' vote-code is $\geqslant 50000$, then the voter can satisfy the coercer only by voting 'YES' (1/4 of ballot pairs).
   (b) If the other ballot's 'YES' vote-code is < 50000, then the voter cannot satisfy the coercer (1/4 of ballot pairs).
3. If both 'NO' vote-codes are $\geqslant 50000$, then
   (a) If at least one 'YES' vote-code is $\geqslant 50000$, then the voter can satisfy the coercer either way. (3/16 of ballot pairs).
   (b) If both 'YES' codes are < 50000, then the voter can satisfy the coercer only by voting 'NO' (1/16 of ballot pairs)

The crucial point is that there are more voters who can satisfy the coercer only by voting 'YES' than there are who are forced to vote 'NO'. By a statistical argument, the coercer will bias the election result towards 'YES' .

Making DEMOS-A incoercible:

The key idea for building a fully coercion resistant DEMOS system is quite simple, however proving its security appears highly non-trivial. The incoercibility mechanism consists of the following two component steps:

1. Provide each voter with a double ballot that consists of K copies of original DEMOS-A ballots, where K is a fixed system parameter. Namely, for options $\mathcal{O} = \{\text{opt}_1, \ldots, \text{opt}_m\}$, a ballot $\mathbf{B} = \left(\mathbf{B}^{(0)}, \mathbf{B}^{(1)}\right)$ with K vote-codes per option per ballot part is represented as the Km-tuple

$$\langle \text{tag}, (c_1^0, \ldots, c_m^0), \ldots, (c_{(r-1)m+1}^0, \ldots, c_{Km}^0), \ldots, (c_1^1, \ldots, c_m^1), \ldots, (c_{(K-1)m+1}^1, \ldots, c_{Km}^1) \rangle$$

   where $c_{(r-1)m+j}^a$ is the r-th vote-code for opt_j in ballot part $\mathbf{B}^{(a)}$.

2. Design an evasion strategy run by the voters that input the code of any coercion programme $\mathcal{A}$, the voter's ballot $\mathbf{B}$ and her intended option opt, either (i) outputs a transformation $\mathbf{B}'$ of $\mathbf{B}$ s.t. when $\mathcal{A}$ sees $\mathbf{B}'$ instructs the voter to cast a vote-code for opt or (ii) aborts, so the voter is coerced. The candidate evasion strategy performs iterations of vote-code permutations and replacements checking if (i) is reached. The number of operations depends on a fixed system parameter L.

The above solution is shown successful against coercers that do not force the voter to abort, i.e. the coercer's instructions always lead to vote submission. Against this meaningful class of attackers, the following security theorem holds, here stated at a not overly formal level.

Theorem. *There is a polynomial $p(\cdot)$ s.t. for every coercer $\mathcal{A}$ and for sufficiently large vote-code size and system parameters K,L at least $1 - \frac{1}{p(K)}$ voters execute the evasion strategy successfully with $1 - \text{negl}(L)$ probability.*

The $\frac{1}{\text{poly}(K)}$ rate of coerced voters seems optimal for this approach, as there is a simple counterexample of a coercer (the one that always requests to see the largest code in a ballot part), where it is impossible to achieve $1 - \text{negl}(K)$ voters avoiding coercion with $1 - \text{negl}(L)$ probability. In addition, the extended new system satisfies the first condition of [4] security, i.e., indistinguishability between an ideal world coercion setting from a real world coercion setting. However, several important technical obstacles need to be overcome, so that the new system is proven incoercible.

8. CONCLUSIONS AND FUTURE WORK

The completion of this PhD thesis concludes an extended formal cryptographic argumentation on the boundaries of optimal E2E verifiability and the relation of e-voting security with human auditing behaviour. The introduction of the DEMOS family initialised to the pair of DEMOS-A and DEMOS-2 e-voting systems answers affirmatively to question **Q1** in Section 1.4, promising election executions where the integrity of the result is proven under the standard model, i.e. without trusting a source of randomness or any other setup assumption. In addition, the honesty of no election administrator or voting supporting device is required.

Despite its apparent strength, the framework in Section 3.3 assumes the consistency of the BB view, which is the remaining gap from claiming E2E verifiability *unconditionally*. We argue that a consistent BB can be easily seen to be a tight condition since without it, it is easy to verify that E2E verifiability of the election cannot be achieved: by controlling the BB, an adversarial EA can distribute voters to their own separate "islands" where within each one the voters will have their own verifiable view of an election result that can be in reality completely skewed. The latter is in agreement with the compromise for tolerating the possibility of *independent execution attacks* in networks without any authentication mechanism discussed by Barak et al. [6]. Below, we cite a relevant excerpt from this work.

> "*In the case of multi-party protocols, an adversary can always partition the honest parties into disjoint subsets. Then, given this partition, the adversary can run separate (and independent) executions with each subset in the partition, where in an execution with a given subset of honest parties $\mathcal{H}$, the adversary plays the roles of all the parties outside of $\mathcal{H}$.* "

Implementing a consistent BB is beyond the scope of this thesis, yet we remark that it can be realised generically via a blockchain primitive (transaction ledger). An alternative solution that assumes fault tolerance for a distributed BB is considered in the D-DEMOS e-voting system discussed in Section 7.1.

Despite the fact that top-tier integrity is a main objective of this thesis, it could not be meaningful without being combined with solid secrecy guarantees. To illustrate this point, imagine an election where privacy was of no concern[1]. Then, a simple open ballot voting procedure, should obviously satisfy Definition 3.2. What makes our E2E verifiability results powerful is that it are proven in line with the voter privacy of DEMOS-A and DEMOS-2 against a passive coercer. Clearly, achieving this full level of integrity against an active coercer would be an significant step towards optimal e-voting system security. Concurrent work towards this direction is discussed in Section 7.2.

As far as studying human behaviour is concerned, this thesis has set the necessary cryptographic background and its mathematically argued results on this matter raise intriguing

[1] There are several such valid voting cases, e.g. in Parliament sessions.

issues. The security analysis of the widely used Helios e-voting system pointed out its weaknesses, in cases where humans do not audit at an acceptable rate. Unfortunately, it appears that this is the case even at highly risk aware electorates such as cryptographers (IACR elections). It appears that Helios is secure only with respect to an ideally trained electorate, whereas its integrity can be breached under easily predictable voter behaviour. Besides, even though resistance against the MitM attack presented in Section 6.6 can be dealt by encouraging trustee auditing, the verifiability weakness is inherent in Helios, hence unavoidable by design.

Our analysis leads to a debate that, beyond its technical basis, can be viewed from a rather political and philosophical lens; if human behaviour, even within protocol specification, *can affect* the security of an e-voting system, then specifying explicitly the extent of the security risks -thus answering question **Q2** in Section 1.4- becomes a top priority. Can these risks be mitigated by significantly better systems, or do they set a security guarantee upper bound, as price for moving responsibility directly to the voters? In order to ask for end-to-end verifiable security, is people's proper training a prerequisite? More generally stated,

Is political maturity an inevitable trade-off for
provenly secure direct democratic procedures?

The robust ceremony model of this thesis could be the means for translating these questions into strict mathematical language. Further, the syntax and definitions in Section 3.6 can be enhanced with new model parameters, such as post-election complaint rate and transducers selection according to statistics over a closed electorate, thus taking history into account (rather than sampling independently from transducer distributions).

Based on the aforementioned discussion, we believe that the findings of this thesis can be a valuable asset for subsequent research. In the following paragraphs, we share our thoughts on problems that we evaluate as challenging directions for future work.

E2E verifiable multi-party computation in the standard model:

A complete e-voting execution (setup, voting and tally phase) is a prominent special case of a *multi-party computation (MPC)* protocol [53, 28] supporting verification, where the computation is w.r.t. the election evaluation function f. In the existing verifiable MPC constructions [9], verifiability held under setup assumptions (e.g. trusted CRS generation and RO) and /or requiring the client perform cryptographic operations. Achieving MPC with E2E verifiability in the standard model according to the standards in this thesis, is an interesting open problem, let alone if it is done with minimum computational requirements. We are confident that the core techniques used in DEMOS-A and DEMOS-2, i.e. randomness extraction from the entropy inserted by the votes (clients) and code-voting-wise setup phase, can be the building blocks for a construction with such features.

Accountable elections in the standard model:

DEMOS systems have been designed respecting the significance of an e-voting system to allow the voters and any third party to be convinced of the election result without relying

in good faith w.r.t. the honesty of the election administrators. Supporting E2E verifiability is a most challenging technical problem in the design of any voting system, and there is only a limited number of systems that can claim they enjoy this feature.

Nevertheless, one may argue that E2E verifiability alone is not the ultimate goal of formalising security from the integrity aspect. In an election execution, it is meaningful to enable an honest complaining voter to prove the invalidity of the procedure w.r.t. their view in front of a *judicial authority* responsible for resolving disputes. On the other hand, an honest collection of election administrators should be able to defend themselves against a malicious complaining voter. The property that captures the ability of an e-voting system to manage accusations among involved parties indisputably is called *accountability* and was formally defined by Küsters, Truderung and Vogt [70], similarly to verifiability, i.e. parameterised by a "global accountability" goal and an adversarial environment.

Given an explicit goal and in all-malicious environment, as it is done in our E2E verifiability model, the dispute is between the voter and the election system. Under such a framework, a straightforward twist that results in an DEMOS system that supports accountability, is to enable the voter to convince the judicial authority of the originality of her ballot, by applying digital signatures for the ballots. However, this would lead to the loss of the PCR property that DEMOS-A and DEMOS-2 achieve, as now the voters cannnot lie about their view. Furthermore, under the [70] framework, the component entities of the election system (EA, VC, trustees and VSDs) must be allowed to defend their individual honesty. Hence, the construction of a DEMOS-based system with the most complex possible level of accountability in the standard model would be a non-trivial research direction.

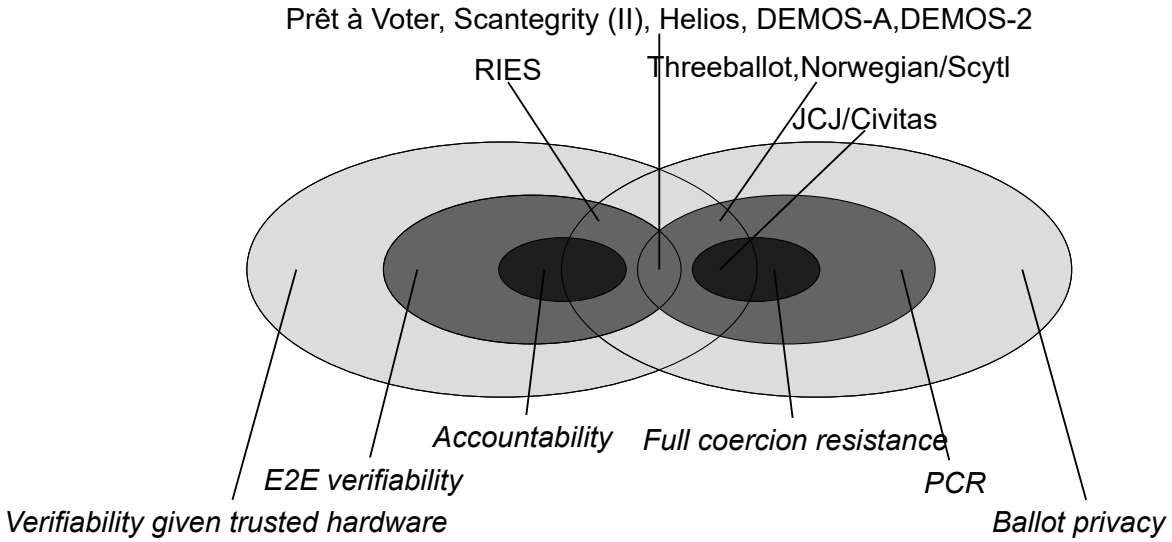
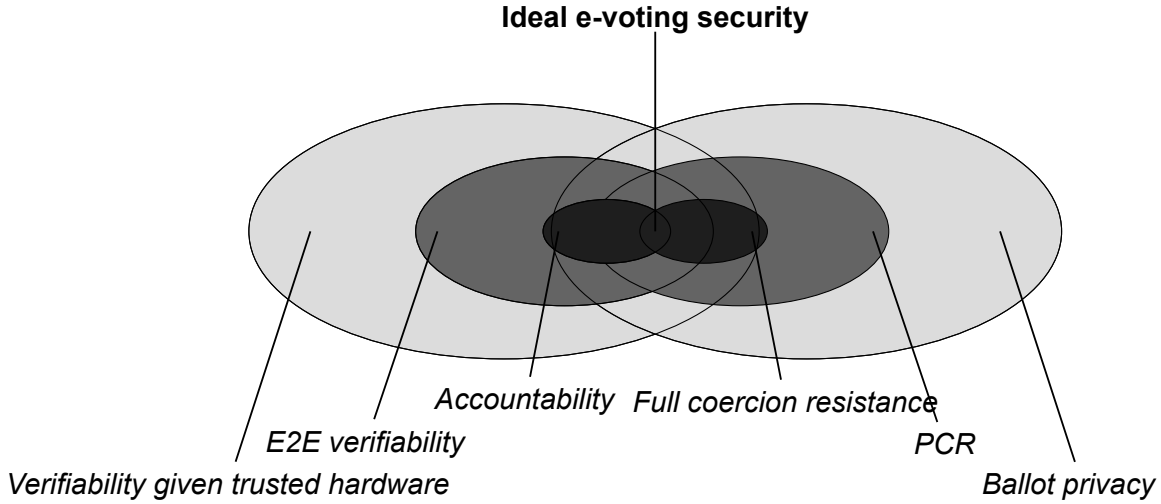


Figure 8.1: The integrity and privacy status of several well-known e-voting systems.

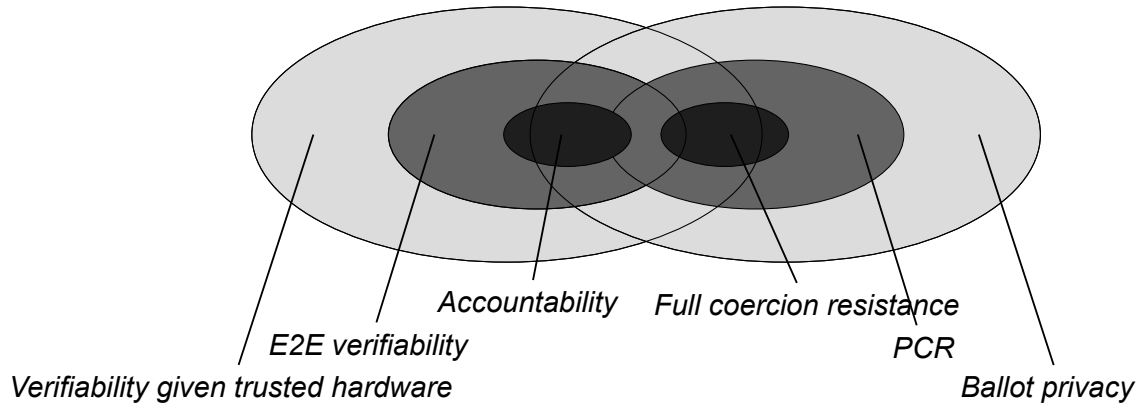
The boundaries of optimal e-voting security:

As already discussed in the thesis's introduction (cf. Chapter 1), integrity and secrecy in voting are in contradictory relation. From an integrity point of view, accountability is the highest goal. Clearly, full coercion resistance is the counterpart for secrecy. Given this fact, the following question arises:

> *Can accountability and full coercion resistance be satisfied concurrently? If not, what is the maximum level of secrecy (resp. integrity) expected in a voting system that satisfies accountability (resp. full coercion resistance)*?



(a) Feasibility of an ideally secure e-voting system.



(b) Incompatibility of accountability and full coercion resistance.

Figure 8.2: The possible scenarios for the boundaries of optimal e-voting security.

Answering this fundamental question would provide the e-voting researcher with a clear design strategy regarding optimal security. For example, if the answer is negative and the construction in Section 7.2 is proven E2E verifiable *and* fully coercion resistant, then it is the best we can hope for given the requirement for top-tier secrecy. Evidence on this question has been given formally by Chevallier-Mames *et al.* [33], where they show the impossibility of unconditional verifiability and privacy, however in more abstract and non-cryptographic framework. In Figure 8.1, we illustrate the security status of several modern e-voting systems. In Figure 8.2, we provide a portrait of two main possible scenarios of optimal security. In the first scenario, (cf. Figure 8.2(a)), an ideal system that combines the best of two worlds is feasible. In the second scenario (cf. Figure 8.2.(b)), such an system is proven impossible. Therefore, enhancing DEMOS-A or DEMOS-2 with accountability (resp. full coercion resistance) would reach optimal security given maximum integrity (resp. secrecy).

Optimising the efficiency of DEMOS-A:

As any code-voting system, DEMOS-A, or its distributed D-DEMOS version (cf. Section 7.2) has many advantages regarding voting device requirements; the voter's client can be of the minimum computational power specification and even so, can be corrupted throughout the whole election period without privacy being compromised. As long as the EA is destroyed after setup or it becomes distributed (e.g. via a secure multi-party protocol), DEMOS-A is E2E verifiable and private under minimum assumptions. We are positive that DEMOS-A can promise secure elections in many real-world settings, and its main limitation is due to the restricted scalability it now offers.Therefore, improving the cryptographic elements in the core of DEMOS-A (or D-DEMOS), so that the demanding election preparation step is at a satisfactory level for national scale executions, e.g. order of millions of voters and order of hundreds of options, is a research goal that would boost the system's usability and public acceptance.

ABBREVIATIONS - ACRONYMS

ASD	Auditing Supporting Device
BB	Bulletin Board
$\mathcal{BPP}$	Bounded-error Probabilistic Polynomial time
CD	Credential Distributor
DDH	Decisional Diffie-Hellman
DLOG	Discrete LOGarithm
EA	Election Authority
e-voting	electronic voting
HVZK	Honest-Verifier Zero-knowledge
iff	if and only if
IND-CPA	INDistinguishability under Chosen Plaintext Attacks
IPS	Interactive Proof System
ITM	Interactive Turing Machine
MitM	Man-in-the-Middle
MPC	Multi-Party Computation
NI	Non-Interactive
$\mathcal{NP}$	Non-deterministic Polynomial time
PCR	Passive Coercion Resistance
PKE	Public-key Encryption
PKI	Public-key Infrastructure
PoK	Proof of Knowledge
PPT	Probabilistic Polynomial Time
PT	Polynomial Time
RO	Random Oracle
r.v.	random variable
SSS	Secret Sharing Scheme

s.t.	such that
TM	Turing Machine
u.a.r.	uniformly at random
UC	Universal Composability
VC	Vote Collector
$\mathcal{VC}$	e-Voting Ceremony
$\mathcal{VS}$	e-Voting System
VSD	Voting Supporting Device
w.l.o.g.	without loss of generality
w.r.t.	with respect to

REFERENCES

[1] *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013*. USENIX Association, 2013.

[2] Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348. USENIX Association, 2008.

[3] Alfred V. Aho, editor. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. ACM, 1987.

[4] Joël Alwen, Rafail Ostrovsky, Hong-Sheng Zhou, and Vassilis Zikas. Incoercible multi-party computation and universally composable receipt-free voting. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 763–780. Springer, 2015.

[5] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *J. Cryptology*, 23(2):281–343, 2010.

[6] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377. Springer, 2005.

[7] Boaz Barak, Guy Kindler, Ronen Shaltiel, Benny Sudakov, and Avi Wigderson. Simulating independence: New constructions of condensers, ramsey graphs, dispersers, and extractors. *J. ACM*, 57(4), 2010.

[8] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[9] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 175–196. Springer, 2014.

[10] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73. ACM, 1993.

[11] Josh Benaloh. Simple verifiable elections. In Dan S. Wallach and Ronald L. Rivest, editors, *2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06, Vancouver, BC, Canada, August 1, 2006*. USENIX Association, 2006.

[12] Josh Benaloh, Michael D. Byrne, Bryce Eakin, Philip T. Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. STAR-vote: A secure, transparent, auditable, and reliable voting system. In *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013* [1].

[13] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 544–553. ACM, 1994.

[14] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In Vijay Atluri and Claudia Díaz, editors, *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, volume 6879 of *Lecture Notes in Computer Science*, pages 335–354. Springer, 2011.

[15] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643. Springer, 2012.

[16] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In Simon [92], pages 103–112.

[17] Bootstrap. Twitter Bootstrap. `http://getbootstrap.com/`, 2013.

[18] Ran Canetti. Security and composition of multi-party cryptographic protocols. *IACR Cryptology ePrint Archive*, 1998:18, 1998.

[19] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[20] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation (extended abstract). In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 504–513. IEEE Computer Society, 1996.

[21] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 209–218. ACM, 1998.

[22] Angelo De Caro and Vincenzo Iovino. jPBC: Java pairing based cryptography. In *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Corfu, Greece, June 28 - July 1, 2011*, pages 850–855. IEEE Computer Society, 2011.

[23] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[24] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 177–182. Springer, 1988.

[25] David Chaum. Surevote: Technical overview. In *Proceedings of the Workshop on Trustworthy Elections*, WOTE, 2001.

[26] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.

[27] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE TIFS*, 4(4):611–627, 2009.

[28] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Simon [92], pages 11–19.

[29] David Chaum, Aleksander Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan T. Sherman, and Poorvi L. Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy*, 6(3):40–46, 2008.

[30] David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors. *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*. Springer, 2010.

[31] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[32] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical voter-verifiable election scheme. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[33] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties of voting schemes. In Chaum et al. [30], pages 191–199.

[34] Nikos Chondros, Alex Delis, Dina Gavatha, Aggelos Kiayias, Charalampos Koutalakis, Ilias Nicolacopoulos, Lampros Paschos, Mema Roussopoulos, Giorge Sotirelis, Panos Stathopoulos, Pavlos Vasilopoulos, Thomas Zacharias, Bingsheng Zhang, and Fotis Zygoulis. Electronic voting systems - from theory to implementation. In Alexander B. Sideridis, Zoe Kardasiadou, Constantine P. Yialouris, and Vasilios Zorkadis, editors, *E-Democracy, Security, Privacy and Trust in a Digital World - 5th International Conference, E-Democracy 2013, Athens, Greece, December 5-6, 2013, Revised Selected Papers*, volume 441 of *Communications in Computer and Information Science*, pages 113–122. Springer, 2013.

[35] Nikos Chondros, Bingsheng Zhang, Thomas Zacharias, Panos Diamantopoulos, Stathis Maneas, Christos Patsonakis, Alex Delis, Aggelos Kiayias, and Mema Roussopoulos. D-DEMOS: A distributed, end-to-end verifiable, internet voting system. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*, pages 711–720. IEEE Computer Society, 2016.

[36] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 354–368. IEEE Computer Society, 2008.

[37] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 372–382. IEEE Computer Society, 1985.

[38] United States Election Assistance Commission. Voluntary voting systems guidelines, 2005.

[39] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *IACR Cryptology ePrint Archive*, 2010:625, 2010.

[40] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[41] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[42] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[43] Cynthia Dwork, editor. *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*. Springer, 2006.

[44] Carl M. Ellison. Ceremony design and analysis. *IACR Cryptology ePrint Archive*, 2007:399, 2007.

[45] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426. ACM, 1990.

[46] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[47] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[48] David Galindo and Srinivas Vivek. A practical leakage-resilient signature scheme in the generic group model. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2012.

[49] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22,*

1984, Proceedings, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[50] Kristian Gjøsteen. Analysis of an internet voting protocol. *IACR Cryptology ePrint Archive*, 2010:380, 2010.

[51] Kristian Gjøsteen. The norwegian internet voting protocol. *IACR Cryptology ePrint Archive*, 2013:473, 2013.

[52] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[53] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Aho [3], pages 218–229.

[54] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[55] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985.

[56] Rop Gonggrijp, Willem-Jan Hengeveld, Eelco Hotting, Sebastian Schmidt, and Frederik Weidemann. RIES - Rijnland Internet Election System: A cursory study of published source code. In Peter Y. A. Ryan and Berry Schoenmakers, editors, *E-Voting and Identity, Second International Conference, VOTE-ID 2009, Luxembourg, September 7-8, 2009. Proceedings*, volume 5767 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2009.

[57] Jens Groth. Evaluating security of voting schemes in the universal composability framework. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, volume 3089 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2004.

[58] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Dwork [43], pages 97–111.

[59] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.

[60] Helios. Helios github repository. `https://github.com/benadida/helios-server`. Last accessed: 2014-07-31.

[61] Helios. Helios privacy claims. `https://vote.heliosvoting.org/privacy`. Last accessed: 2014-07-31.

[62] Engelbert Hubbers, Bart Jacobs, and Wolter Pieters. RIES - internet voting in action. In *29th Annual International Computer Software and Applications Conference, COMPSAC 2005, Edinburgh, Scotland, UK, July 25-28, 2005. Volume 1*, pages 417–424. IEEE Computer Society, 2005.

[63] Internet Policy Institue. Report of the national workshop on internet voting: Issues and research agenda, March 2001.

[64] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. *IACR Cryptology ePrint Archive*, 2002:165, 2002.

[65] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In Vijay Atluri, Sabrina De Capitani di Vimercati, and Roger Dingledine, editors, *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005*, pages 61–70. ACM, 2005.

[66] Jesse Kamp and David Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM J. Comput.*, 36(5):1231–1247, 2006.

[67] Aggelos Kiayias, Michael Korman, and David Walluck. An internet voting system supporting user privacy. In *22nd Annual Computer Security Applications Conference (ACSAC 2006), 11-15 December 2006, Miami Beach, Florida, USA*, pages 165–174. IEEE Computer Society, 2006.

[68] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 468–498. Springer, 2015.

[69] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 2010.

[70] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 526–535. ACM, 2010.

[71] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 122–136. IEEE Computer Society, 2010.

[72] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: New insights from a case study. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 538–553. IEEE Computer Society, 2011.

[73] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash attacks on the verifiability of e-voting systems. In *IEEE Symposium on Security and Privacy*, pages 395–409. IEEE Computer Society, 2012.

[74] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 353–365. Springer, 1990.

[75] David Lichtenstein, Nathan Linial, and Michael E. Saks. Imperfect random sources and discrete controlled processes. In Aho [3], pages 169–177.

[76] Rebecca Mercuri. A better ballot box? *IEEE Spectrum*, 39(10):46–50, 2002.

[77] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 367–378. IEEE Computer Society, 2006.

[78] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In Dwork [43], pages 373–392.

[79] Jeff Mott. Crypto-JS. `http://code.google.com/p/crypto-js/`, 2015.

[80] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Votehere, Inc. whitepaper, 2004.

[81] Peter G. Neumann. Risks to the public in computers and related systems. *ACM SIGSOFT Software Engineering Notes*, 26(1):14–38, 2001.

[82] P.G. Neumann. Security criteria for electronic voting. In *National Computer Security Conference*, pages 478–481, September 1993.

[83] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer, 1991.

[84] Stefan Popoveniuc and Benjamin Hosp. An introduction to Punchscan. In Chaum et al. [30], pages 242–259.

[85] Stefan Popoveniuc, John Kelsey, Andrew Regenscheid, and Poorvi L. Vora. Performance requirements for end-to-end verifiable elections. In Douglas W. Jones, Jean-Jacques Quisquater, and Eric Rescorla, editors, *2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '10, Washington, D.C., USA, August 9-10, 2010*. USENIX Association, 2010.

[86] Carla Ràfols. Stretching Groth-Sahai: NIZK proofs of partial satisfiability. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 247–276. Springer, 2015.

[87] Ran Raz. Extractors with weak random seeds. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 11–20. ACM, 2005.

[88] Herman Robers. Electronic elections employing DES smartcards. Master's thesis, TU Delft, 1998.

[89] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer, 1995.

[90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[91] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[92] Janos Simon, editor. *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. ACM, 1988.

[93] SJCL. Stanford Javascript Crypto Library. `http://crypto.stanford.edu/sjcl/`, 2013.

[94] Warren D. Smith. Three voting protocols: Threeballot, VAV, and Twin. In Ray Martinez and David Wagner, editors, *2007 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'07, Boston, MA, USA, August 6, 2007*. USENIX Association, 2007.

[95] Ben Smyth, Steven Frink, and Michael R. Clarkson. Computational election verifiability: Definitions and an analysis of helios and JCJ. *IACR Cryptology ePrint Archive*, 2015:233, 2015.

[96] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From Helios to Zeus. In *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013* [1].

[97] Dominique Unruh and Jörn Müller-Quade. Universally composable incoercibility. *IACR Cryptology ePrint Archive*, 2009:520, 2009.

[98] Melanie Volkamer. *Evaluation of electronic voting: requirements and evaluation procedures to support responsible election authorities*. PhD thesis, University of Koblenz-Landau, 2009.

[99] Filip Zagórski, Richard Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. Remotegrity: Design and use of an end-to-end verifiable remote voting system. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, volume 7954 of *Lecture Notes in Computer Science*, pages 441–457. Springer, 2013.

[100] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.