



**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCE  
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS**

**BSc THESIS**

# **Implementation of DEMOS Voting**

**Marios M. Levogiannis**

**Supervisor: Aggelos Kiayias, Associate Professor**

**ATHENS**

**NOVEMBER 2016**



**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

## **Υλοποίηση του DEMOS Voting**

**Μάριος Μ. Λεβογιάννης**

**Επιβλέπων: Άγγελος Κιαγιάς, Αναπληρωτής Καθηγητής**

**ΑΘΗΝΑ**

**ΝΟΕΜΒΡΙΟΣ 2016**

**BSc THESIS**

Implementation of DEMOS Voting

**Marios M. Levogiannis**

**S.N.: 1115201100058**

**SUPERVISOR: Aggelos Kiayias, Associate Professor**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

Υλοποίηση του DEMOS Voting

**Μάριος Μ. Λεβογιάννης**

**A.M.: 1115201100058**

**ΕΠΙΒΛΕΠΩΝ: Άγγελος Κιαγιάς, Αναπληρωτής Καθηγητής**

## **ABSTRACT**

This work deals with the implementation challenges of electronic voting systems. The first part analyzes the standards that an ideal voting system (either traditional or electronic) should comply with. Focus is put on the implementation details of electronic voting systems and how they compare to paper-based ones. The second part describes a new electronic voting system that was implemented as part of this work. The goal of this system is to fulfill the above requirements.

**SUBJECT AREA:** Computer Security

**KEYWORDS:** electronic voting, security, integrity, privacy, verifiability

## ΠΕΡΙΛΗΨΗ

Η εργασία αυτή ασχολείται τις προκλήσεις στην υλοποίηση των συστημάτων ηλεκτρονικής ψηφοφορίας. Στο πρώτο μέρος αναλύονται οι προδιαγραφές με τις οποίες τα συστήματα ψηφοφορίας (είτε παραδοσιακά, είτε ηλεκτρονικά) οφείλουν να συμμορφώνονται. Έμφαση δίνεται στις λεπτομέρειες υλοποίησης των συστημάτων ηλεκτρονικής ψηφοφορίας και στο πώς αυτά συγκρίνονται με τα συστήματα που βασίζονται σε έντυπα. Στο δεύτερο μέρος περιγράφεται ένα νέο σύστημα ηλεκτρονικής ψηφοφορίας που υλοποιήθηκε στα πλαίσια αυτής της εργασίας. Στόχος αυτού του συστήματος είναι να ικανοποιήσει τις παραπάνω απαιτήσεις.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:** Ασφάλεια Υπολογιστικών Συστημάτων

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:** ηλεκτρονική ψηφοφορία, ασφάλεια, ακεραιότητα, ιδιωτικότητα, επαληθευσιμότητα

Στους γονείς μου.

## ΕΥΧΑΡΙΣΤΙΕΣ

Κατ' αρχήν θα ήθελα να ευχαριστήσω τον καθηγητή κ. Άγγελο Κιαγιά, που μου έδωσε την δυνατότητα να ασχοληθώ με το συγκεκριμένο αντικείμενο καθώς και για την καθοδήγηση και την υπομονή του κατά τη διάρκεια της εκπόνησης αυτής της εργασίας. Επίσης θέλω να ευχαριστήσω τον υπεύθυνο της πτυχιακής εργασίας Bingsheng Zhang για την εξαιρετική συνεργασία που είχαμε και τον χρόνο που αφιέρωσε για να με βοηθήσει. Τέλος θέλω να ευχαριστήσω τον Θωμά Ζαχαρία τις πολύτιμες παρατηρήσεις του.



# CONTENTS

<b>PREFACE</b> . . . . .	<b>14</b>
<b>1. INTRODUCTION</b> . . . . .	<b>15</b>
<b>2. STANDARDS OF FAIR AND TRANSPARENT ELECTIONS</b> . . . . .	<b>16</b>
<b>2.1 Equal Access to Electoral Centers</b> . . . . .	16
<b>2.2 Secrecy</b> . . . . .	16
<b>2.3 Coercion Resistance</b> . . . . .	16
<b>2.4 Cast-as-Intended Verifiability</b> . . . . .	17
<b>2.5 Recorded-as-Cast Verifiability</b> . . . . .	18
<b>2.6 Tallied-as-Recorded Verifiability</b> . . . . .	18
<b>2.7 Universal Verifiability</b> . . . . .	18
<b>2.8 End-to-End Verifiability</b> . . . . .	19
<b>2.9 Voter Eligibility</b> . . . . .	20
<b>2.10 One-Voter-One-Vote</b> . . . . .	20
<b>2.11 Fault Tolerance</b> . . . . .	20
<b>2.12 Fairness</b> . . . . .	20
<b>2.13 Receipt-Freeness</b> . . . . .	21

<b>3. DEMOS VOTING</b> . . . . .	<b>23</b>
<b>3.1 Description</b> . . . . .	<b>23</b>
<b>3.2 Implementation</b> . . . . .	<b>23</b>
3.2.1 Software that was used . . . . .	23
3.2.2 Server architecture . . . . .	24
3.2.3 Database schema . . . . .	24
3.2.4 Binary data encoding . . . . .	25
3.2.5 Key derivation function . . . . .	25
3.2.6 Server-to-server communication . . . . .	25
3.2.7 Election with parties and candidates . . . . .	26
3.2.8 Ballots . . . . .	26
3.2.9 Security code . . . . .	27
3.2.10 Vote-code . . . . .	28
3.2.11 Receipt . . . . .	29
3.2.12 Cryptographic operations . . . . .	29
<b>3.3 Presentation</b> . . . . .	<b>30</b>
3.3.1 Election Authority . . . . .	30
3.3.1.1 Home page . . . . .	30
3.3.1.2 User management . . . . .	30
3.3.1.3 Creating a referendum . . . . .	30
3.3.1.4 Adding a question to the referendum . . . . .	32
3.3.1.5 Creating an election . . . . .	32
3.3.1.6 Adding a party to the election . . . . .	33
3.3.1.7 Checking the election status . . . . .	34
3.3.1.8 Browsing the list of elections . . . . .	35
3.3.2 Ballot Distribution Server . . . . .	36
3.3.2.1 Selecting the distribution method . . . . .	36
3.3.3 Virtual Ballot Box . . . . .	37
3.3.3.1 Presentation of the ballot . . . . .	37
3.3.3.2 About the security code . . . . .	38

3.3.3.3	Filling in the ballot . . . . .	39
3.3.3.4	Confirming the vote-codes . . . . .	39
3.3.3.5	Verifying the receipts . . . . .	40
3.3.4	Audit and Results . . . . .	41
3.3.4.1	Election results . . . . .	41
3.3.4.2	Verification phase . . . . .	41
3.3.4.3	Access to audit information . . . . .	43
<b>3.4</b>	<b>Deployment . . . . .</b>	<b>44</b>
3.4.1	National Elections 2015 - Pilot Experiment . . . . .	44
3.4.2	University of Athens . . . . .	45
3.4.3	Labour Institute of the Greek General Confederation of Workers . . . . .	45
<b>4.</b>	<b>CONCLUSIONS AND FUTURE WORK . . . . .</b>	<b>46</b>
	<b>ABBREVIATIONS - ACRONYMS . . . . .</b>	<b>47</b>
	<b>ANNEX I . . . . .</b>	<b>48</b>
	<b>REFERENCES . . . . .</b>	<b>61</b>

**LIST OF FIGURES**

Figure 1: Schema of the database . . . . . 24

## LIST OF IMAGES

Image 1:	Home page . . . . .	30
Image 2:	Creating a referendum . . . . .	31
Image 3:	Adding a question to the referendum . . . . .	32
Image 4:	Creating an election . . . . .	33
Image 5:	Adding a party to the election . . . . .	34
Image 6:	Checking the election's setup progress . . . . .	35
Image 7:	List of elections . . . . .	35
Image 8:	Ballot distribution . . . . .	36
Image 9:	Sample ballot . . . . .	37
Image 10:	Security code input . . . . .	38
Image 11:	Selecting options . . . . .	39
Image 12:	Confirming vote-codes . . . . .	40
Image 13:	Verifying receipts . . . . .	40
Image 14:	Referendum results . . . . .	41
Image 15:	Auditing the ballot part that was cast . . . . .	42
Image 16:	Example of audit information in JSON format . . . . .	42
Image 17:	Auditing the ballot part that was not cast . . . . .	43
Image 18:	Polling station . . . . .	44
Image 19:	Ballot 105 - Part A . . . . .	49
Image 20:	Ballot 105 - Part B . . . . .	50
Image 21:	Ballot 109 - Part A (page 1/4) . . . . .	51
Image 22:	Ballot 109 - Part A (page 2/4) . . . . .	52
Image 23:	Ballot 109 - Part A (page 3/4) . . . . .	53
Image 24:	Ballot 109 - Part A (page 4/4) . . . . .	54
Image 25:	Ballot 109 - Part B (page 1/4) . . . . .	55
Image 26:	Ballot 109 - Part B (page 2/4) . . . . .	56
Image 27:	Ballot 109 - Part B (page 3/4) . . . . .	57
Image 28:	Ballot 109 - Part B (page 4/4) . . . . .	58

## PREFACE

This thesis was completed at the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens during the fall of 2016. Its purpose was to implement a secure e-voting system that meets the fundamental requirements of election systems (integrity, privacy, verifiability, etc), while remaining easy to use.

DEMOS Voting is an open source, web-based, public auditable e-voting system. More information about it can be found at:

- <http://www.demos-voting.org/>

The source code is available at:

- <https://github.com/mlevogiannis/demos-voting/>

## 1. INTRODUCTION

Electronic voting has spread throughout the world without sufficient attention to security, transparency and reliability. The dramatic increase in Internet usage over the last decades has opened the possibility for anyone to develop and offer unproven electronic voting services. Although e-voting has the potential for increasing the quality of democratic representation, it needs careful design in order to create a trustworthy system.

In traditional, paper-based elections, voters submit their ballots, enclosed in sealed envelopes, into a clear ballot box. At the end of the voting period, the authorities open the submitted ballots, count the votes and specify the outcome. On the other hand, in e-voting systems all the above procedures are handled by computers.

Properly implemented e-voting systems offer several advantages, such as fast and accurate counting of ballots, reduced the costs and improved accessibility for disabled or remote voters. However, they are susceptible to new types of attacks and electoral fraud. As a result, they need to be carefully designed, focusing on security.

There are two types of e-voting systems: Remote Voting and On-site Voting. The former lets the voters participate remotely, usually through the Internet using their own device (e.g. personal computer or mobile phone). The latter requires the voters to cast their votes in a polling station using specially designed voting machines.

E-voting systems have been employed in many countries for national, state-wide and municipal elections. However, for most such voting systems used in practice, voters have no guarantees that their votes have actually been counted. This is where verifiability comes in.

Constraints of secure voting systems are contradictory. For example, the requirement for voter privacy contradicts verifiability. No one should be able to know how a voter voted, even if they want them to. But on the other hand, the voter should be able to confirm that their vote was counted as she intended. Many systems attempt to overcome this by using cryptography and highly interactive protocols.

Organization of this thesis: Chapter 2 presents the required security properties of voting systems, describing how they are achieved both in traditional and electronic voting protocols. Chapter 3 describes the implementation and presents DEMOS Voting, a new e-voting system that is designed to be secure and verifiable.

## **2. STANDARDS OF FAIR AND TRANSPARENT ELECTIONS**

In this chapter, the requirements of voting protocols with regard to fairness and transparency will be discussed. They have been extensively studied in literature both from a political and a cryptographic point of view. These are equal access to electoral centers, secrecy, coercion resistance, cast-as-intended verifiability, recorded-as-cast verifiability, tallied-as-recorded verifiability, universal verifiability, end-to-end verifiability, voter eligibility, one-voter-one-vote, fault tolerance, fairness and receipt-freeness [1].

### **2.1 Equal Access to Electoral Centers**

Voting systems should ensure unrestricted and equal access of all eligible voters to electoral centers. Remote e-voting systems offer a high level of accessibility. Voters can vote using general purpose equipment that they already possess, which allows for a higher voter turnout, especially among young people. Furthermore, specially designed devices can be used so that people with disabilities are not discouraged from participating. On the other hand, access to polling stations of on-site voting systems (either paper-based or electronic) may pose unsurpassed difficulties for disabled people or those of remote regions.

### **2.2 Secrecy**

Voting systems should also ensure the non-identifiability of the voters. In traditional voting systems secrecy is physically protected, but e-voting systems may be vulnerable to violations of secrecy. Secure e-voting systems achieve this property using cryptography, in particular by encrypting the specially encoded votes.

### **2.3 Coercion Resistance**

Coercion resistance is an important security requirement of voting protocols. A potential coercer should not be able to influence the behavior of a voter. Thus, a voting protocol is said to be coercion resistant if it prevents voter coercion and vote buying [2].

Coercion and vote buying by party officials and candidates are the most common types of election fraud. E-voting systems can offer several improvements to the situation. Voters may be allowed to vote multiple times and only the last one will be taken into account, or the system may support fake ballots which will not be counted during the tallying phase. This way coercion becomes difficult to achieve, as the coercer cannot ensure that the recorded vote is the voter's final decision [1].



But still, perfect resistance to coercion cannot be achieved. There are several forms of serious, real-world attacks that an adversary can mount against voting protocols [3]:

- **Randomization attack:** The idea is for the adversary to coerce a voter to submit a randomly filled ballot. The attacker and perhaps even the voter are not aware of what candidate the ballot was cast for. The effect of this attack is to nullify the choice of the voter with a large probability. For example in an election with two parties, candidates of party *A* would benefit from this attack against voters of party *B*.
- **Forced-abstention attack:** Voting schemes that authenticate voters in order to participate in the election are susceptible to this attack. If the coercer can see who has voted and who has not, they can use this information to threaten voters to refrain from voting.
- **Simulation attack:** The attacker can coerce a voter to reveal their secrets (e.g. steal or buy them) after the registration process but prior to the election process. The attacker can now act on behalf of the voter.

Furthermore, a coercer may use an irregular method to test the loyalty of a voter in a probabilistic way. They may give a garbage vote in one out of every one hundred coerced voters. If the voter is unable to decide whether it is garbage or not, the adversary may distinguish a voter following the coercer's instructions from a voter who is trying to cheat the coercer. For example, if the voter casts their own vote instead of the one given by the coercer, the coercer will notice this difference. However, this comes at the cost of losing a particular vote, but it is a means of applying pressure on the voters to cooperate. This is known as a fault attack [4].

## 2.4 Cast-as-Intended Verifiability

A voting protocol is cast-as-intended verifiable if the contents of each vote can be audited in order to ensure that they match the voter's selections. Thus, a corrupt voting device is not able to cast a ballot for an option different than the one chosen by the voter.

The main problem with most cast-as-intended verification systems is that they are not user friendly as they usually require the voter to perform complex computations or to engage in highly interactive protocols. As a result and in order to avoid discouraging less skilled voters from participating, many voting protocols make this part of the procedure optional. This does not solve the problem though, but instead opens new attack vectors. Adversaries will target non-skilled voters, who will probably not use the verification system, and this way they will succeed without being detected [5].

In general, cast-as-intended verifiability is inherent to traditional, paper-based voting systems, as the voters themselves put the ballot with their choices in an envelope, which is then sealed.

## 2.5 Recorded-as-Cast Verifiability

Recorded-as-cast verifiability ensures that a vote is recorded by the election system correctly. In e-voting systems, this requirement is typically implemented by giving a receipt to the voter, and posting all receipts on a public bulletin board. The voters then may check that their receipts are correctly posted there. However, this property has to be achieved without the cost of disclosing any information on how they voted to a potential malicious entity controlling their device.

If a voter detects that their vote is not correctly recorded, they must be able to bring some sort of evidence, so that false complaints are impossible. This would be possible, for example, using signed receipts that a dishonest voter would not be able to forge. Furthermore, the proof that the honest voter needs to provide is not necessarily required to be an irrefutable proof of malfeasance, but a probabilistic proof is usually sufficient. For example, the voter may only be able to check one of part of their ballot, which the voting system cannot predict before it is checked [6].

Again, recorded-as-cast verifiability is inherent to traditional, paper-based voting systems, as the voters themselves put their ballots (possibly enclosed in a sealed envelope) into a clear ballot box.

## 2.6 Tallied-as-Recorded Verifiability

Every voter should be able to verify that their vote has been accurately included in the final tally, without any requirement to trust the system components [7]. This can be achieved with any of the following cryptographic methods [8]:

- Verifiable mixing: Encrypted votes are dissociated from the voters' name by shuffling. This allows the authorities to prove that votes have not been tampered with, but prevents them from tracing any given vote to the corresponding voter. The votes are then decrypted in order to be counted.
- Homomorphic encryption: Encrypted votes can be tallied without being decrypted first. The encrypted result can then be decrypted to prove that the tally is correct.

This property is difficult to achieve in paper-based voting protocols, as counting is done manually by the counting agents (who are usually appointed by political candidates and parties) without the presence of the voter.

## 2.7 Universal Verifiability

Universal verifiability is the property of an e-voting system that enables anyone to check that the tally of recorded ballots published on a bulletin board is computed properly [9].

It differs from the previously-discussed verifiability properties as it refers to external observers that are interested in the final result and not in individual votes. Verification of the result should be done without compromising the voters' privacy.

Individual verifiability (cast-as-intended, recorded-as-cast, tallied-as-recorded) and universal verifiability are not sufficient to guarantee the global verifiability. What they ignore is that dishonest authorities/voters can break the integrity of ballots of honest voters by ill-formed ballots [10].

## 2.8 End-to-End Verifiability

An end-to-end (E2E) verifiable voting system allows the voters to verify the process and the final result by checking a public bulletin board. The purpose of such systems is to give voters the ability to detect a malicious election authority that tries to misrepresent the election outcome. In an end-to-end verifiable election, a voter should be able to check that their vote was cast as intended, recorded as cast, and tallied as recorded [11].

Formally, an election is said to be end-to-end verifiable if and only if [6]:

- Presented ballots are well-formed: The representation of the voter's choices on the ballot agrees with the representation that will be read by the rest of the election system. Otherwise, a voter might be convinced they had cast a vote for candidate *A*, when their ballot encoded a vote for candidate *B*.
- Cast ballots are well-formed: Cast ballots can neither contain multiple votes, nor can contain negative votes to decrease the final count of votes of a specific candidate.
- Recorded as cast: The ballot the voter cast is the one that was received and saved by the voting system.
- Tallied as recorded: The votes on the cast ballots are counted correctly to get the public tally.
- Consistency: The set of ballots subject to the recorded as cast check is the same as the set of ballots subject to the tallied as recorded check.
- Recorded ballots have one-to-one correspondence to unique voters: No ballots are included in the final tally that could not have been checked by at least one voter.

If any votes were added, deleted, changed, or invalidated after being cast, there is a substantial probability it will be detected by any observer that can verify, as it is likely to result in the failure of at least one of the above checks. This would provide strong evidence that the reported result is not correct.

It is now obvious that end-to-end verifiability may come into conflict with voter privacy. In order to maintain strong voter privacy, voting systems usually utilize cryptographic methods

which require computations on the client side. To avoid potential client-side vulnerabilities, a technique called code-voting can be used. With code-voting, the election authority corresponds vote-codes to voter choices. Without knowledge of the codes, malicious devices cannot sensibly modify voter choices. On the other hand, this technique does not protect against a malicious election authority, which inevitably has knowledge of all vote-codes [12].

In some systems it is possible for voters to pass or outsource audit information to a third party thus enabling a single external entity to ensure all levels of verifiability [1].

## **2.9 Voter Eligibility**

Only eligible voters should be able to participate in the election process. This means that eligible voters should not be denied the right to vote, while ineligible voters should not be able to cast any vote.

## **2.10 One-Voter-One-Vote**

An e-voting system should make sure that the “one voter, one vote” (1V1V) principle is respected. While voter eligibility ensures that only eligible voters can vote, 1V1V ensures that they can vote only once, that is malicious parties (including the eligible voters themselves) should not be able to duplicate votes [13]. To achieve this property, co-operation between the ballot casting system and the registration system is required. Furthermore, all cast ballots must have the same influence on the result.

## **2.11 Fault Tolerance**

Fault tolerance is the property of a voting system that enables it to continue operating properly in the event of the failure of one or more of its components. This property is extremely important to e-voting systems, whose components are electronic devices that are open to a wide range of attack vectors. It should be possible to recover from such faults, or at least to detect them [1]. For example, the servers controlling the voting process may fail or become unreachable. A distributed version of the voting system could survive such a scenario.

## **2.12 Fairness**

Fairness is the property that ensures that no early results can be obtained which could influence the remaining voters. Although it should be impossible for an attacker to learn

partial results during the ballot-casting stage of an election, it is also impossible to prevent “exit polls”, i.e. people who willingly reveal their vote when asked [14]. It is possible to violate this property of e-voting systems if an adversary controls one or more of the ballot-casting or tallier servers.

### 2.13 Receipt-Freeness

Receipt-freeness is the property of voting protocols that a voter cannot create a receipt which proves how they voted [15]. While privacy protects honest voters, receipt-freeness aims at protecting vote privacy even when voters willingly provide information to an attacker. The ability of a voter to obtain a receipt of the way they voted opens the possibility for an adversary to coerce (e.g. threaten) a voter to vote in a particular manner, or a voter to sell or auction their vote. In physically-based election systems, the voting booth does not only allow voters to keep their votes secret, but it actually forces them to do so.

Formally, a receipt  $R$  is an object that proves that a voter  $V$  cast a vote for candidate  $C$ . It has the following properties [16]:

- $R$  can only have been generated by  $V$ .
- $R$  proves that  $V$  chose candidate  $C$ .
- $R$  proves that  $V$  cast their vote.

The distinction between the second and the third property is subtle. In a traditional setting with a voting booth and paper ballots, a voter could choose the candidates they wish, leave the voting booth without casting the ballot and then show it to anyone. While this satisfies the first and the second properties, the vote is not valid as the voter did not cast the ballot.

Receipt-freeness requires that there must be some private time between the voter and the voting procedure. If the potential coercer or vote-buyer is always present, it is not possible to achieve receipt-freeness. In such a scenario, the adversary could play the role of the voter while the voter is passive, so isolation between them is necessary. For traditional election systems, the voting booth is able to separate voters from outsiders. For e-voting systems, the voting booth can be modeled as an untappable channel between the voter and the tallying authority. Any message sent between them must be perfectly secret to the attacker.

The difference between coercion-resistance and receipt-freeness lies in the powers of the coercer to interact with the voter during the voting stage. In coercion-resistance, the coercer has the ability to cooperate with the voter, or even vote on behalf of them. On the contrary, in receipt-freeness the coercer simply examines evidence gained from observing the election process. This includes information provided by the voter [4].

Receipt-freeness is hard to achieve simultaneously with universal verifiability. Traditional election systems are able to satisfy the first property, but fail to achieve the second one.

On the other hand, several e-voting systems satisfy the second property, but fail to meet the first one. For example, the voter may be required to perform secret calculations on a computer which may be running malicious code or be completely untrusted (e.g. provided by an adversary). In both cases, the adversary can learn the secret and find out how the voter voted [17].

## 3. DEMOS VOTING

As part of this work, the e-voting system described in [11] was implemented. The source code is available at: <https://github.com/mlevogiannis/demos-voting/>.

### 3.1 Description

DEMOS Voting is an open source, web-based, public auditable e-voting system. The system consists of four main components:

- Election Authority (EA)
- Ballot Distribution Server (BDS)
- Voter Bulletin Board (VBB)
- Audit Bulletin Board (ABB)

The EA is the supervisor that administers an election, generates the ballots and distributes the secret election key among several trustees. The BDS is responsible for authorizing the voters. The ABB is used by the EA to record all the information needed for verifiability of the election. The VBB is the ballot box that the voters use to cast their votes.

### 3.2 Implementation

#### 3.2.1 Software that was used

The system is built as a web application. The server-side logic is implemented in the Python [18] programming language, using the Django [19] web framework. The code runs under both Python 2 and 3, using the six [20] compatibility layer. Celery [21] is used for event scheduling, and execution of asynchronous tasks. pyOpenSSL [22] and petlib [23] libraries, which are thin wrappers around OpenSSL, are used to perform cryptographic operations. Django REST framework [24] is used to provide a REST API for accessing the audit information. To generate paper ballots, ReportLab PDF library [25] is employed. The client-side is written in HTML, CSS and JavaScript. To design a responsive and user friendly interface, the front-end uses the Bootstrap [26] framework. The back-end uses the jQuery [27] library for code simplicity and readability. The Stanford Javascript Crypto Library [28] is utilized for cryptographic operations.

### 3.2.2 Server architecture

To accommodate the requirements of the system’s architecture, the code is organized into five Django applications. The term application describes a Python package that provides a set of features. The *common* application contains the abstract database models, utilities, templates and everything that is shared between the other applications. Each of the main applications corresponds to one of the system’s four main components: the *ea* application to the Election Authority, the *bds* application to the Ballot Distribution Server, the *vbb* application to the Voter Bulletin Board and the *abb* application to the Audit Bulletin Board. The four main applications can (and should) be installed independently on different servers.

### 3.2.3 Database schema

Figure 1 shows the schema of the database. The tables are implemented as Django abstract model classes and are defined in the *common* package. Component-specific fields (e.g. security code, vote-codes, receipts, encoded candidate commitments, etc) are defined in the respective package sub-classes.

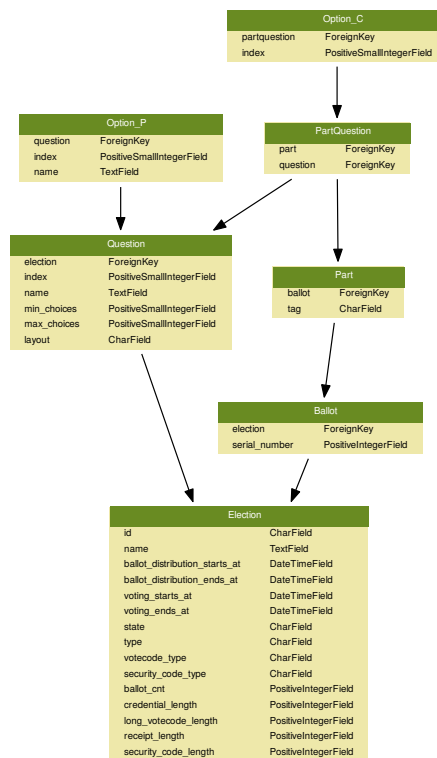


Figure 1: Schema of the database



### 3.2.4 Binary data encoding

Arbitrary byte data that need to be manipulated by users are represented as Base32 encoded strings. This notation uses a restricted set of symbols that can be conveniently used by humans and efficiently processed by computers. The exact variation is described in [29]. The symbol set consists of 10 digits and 22 letters (all A-Z excluding I, L, O and U). It is case-insensitive and, to avoid confusion with digits when decoding, i and l are treated as 1 and o is treated as 0. Hyphens (-) can be inserted into symbol strings to improve readability.

### 3.2.5 Key derivation function

The auditing stage after the election requires that voters can check the validity of the data printed on their ballots (e.g. credentials, vote-codes, etc). The same process is also required by the voting stage, where the system needs to authenticate ballots and cast vote-codes. To achieve this, password hashing techniques are employed. The hash string format conforms to the Modular Crypt Format (MCF) which is described in [30], but uses a custom implementation. Hashes have the format *\$identifier\$content*, where *identifier* is an alphanumeric string uniquely identifying a particular scheme and *content* is the contents of the scheme.

The default hash scheme is *pbkdf2-sha512*. It uses PBKDF2 as the key derivation function and HMAC-SHA512 as the pseudo-random function. The number of iterations is 100000 and the cryptographic salt's length is 16 bytes. The contents of this scheme have the format *iterations\$salt\$digest*.

The advantage of using a standardized scheme for encoding hash strings is that their parameters can be easily changed without breaking backward compatibility. For example, it is recommended that the number of iterations increases over time as CPU speeds increase. Furthermore, the scheme itself can be changed if it is required by the security policy of a particular installation or it is found to be insecure (hypothetical scenario). Possible candidates are *bcrypt*, *scrypt*, *argon2*, etc.

### 3.2.6 Server-to-server communication

A private web API is employed for the communication between the system's components (servers). All requests are made over HTTPS, using the JSON format for data serialization. The private API uses a custom HTTP scheme based on a keyed-HMAC for authentication.

To authenticate a request, the *client* first concatenates the request's elements to form a string. Then, a pre-shared secret key is used to calculate the HMAC of that string, which is the request's signature. Finally, the signature is added as a parameter to the request's HTTP headers. When the *server* receives a request, it fetches the *client's* pre-shared secret key and repeats the above process to compute its signature. If the calculated sig-

nature matches the received signature, then the *client* must have access to the secret key and the request proceeds. Otherwise, the request is rejected and an error is returned. To avoid replay attacks, the authentication scheme mandates that the *client* includes a unique nonce value and a timestamp in the request's elements, before the calculation of the signature. As a result, the *server* needs to maintain a finite list of past nonces.

### 3.2.7 Election with parties and candidates

From the description of subsection 3.2.3, it can be seen that the system directly supports only referendums with one or more questions and multiple options. An election with parties and candidates can be modeled as a referendum with two questions.

The first question is the party list. Options are the parties that take part in the election. The set of options also includes a blank party, which means *none of the above*. Apart from this, the party list is just an ordinary question.

The second question is the candidate list. Options are the candidates of all parties that take part in the election. Things here are more complicated, as candidates must be grouped by party. The ordering of the candidate groups in the candidate list is the same as the order in which parties appear in the party list. Since voters are usually allowed to select multiple candidates, extra blank candidates are added to all candidate groups, equal to the maximum number of selections. The group of the blank party has only blank candidates. In order to protect voters' privacy, all parties should have exactly the same number of candidates. Otherwise, voting for a party with  $N$  candidates would leak information about which parties the voter had not voted for, that is those with a number of candidates different than  $N$ . To overcome this issue, blank candidates are added to all candidate groups until they reach the number of candidates of the party with most candidates.

In the voting phase, voters need to choose exactly one option from the party list and exactly as many options as the maximum number of selections from the candidate list. Using the special grouping of candidates described in the previous paragraph, it is trivial for the VBB to verify that the selected candidates belong to the selected party. The candidates associated with the  $i$ -th option of the first question will be in range  $[(c+s)*i, (c+s)*i+s-1]$  of the second question, where  $c$  is the number of candidates of the party with most candidates and  $s$  is the maximum number of selections. Furthermore, if a voter wishes to cast votes for a number of candidates less than the maximum number of selections, they will have to include blank candidates until this number is reached. This should usually be done automatically by the back-end of the voting client.

### 3.2.8 Ballots

The ballots used by the system are double ballots. Every ballot consists of two equivalent parts, identified by the tags  $A$  and  $B$ . The voter will flip a coin to choose the part that they are going to use for voting. In the verification phase, the unused part will be opened and

the voter would be able to check that the correspondence of vote-codes (see subsection 3.2.10) and options in this part has not been tampered with. Otherwise, a malicious EA will be caught with probability  $1/2$ .

Every double ballot is identified by a serial number, starting from 100. Furthermore, each ballot part has a random 128-bit credential, which is used by the VBB for voter authorization. The number of ballots must be specified before the referendum/election setup begins and it cannot be changed afterwards. Ballots are distributed by the BDS.

### 3.2.9 Security code

After the EA has prepared the vote-code (see subsection 3.2.10) and encoded candidate pairs, it needs to decide how they are going to be ordered on the VBB and ABB. Shuffling these pairs is necessary in order to protect the voters' privacy. Each question's ordering is determined by a random permutation. The permutation indices of all questions are stored in the security code, which is printed on the upper right corner of the ballot part. The security code must never be given to any third-party as it can be used to reveal the meaning of the vote-codes. However, it can be used by the voting client to offer a user-friendly voting interface. If the voter does not trust that their client will keep it secret, they can still vote by directly casting the vote-codes printed on their ballot.

Consider a referendum with  $n$  questions  $Q_0, \dots, Q_{n-1}$ , where each question  $Q_i$  has  $x_i$  options. The number of permutations for each one is  $x_i!$ , so there are  $(x_0!) \cdot \dots \cdot (x_{n-1}!)$  possible shuffles in total. Therefore, a random integer in range  $[0, x_i! - 1]$  is the permutation which will determine the order of the vote-code and encoded candidate pairs. The random permutation of a question  $Q_i$  is stored in the least significant bits of the security code, but after the random permutation of question  $Q_{i-1}$ .

Now consider an election with parties and candidates. Although the party list is a simple question, the candidate list requires special handling. For the purposes of security code splitting, each candidate group will be treated as separate question. Let  $p$  be the number of parties and  $c$  the number of candidates of each group (remember that all candidate groups have the same number candidates). The number of permutations of the first question (party list) is  $p!$ , while the number of permutations of every other "question" (candidate group) is  $c!$ . There are  $(p!) \cdot (c!)^p$  possible shuffles in total. To get the shuffling of the candidate list, first the candidates of each group have to be shuffled using their corresponding permutation index and then the candidate groups will be shuffled using the party list's permutation index. This way, the property described in the last paragraph of subsection 3.2.7 is still valid and the VBB can associate candidates with parties without access to the security code.

The security code is then encoded in a string format, which can be either *numeric* or *alphanumeric*. The numeric format uses digits from 0 to 9, while an alphanumeric format uses the Base32 alphabet described in subsection 3.2.4. Numeric security codes are user-friendly, but will overflow their maximum value (see below) much faster compared to alphanumeric ones. On the contrary, alphanumeric security codes have a greater "capac-

ity” compared to numeric ones, but are more difficult for users to manipulate.

The security code's length is variable, but has some fixed lower and upper bounds. The minimum length is 4 characters, the maximum length is 8 characters. If the length of the security code after it has been encoded is less than the minimum length, it is padded with random bits, which will be ignored during decoding. On the other hand, if it is greater than the maximum length, the scheme described in the previous paragraphs is dropped and a completely random security code is generated. It will then be used as source to a randomness extractor in order to generate a permutation index greater than those that could be stored in the security code.

Finally, referendums/elections without a security code are also supported. By using the only available source of entropy, the ballot part's credential, a randomness extractor is used to generate the random permutations. In that case, the voter's privacy is not guaranteed, as this credential is also part of the voting booth's URL and is sent over the wire. Although the connection between the voter and the voting booth is encrypted, an adversary that controls the VBB could see this value and associate vote-codes with the actual options.

The current implementation uses HMAC-SHA512 as the randomness extractor. The key is the credential of the ballot part and the message is the security code (if enabled) together with the index of the question.

### 3.2.10 Vote-code

Ballot casting utilizes a code-voting approach. The EA corresponds vote-codes to commitments posted in the ABB, and voters cast their votes by simply sending the vote-codes that they prefer to the VBB. 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” (see [11] for more information).

Consider a question  $Q_i$  with  $x_i$  options. Each ballot part will contain a list of  $x_i$  vote-codes corresponding to the list of this question's options. A vote-code can be either *short* or *long*, depending on the election's configuration. A short vote-code is a unique random integer in range  $[1, x_i]$ . On the other hand, a long vote-code is the 16 least significant characters of the Base32-encoded output of HMAC-SHA256, where the key is the credential of the ballot part and the message is the security code (if enabled) together with the index  $i$  of the question and the index  $j$  of the option.

Short vote-codes are user-friendly and easy to use. However, it is obvious that they are predictable. If the VBB or ABB are controlled by an adversary, then it is possible to intercept and change the voters' cast vote-codes. But since short vote-codes cannot be associated with options without the security code, the adversary can only corrupt a voter's ballot by casting random vote-codes.

A referendum/election that uses long vote-codes is not susceptible to the randomization attack described in the previous paragraph. Since a long vote-code is unpredictable (if

a security code is used) and unique across all options, it is possible to use its hash to associate it with its corresponding option during the voting phase. As a result it is sufficient to post only hashes in the VBB and ABB. If an adversary tampers with cast vote-codes, they will be caught during the auditing phase.

### **3.2.11 Receipt**

End-to-end verifiability requires that voters can obtain a receipt at the end of the ballot casting procedure that allows them to verify that their vote was cast as intended, recorded as cast and tallied as recorded. Receipts are generated by the EA and are associated with the vote-code and encoded candidate pairs.

The way receipts are generated depends on the type of vote-codes that is used. In the short vote-code case, a receipt is a random string of length 8 over the Base32 alphabet. In the long vote-code case, a receipt is the Base32-encoded RSA signature of the long vote-code. Since RSA signatures are very long and in order to be consistent with short vote-codes' receipts, only the last 8 characters of a long vote-codes' receipts are printed on ballots.

### **3.2.12 Cryptographic operations**

Implementation of the cryptographic operations (homomorphic commitment scheme, zero-knowledge proofs, trustees, etc) described in [11] is not covered by this work.

### 3.3 Presentation

#### 3.3.1 Election Authority

##### 3.3.1.1 Home page

From the home page, authenticated users have access to the functionality offered by the EA, as shown in image 1.

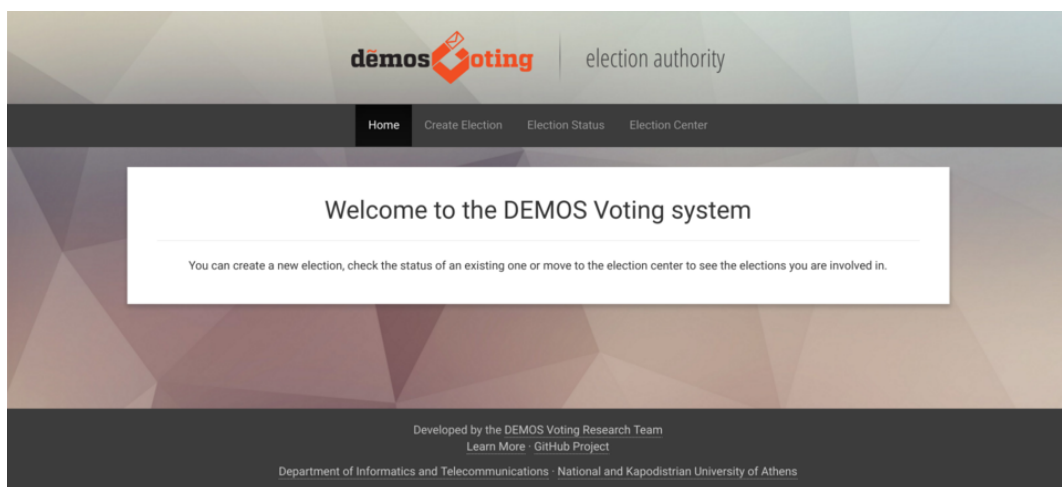


Image 1: Home page

This includes:

- Creating a new election.
- Checking the status of an election.
- Browsing the list of available elections.

##### 3.3.1.2 User management

Only registered users can create elections. User registration is not handled by the system, but a single sign-on service is typically employed (e.g. the university's CAS server). The user who creates an election is considered the administrator of this election.

##### 3.3.1.3 Creating a referendum

Creating a referendum is as simple as filling in a form. Image 2 shows an example of the creation page for a referendum.

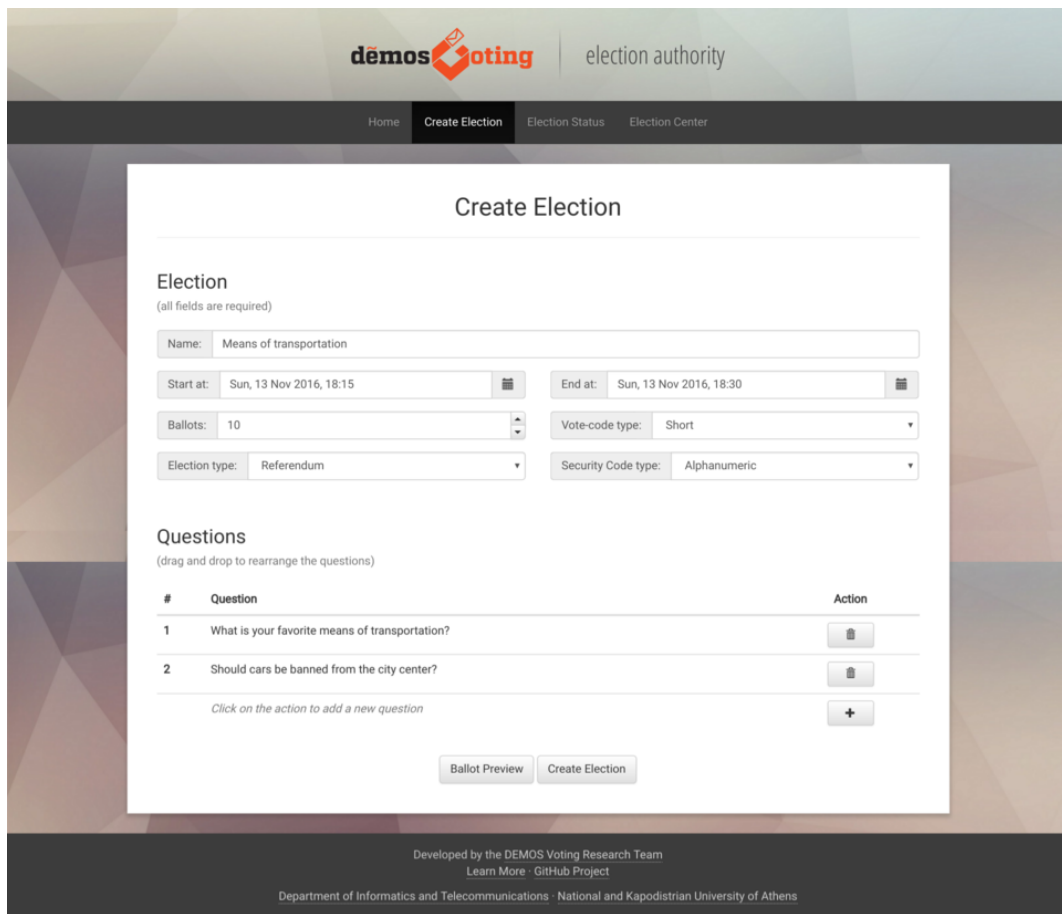


Image 2: Creating a referendum

The administrator needs to select:

- The referendum's name.
- The date and time when voting is expected to begin.
- The date and time when voting is expected to end.
- The number of ballots that will be generated.
- The type of security code that will be used (none, numeric or alphanumeric).
- The type of vote-codes that will be used (short or long).

Using numeric security codes may lead to decreased privacy, but on the other hand alphanumeric security codes make the voting process less user-friendly. Not using a security code at all implies even less privacy, as anybody would be able to match vote-codes to options/candidates (although they still would not know the voter that cast the ballot). The same applies to vote-codes, short vote-codes are user-friendly but easy to guess, on the contrary long vote-codes are user-hostile but unpredictable.

### 3.3.1.4 Adding a question to the referendum

One or more questions can be added to the referendum, as shown in image 3:

Image 3: Adding a question to the referendum

The administrator needs to select:

- The question's name.
- The minimum number of choices that a voter can make.
- The maximum number of choices that a voter can make.
- Whether the options will be presented in a one- or two-column layout.

Finally, two or more options need to be added. Options can be rearranged with a simple drag-and-drop operation.

### 3.3.1.5 Creating an election

Creating a election is almost the same as creating a referendum. Image 4 shows an example of the creation page for an election.



The screenshot shows the 'Create Election' form in the DEMOS Voting system. The form is titled 'Create Election' and is part of the 'election authority' interface. It contains several sections:

- Election** (all fields are required):
  - Name: National Elections
  - Start at: Sun, 13 Nov 2016, 19:00
  - End at: Sun, 13 Nov 2016, 19:15
  - Ballots: 10
  - Election type: Elections
  - Electoral system: Proportional representation
  - Vote-code type: Long
  - Security Code type: Numeric
  - Maximum choices: 3
- Parties** (drag and drop to rearrange the parties):

#	Party	Action
1	Pink	[Delete]
2	Blue	[Delete]
3	Green	[Delete]
4	Red	[Delete]
5	Brown	[Delete]
6	Cyan	[Delete]
7	Orange	[Delete]
		[+]

*Click on the action to add a new party*

At the bottom of the form are two buttons: 'Ballot Preview' and 'Create Election'.

Image 4: Creating an election

The only difference is that the administrator additionally needs to select:

- An electoral system (currently only proportional representation is supported).
- The maximum number of candidates that a voter can select.

### 3.3.1.6 Adding a party to the election

Adding a party to the election is almost the same as adding a question to the referendum, as shown in image 5:

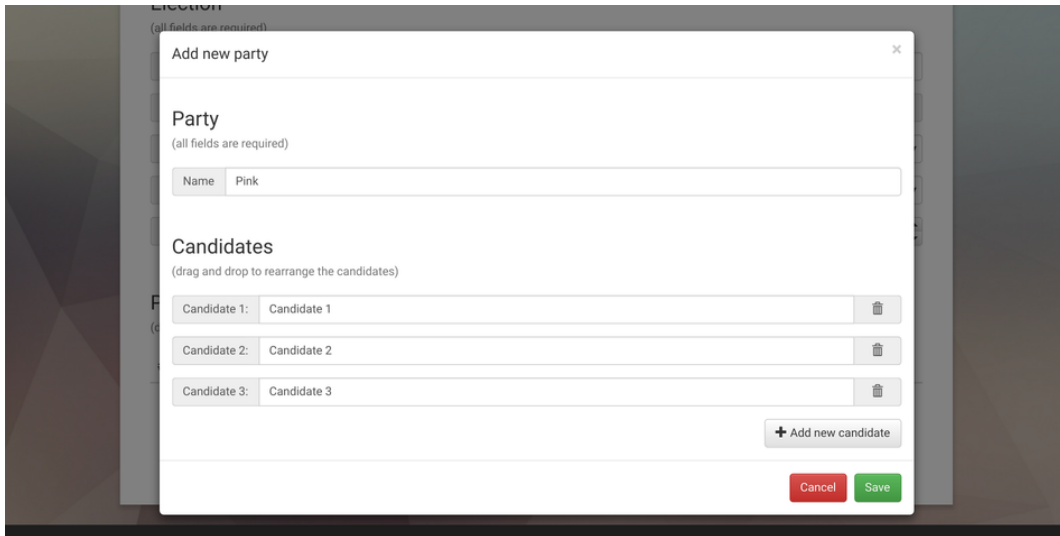


Image 5: Adding a party to the election

The only difference is that the administrator does not have to select the minimum and maximum number of choices that a voter can make. The minimum number is always 0, as a voter may opt to vote for a party but not for any of its candidates. The maximum number is common to all parties and is specified in the election's creation page.

Finally, one or more candidates need to be added. Candidates can be rearranged with a simple drag-and-drop operation.

### 3.3.1.7 Checking the election status

After an election has been defined, the election committee need to submit their public keys so that election setup can begin. The administrator can then then check the election's status, as shown in image 6.

## Implementation of DEMOS Voting

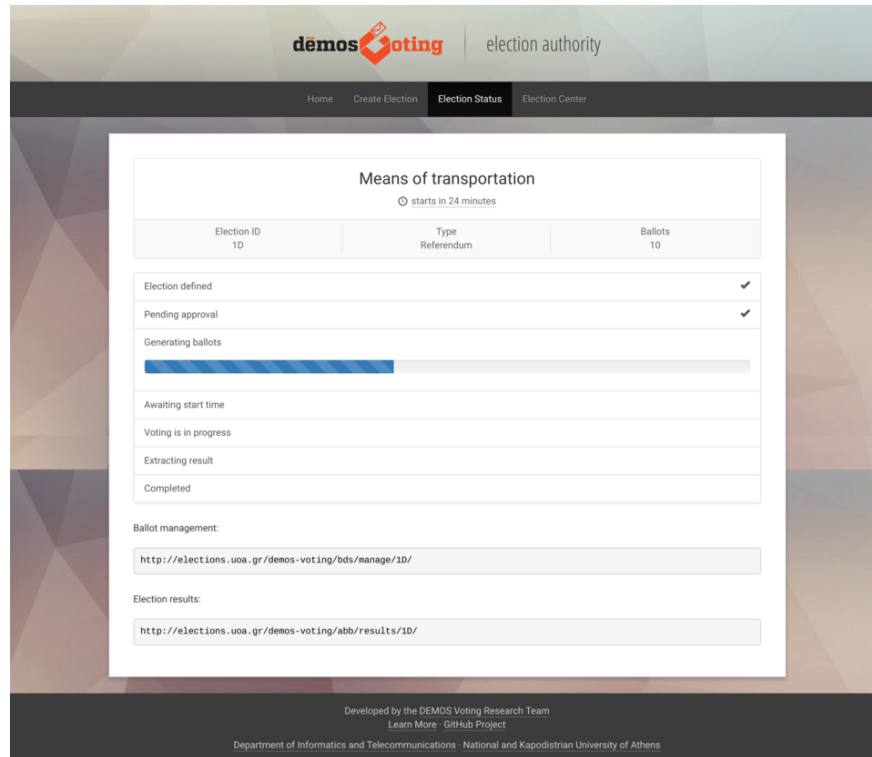


Image 6: Checking the election's setup progress

### 3.3.1.8 Browsing the list of elections

A user can browse the list of elections to which they have administrative access, as shown in image 7.

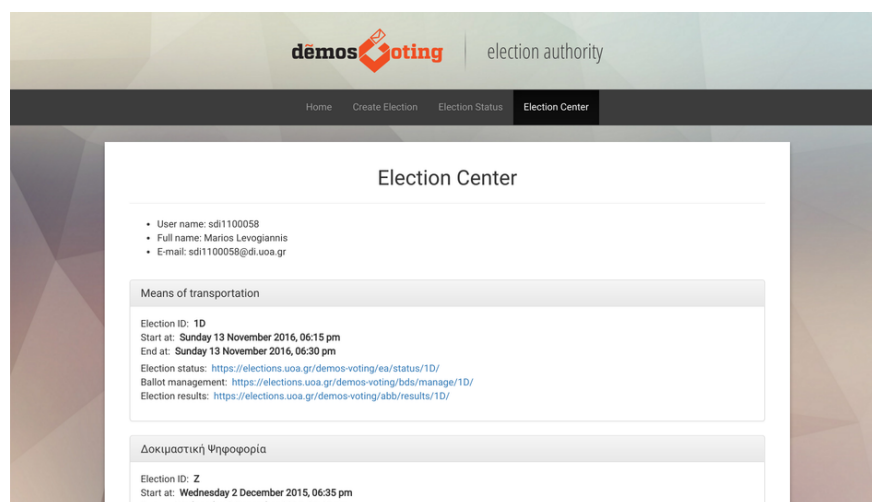


Image 7: List of elections

### 3.3.2 Ballot Distribution Server

#### 3.3.2.1 Selecting the distribution method

The administrator is responsible for distributing the ballots, as shown in image 8.

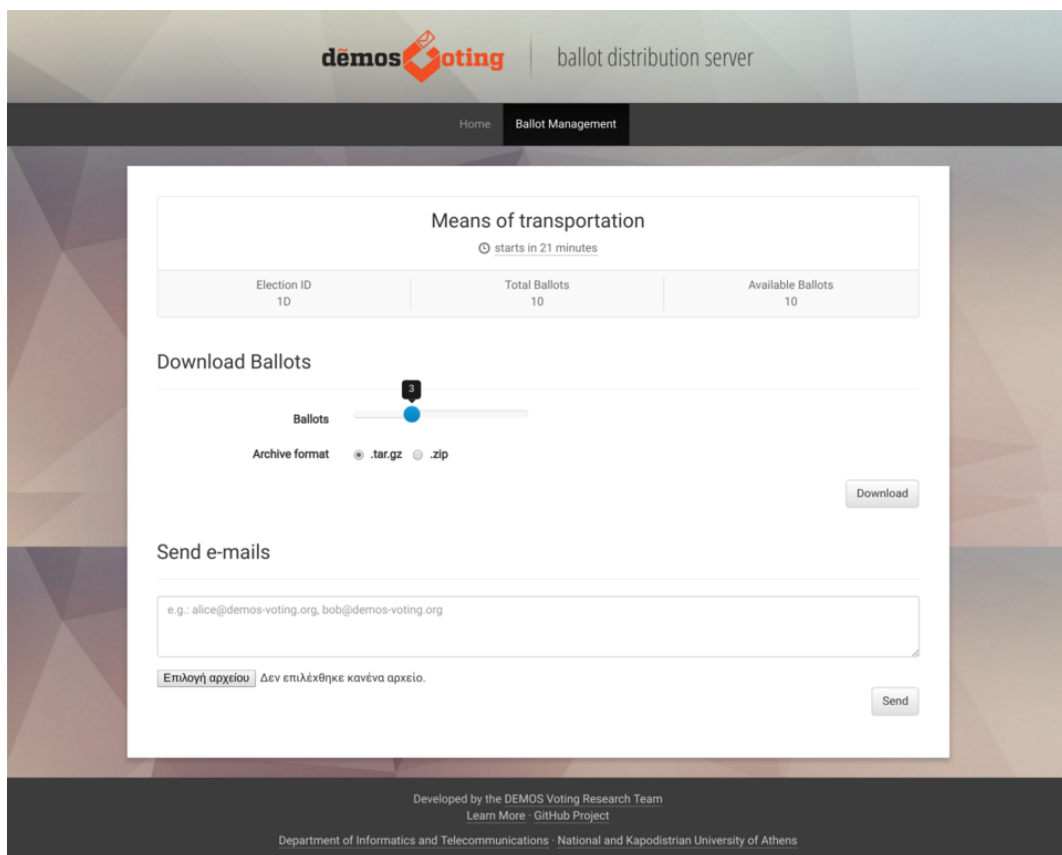


Image 8: Ballot distribution

The administrator may choose to:

- Download a random subset of the ballots.
- Send ballots to voters by email.
- Specify a number of constraints for registered users.

Ballots can be downloaded in *.tar.gz* or *.zip* archive format. Afterwards, they have to be distributed manually, for example they can be printed and sent to a polling station.

Another option is ballots to be distributed by email. The administrator can specify a list of comma-separated email addresses, or upload a large file in CSV format. The actual distribution will be handled by the system.

Finally, any registered user can request a ballot. The administrator has to specify constraints on the users that can participate in the election. These may include groups that a user belongs to, or attributes that a user may have (e.g. obtained from an LDAP server).

### 3.3.3 Virtual Ballot Box

#### 3.3.3.1 Presentation of the ballot

After a voter has received their ballot, they can proceed to the voting booth. Image 9 shows part A of ballot 105 for the referendum of subsection 3.3.1.3. Part B of this ballot, as well as a ballot for the election of subsection 3.3.1.5 can be found in Annex I.

**Serial number: 105** **Security code: ZQE1**

**Election Name:** Means of transportation  
**Election ID:** 1D

---

**Question 1:** What is your favorite means of transportation?

Option	Vote-code	Receipt
Car	2	Q5A6J2KW
Motorcycle	5	KWZJDR4C
Bus	3	4E7QM9G2
Train	1	FM68Q8YC
Bicycle	4	M65Z688F




**Question 2:** Should cars be banned from the city center?

Option	Vote-code	Receipt
Yes	2	3BXANJ8Z
No	1	MC759KK2

---

**Virtual Ballot Box:**  
<https://elections.uoa.gr/demos-voting/vbb/1D/XMNX2V6K4JFTQJ1MQEC0Y4ZPYEYK/>

**Audit and Results:**  
<https://elections.uoa.gr/demos-voting/abb/1D/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 9: Sample ballot

All ballots are double ballots, i.e. they consist of two parts that are labeled by the tags A and B. Both ballot parts are equivalent, but only one of the two can be cast. Each ballot part has an URL that links to the voting booth, which includes the election ID and a unique token. The information encoded by the token is the ballot's serial number, the part's tag (A or B) and a random credential for authenticating that ballot part (no user registration is required). The voter can access the voting booth with any of the following methods:

- Click on the URL, or copy and paste the URL in their web browser.
- Use the camera of their mobile device to scan the QR code which encodes the URL.
- If any of the above does not work, they will have to type the URL into their web browser's address bar.

The voter has to randomly choose (e.g. flip a coin) which of the two to use for voting and which to use for auditing.

### 3.3.3.2 About the security code

After following the URL of the ballot part selected for voting, the voter is presented with a single-page application. The first dialog prompts them to type their ballot part's security code, as shown in image 10. The security code can be found on the top-right of each ballot part.

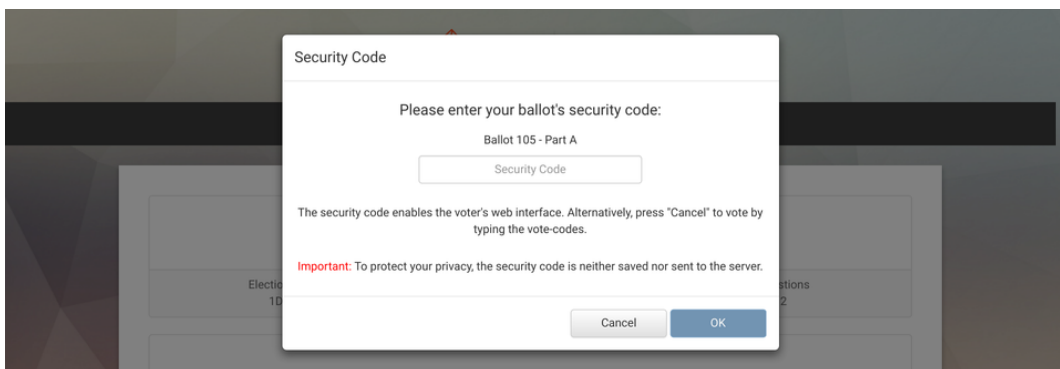


Image 10: Security code input

The security code will be used to “undo” the shuffling of the vote-codes and enable a user-friendly voting interface. The value of the security code is neither saved nor sent to the server, but all the processing happens on the voter's web browser. Alternatively, the voter may choose not to provide their security code (e.g. they may be afraid that their system is controlled by a malicious entity). In that case, they will be prompted to type the vote-codes that correspond to the options that they wish to vote for.

### 3.3.3.3 Filling in the ballot

The voter can now fill in the ballot, selecting the desired options. To navigate between questions they can use the arrows to the left and right of the question's options. This is shown in image 11.

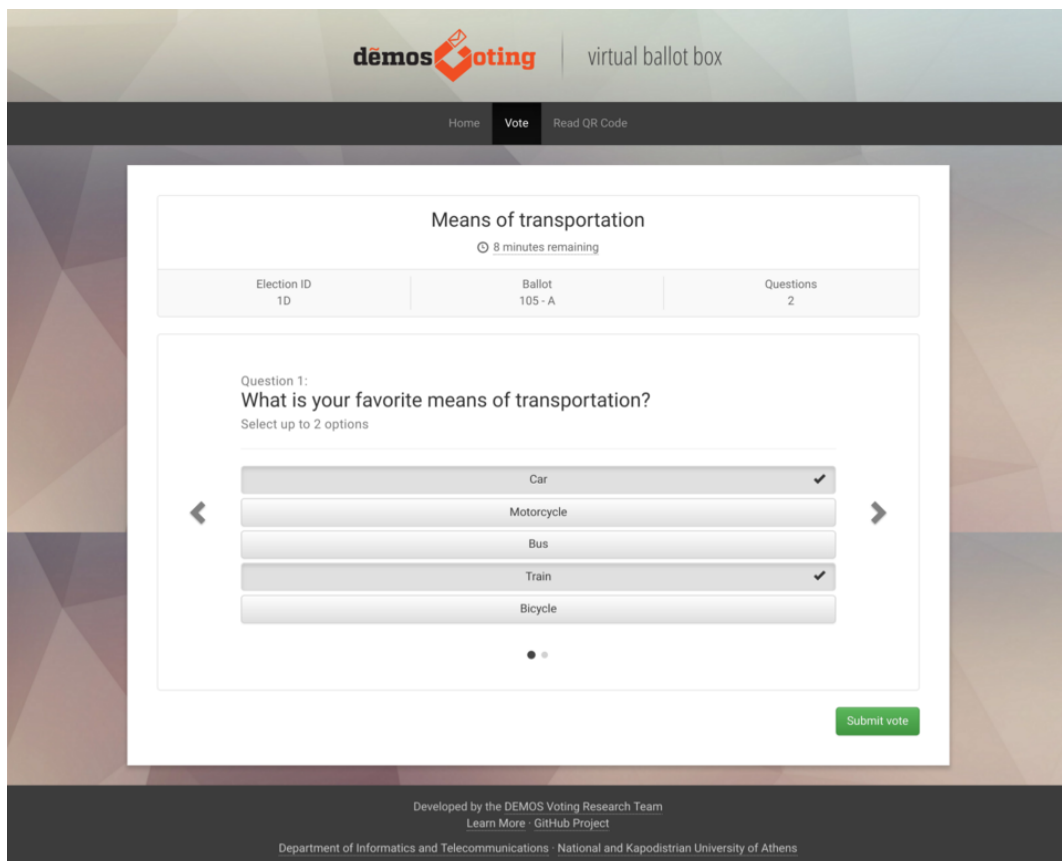


Image 11: Selecting options

### 3.3.3.4 Confirming the vote-codes

After the voter has filled in the ballot, they have to click *Submit*, otherwise their vote will not be recorded. A dialog will prompt them to review their selections and verify that the vote-codes match those on their ballot, as shown in image 12. If they do, the voter can click *Confirm* and their vote will be permanently cast.

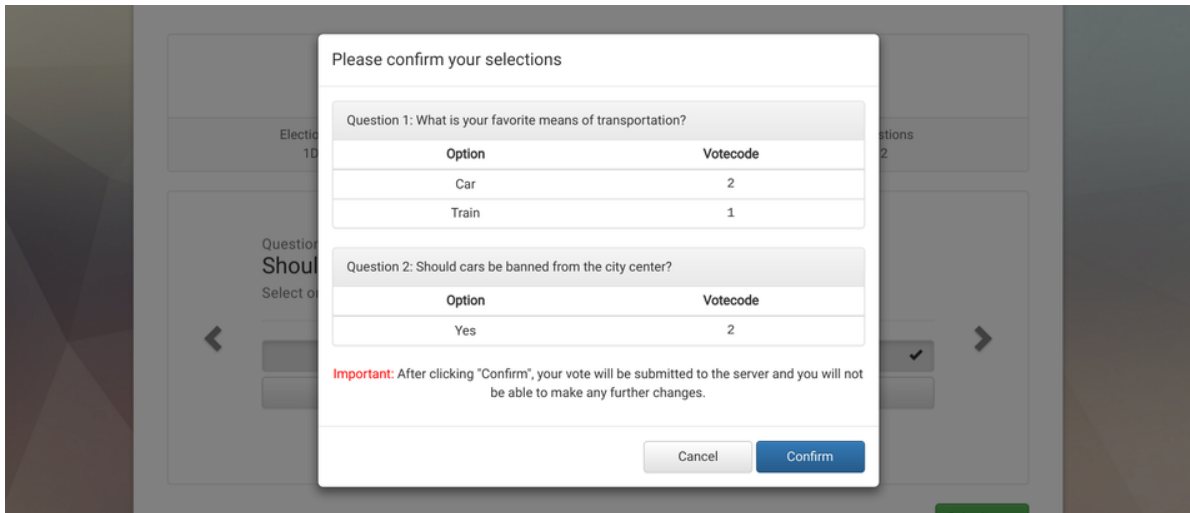


Image 12: Confirming vote-codes

### 3.3.3.5 Verifying the receipts

Upon confirmation, the next dialog will inform the voter that their ballot was cast successfully, as shown in image 13. Now the voter has to verify that the receipts returned by the server match those on their ballot. This is to ensure that the selected options have indeed been recorded.

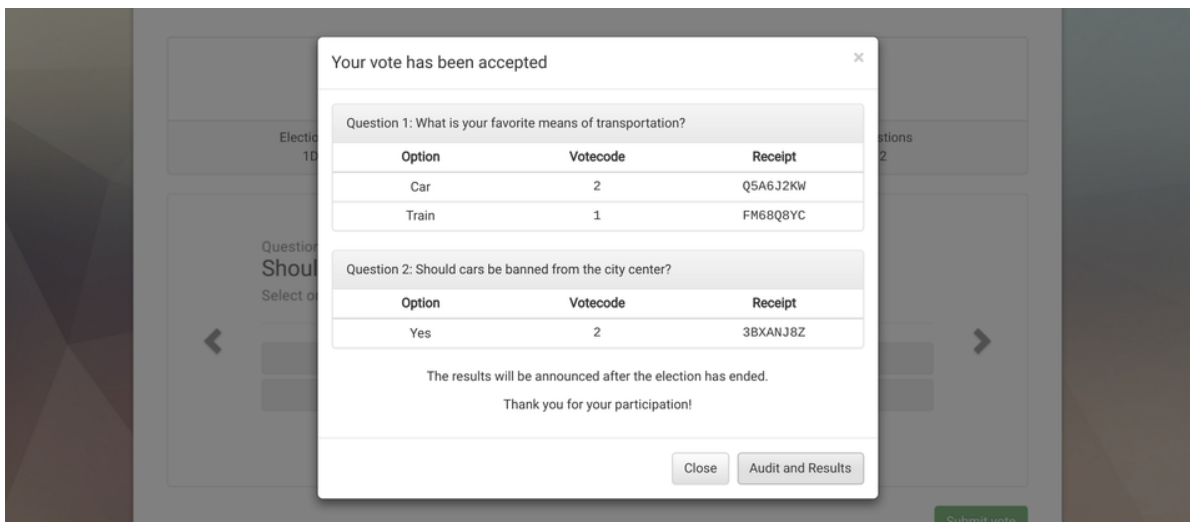


Image 13: Verifying receipts



### 3.3.4 Audit and Results

#### 3.3.4.1 Election results

After the voting period ends, the election committee has to compute the tally and generate the verification proofs using their secret keys. The results are then posted on the bulletin board, as shown in image 14.

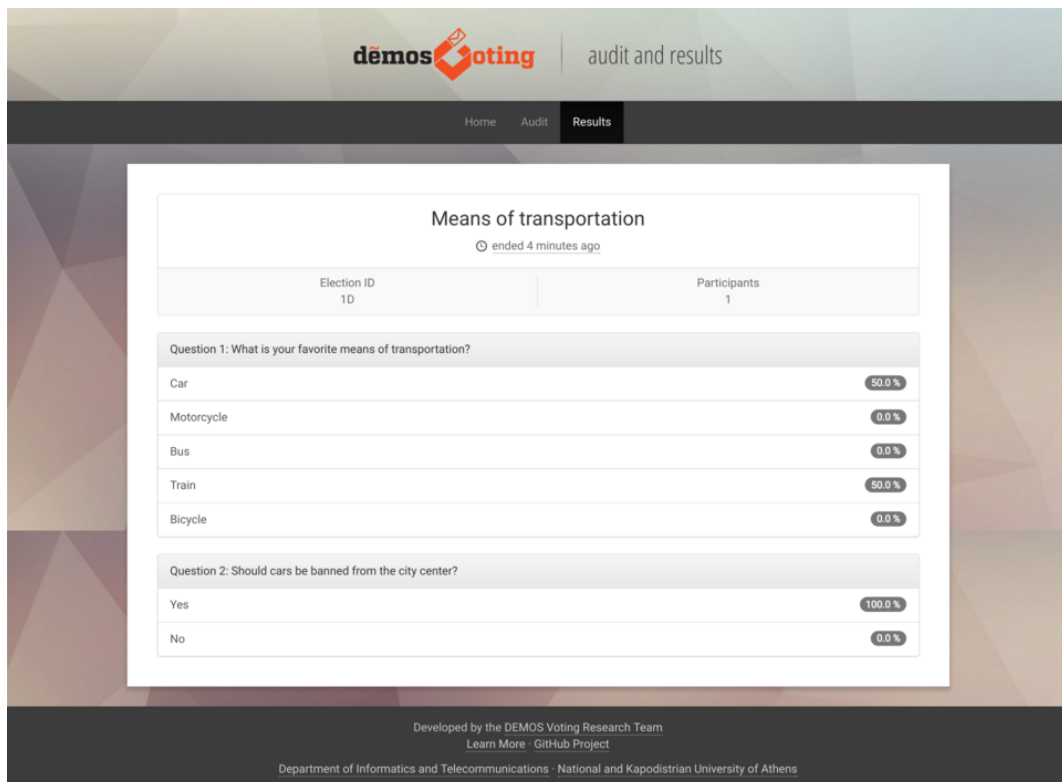


Image 14: Referendum results

#### 3.3.4.2 Verification phase

In the verification phase, voters can verify that their selections have been recorded properly, as shown in image 15. The easy-to-use interface makes it possible for non-skilled users to audit their ballots.

## Implementation of DEMOS Voting

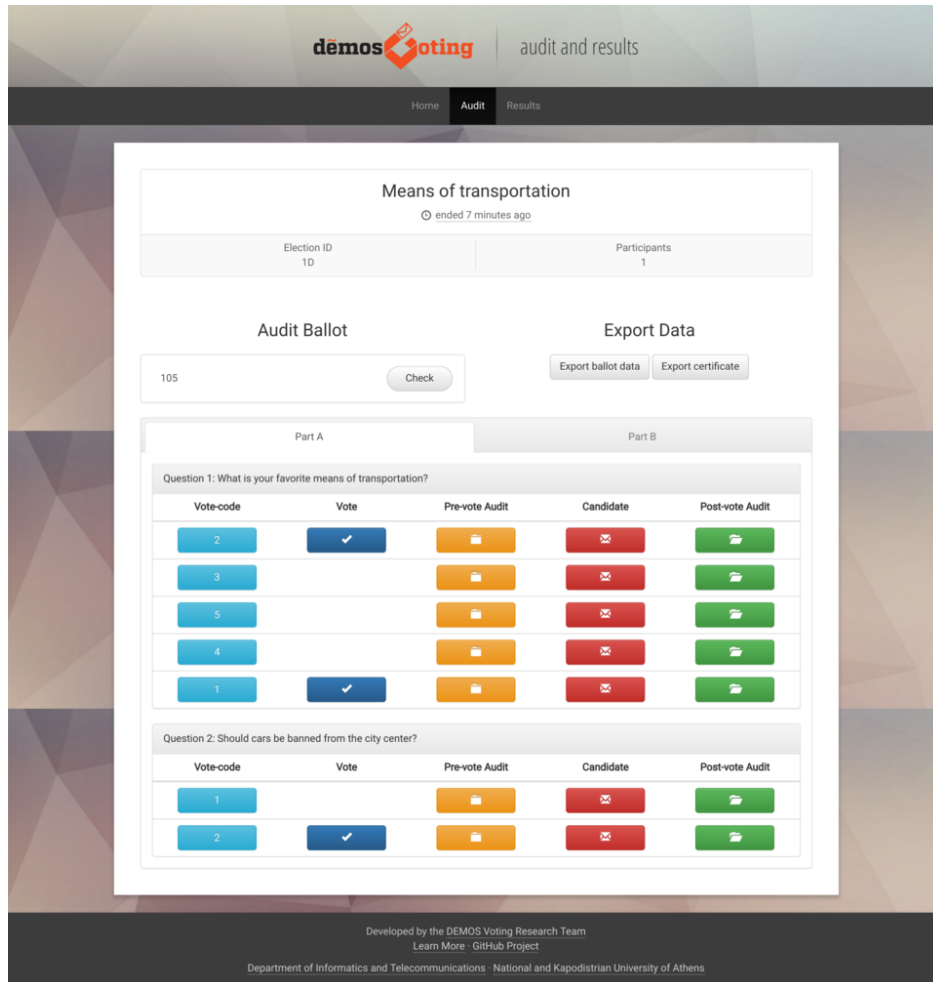


Image 15: Auditing the ballot part that was cast

The voter may also check the consistency of their receipt with the audit information, as shown in image 16. It is obvious that this verification step does not reveal the vote, even to someone that has access to the receipts returned by the Virtual Ballot Box. The reason is that the cast vote-code alone does not leak any information about the associated option.

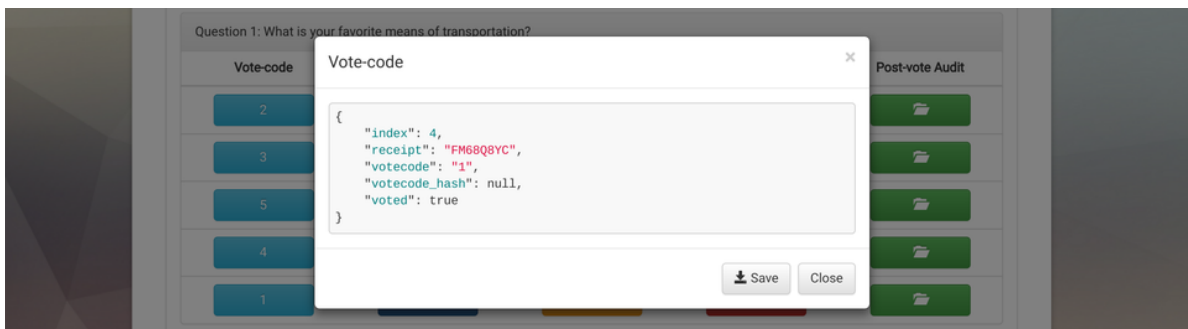


Image 16: Example of audit information in JSON format

Furthermore, the Election Authority opens the commitments and restores the original ordering of the vote-codes in the ballot part that the voter selected for auditing, as shown in image 17.

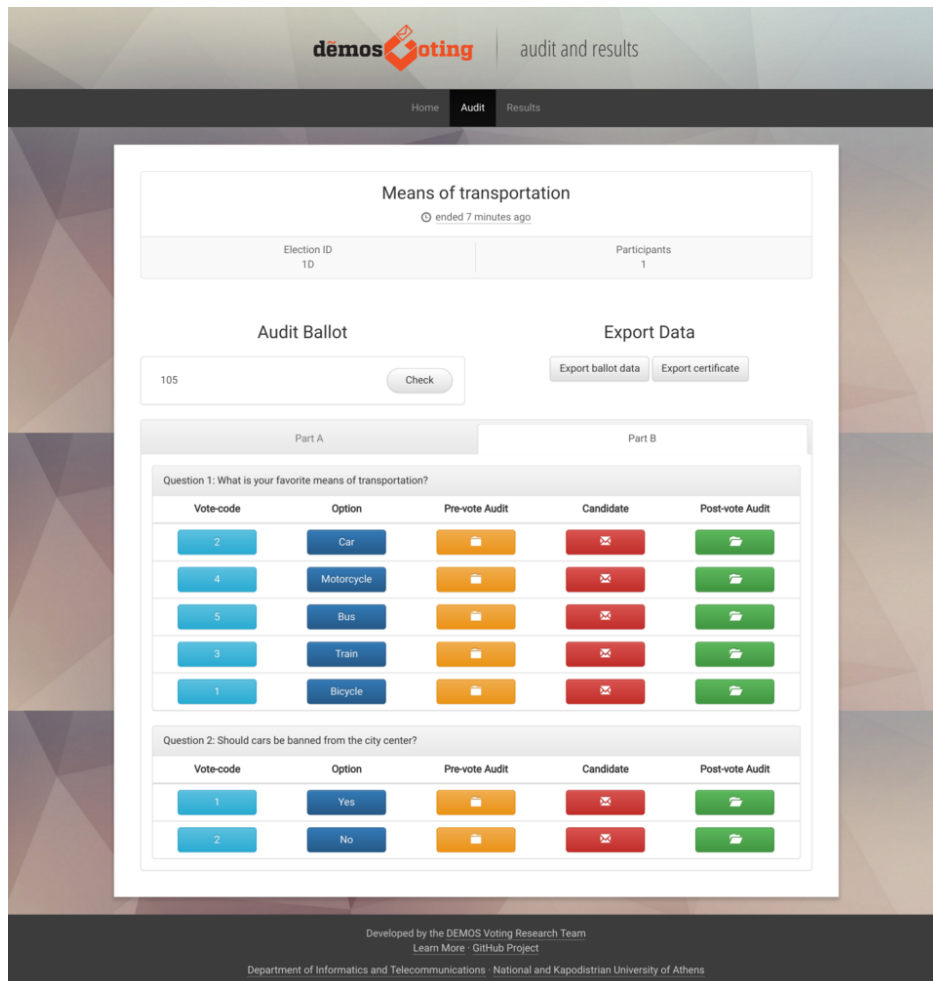


Image 17: Auditing the ballot part that was not cast

As in the verification of the voting part, the entirely opened auditing part does not reveal the vote to anyone because the two ballot parts are completely independent. It only serves as a check that the correspondence of the vote-codes and options in this part has not been tampered with.

### 3.3.4.3 Access to audit information

Programmatic access to all data posted on the bulletin board is provided by a REST API. Using this API, any third-party can verify the election result or individual ballots. Automated tools can be developed for this purpose. As described in the previous subsections, this would not compromise the voters' privacy.

The following HTTP request will fetch ballot 100 of election 1D:

```
GET https://elections.uoa.gr/demos-voting/abb/api/elections/1D/
ballots/100/
```

Response is in JSON format:

```
{
  "serial_number": 100,
  "cast_at": null,
  "parts": [
    "https://elections.uoa.gr/demos-voting/abb/api/elections/1D/
ballots/100/parts/A/",
    "https://elections.uoa.gr/demos-voting/abb/api/elections/1D/
ballots/100/parts/B/"
  ]
}
```

The ballot object used in this example may not contain any significant information, but it has links to part objects A and B, which in turn will have links to the question and option objects that hold all the necessary audit information.

### 3.4 Deployment

#### 3.4.1 National Elections 2015 - Pilot Experiment

An early version of the e-voting system was tested during the National Elections 2015 in Athens, Greece. During the experiment, which took place on Sunday, 25 January 2015, a number of 400 voters voted in special polling stations that were placed outside of the two main polling places.



Image 18: Polling station

### 3.4.2 University of Athens

The system has been installed for the needs of the University of Athens. The instances of the main components (EA, BDS, VBB and ABB) are installed on different Linux Containers (LXCs), running on a CentOS server, which acts as a reverse proxy server. The Apache HTTP Server is used as the web server and the PostgreSQL database is used for data storage. It has not been officially tested yet.

- <https://elections.uoa.gr/demos-voting/>

### 3.4.3 Labour Institute of the Greek General Confederation of Workers

The system has also been deployed by the *Labour Institute of the Greek General Confederation of Workers* (INE-GSEE), as part of the programme *Digital portal, e-Democracy and electronic learning*. An official presentation took place on Wednesday, 25 November 2015, at Titania Hotel, Athens.

The security policy of the organization requires that the voting system is only available from inside their private network. As a result, its current status is unknown.

## 4. CONCLUSIONS AND FUTURE WORK

DEMOS Voting is a new, end-to-end verifiable, e-voting system. E2E verifiability is a strong level of security for election systems that has been widely accepted as a fundamental requirement for their adoption. The system can be used by organizations or communities that require trustworthy elections.

The e-voting system is open source software and is built on top of other open source software components. Anyone can study and review the code. However this is not sufficient for reliable elections, as voters cannot be sure about which software is actually running on the election server. It is obvious that cryptographic verifiability is absolutely necessary.

Unfortunately, DEMOS Voting does not come without its deficiencies. It uses a centralized EA which maintains all secrets throughout the entire election procedure, a centralized VBB which collects votes and a centralized ABB where the EA stores the result and the verification data. In other words, there are multiple single points of failure. A distributed version of the cryptographic implementation of the system can be the solution to the problem, as described in [31].

## ABBREVIATIONS - ACRONYMS

E-Voting	Electronic Voting
E2E	End-to-End
1V1V	One Voter, One Vote
EA	Election Authority
BDS	Ballot Distribution Server
VBB	Voter Bulletin Board
ABB	Audit Bulletin Board
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
JSON	JavaScript Object Notation
HMAC	Hash-based Message Authentication Code
SHA-2	Secure Hash Algorithm 2
MCF	Modular Crypt Format
PBKDF2	Password-Based Key Derivation Function 2
CSV	Comma-Separated Values
CAS	Central Authentication Service
LDAP	Lightweight Directory Access Protocol
URL	Uniform Resource Locator
REST	Representational State Transfer
API	Application Programming Interface

## **ANNEX I**

In the following pages, two ballots are presented:

- A sample ballot for the referendum of subsection 3.3.1.3 (images 19 to 20).
- A sample ballot for the election of subsection 3.3.1.5 (images 21 to 28).



**Serial number: 105**

**Security code: ZQE1**

**Election Name:** Means of transportation

**Election ID:** 1D

---

**Question 1:** What is your favorite means of transportation?

Option	Vote-code	Receipt
Car	2	Q5A6J2KW
Motorcycle	5	KWZJDR4C
Bus	3	4E7QM9G2
Train	1	FM68Q8YC
Bicycle	4	M65Z688F

**Question 2:** Should cars be banned from the city center?

Option	Vote-code	Receipt
Yes	2	3BXANJ8Z
No	1	MC759KK2

---

**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1D/XMNX2V6K4JFTQJ1MQEC0Y4ZPYEYK/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1D/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 19: Ballot 105 - Part A

**Serial number: 105**

**Security code: 7V57**

**Election Name:** Means of transportation

**Election ID:** 1D

---

**Question 1:** What is your favorite means of transportation?

Option	Vote-code	Receipt
Car	2	V7PBX5ZB
Motorcycle	4	AT9CYYNX
Bus	5	C4Q6P1D5
Train	3	F8F1470X
Bicycle	1	KCQ5JRQJ

**Question 2:** Should cars be banned from the city center?

Option	Vote-code	Receipt
Yes	1	JAV0FXSD
No	2	FEN88ZQQ

---

**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1D/ZMYH9H6N7E7BBQJSTJY9ECP7HY5Q/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1D/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 20: Ballot 105 - Part B

**Serial number: 109**

**Security code: 96951707**

**Election Name:** National Elections

**Election ID:** 1E

**Party:** Pink

**Vote-code:** 6V0J-Y4FT-02RK-XRT5

**Receipt:** EQGY7AVB

Candidate	Vote-code	Receipt
Candidate 1	888M-QY8C-30Z1-0KGQ	933Q9D38
Candidate 2	DRHV-0YRX-9A8R-PEQS	KANXGRRB
Candidate 3	7SMF-VD2X-6KFH-PDPV	TC9EM1TE
Blank	1GZH-ZXQX-9JQX-SB8V	8XDVYGRC
Blank	6SZ3-10MQ-FTE6-Z73V	27AW715E
Blank	JVDC-94E0-CY7P-AADG	T4XWE413

**Party:** Blue

**Vote-code:** WARX-6CDY-WSM5-ESBG

**Receipt:** WZYK8BJ8

Candidate	Vote-code	Receipt
Candidate 4	1489-W8ME-6Q82-WQPT	5MRHYYGX
Candidate 5	HH2A-KAM4-B0EP-7AZ5	5DV53PTG
Candidate 6	57NT-6TXX-FVSJ-2KW2	08TNB9F1
Candidate 7	T63E-VCNR-26V9-APE4	Z9PJAZA5
Candidate 8	21PH-M48E-EH28-G1WW	MGVZVMEG
Blank	BMSM-8WX2-PAZG-M0KC	NPWACNS2

**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/DPK89J28T2CRJ2EW7PYR4GMQH6R4/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 21: Ballot 109 - Part A (page 1/4)

**Serial number: 109**

**Security code: 96951707**

**Election Name:** National Elections

**Election ID:** 1E

-----  
**Party:** Green

**Vote-code:** 7RHB-WTXX-W8MG-1FYT

**Receipt:** PTFT5K3K

Candidate	Vote-code	Receipt
Candidate 9	HXX1-PQ7E-7KVX-AGQM	RVQDHTJZ
Candidate 10	M38M-CK87-C4Z1-R6EP	FES1WXN2
Candidate 11	GMBJ-P6VZ-N2P9-TD7P	150XY5B6
Candidate 12	YYN6-642P-6EQ9-A8WG	66ER9S28
Blank	GH33-E796-W5Z1-WDNW	BMGVB50Z
Blank	9GA9-SBPV-BBB5-DCPN	X2GGR1R6

**Party:** Red

**Vote-code:** DH9Y-X11F-0CQR-HXNQ

**Receipt:** SPGSPJ1

Candidate	Vote-code	Receipt
Candidate 13	RRNY-V8KW-V32D-A19T	CF586HVH
Candidate 14	4RVJ-7G3M-RY7V-NZCE	FHM9ACQT
Candidate 15	012H-C71C-ZX47-9VAH	BY546NZF
Candidate 16	V4NW-RQXB-S2GZ-DFD6	YXZMEF0V
Candidate 17	3WBQ-J56G-993G-HQYM	T3VRFY2C
Blank	FC2N-KXNG-X4RM-2KV8	KKTM7K3W

-----  
**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/DPK89J28T2CRJ2EW7PYR4GMQH6R4/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 22: Ballot 109 - Part A (page 2/4)

**Serial number: 109**

**Security code: 96951707**

**Election Name:** National Elections

**Election ID:** 1E

**Party:** Brown

**Vote-code:** XKZD-MS7B-WK9X-2AYS

**Receipt:** NMR33RB4

Candidate	Vote-code	Receipt
Candidate 18	N95P-4GPP-H97V-60SW	1CGR8YEC
Candidate 19	454K-KPC1-12N4-PYB7	13B27S68
Candidate 20	PDDS-8NYM-XADY-D4VZ	0XC6PE53
Blank	PAF4-8EVR-7VYP-YTZD	Q286FWCH
Blank	6KEF-93WG-WGR6-S9TD	RDMH57FJ
Blank	7HAM-E69R-MF3F-D3MJ	6WS9EHQH

**Party:** Cyan

**Vote-code:** 4DME-EZK3-NDCS-GFG2

**Receipt:** D3Z9T31E

Candidate	Vote-code	Receipt
Candidate 21	Q33H-HXY3-AH41-XJDR	2TYEEYN7
Candidate 22	SRYE-wWTW-3QY8-NDQ1	EQ9Q1H2J
Candidate 23	9A1E-KBS4-4D8Q-4JVH	RJG4DMYQ
Candidate 24	WP6Z-14A2-W41Q-ZE39	RPYSEZ4H
Candidate 25	E1YK-N293-PP9W-6JTG	ZQNA4WJ1
Blank	WQ50-HTAC-ZJN4-H4X9	JRHWO49D

**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/DPK89J28T2CRJ2EW7PYR4GMQH6R4/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 23: Ballot 109 - Part A (page 3/4)

**Serial number: 109**

**Security code: 96951707**

**Election Name:** National Elections

**Election ID:** 1E

-----  
**Party:** Orange

**Vote-code:** XG0N-B6Q7-9EWM-GZHJ

**Receipt:** S21ZD0JR

Candidate	Vote-code	Receipt
Candidate 26	GMFJ-FV54-D7AN-R7XN	BWXWX1XT
Candidate 27	0HYF-27RH-MS6Z-5ZJC	K0TGE0P3
Candidate 28	7ZCG-F421-Tw21-HHY1	Y4JK94Q1
Blank	YMS8-GQZN-JHT7-QSNT	CKB5K4PK
Blank	AFWD-TSMD-V5RV-T5N8	9YQ1XDSP
Blank	294S-AQH3-KPYD-4Z8K	M5S3G0N3

**Party:** Blank

**Vote-code:** H9HG-WKQX-HTTN-73RC

**Receipt:** J23JGV20

Candidate	Vote-code	Receipt
Blank	P1SH-NSJG-RFXE-CJ8G	0NV46B6E
Blank	8G56-VE0R-X6YG-D4RX	04WBSN33
Blank	P5T3-DMNK-75R4-PTF2	ZYFSGQ4Q
Blank	04D0-C0KA-Z5RA-S52Q	NSV8ZHYJ
Blank	9H6F-F1X4-V8R3-CCVG	D4PJDC4A
Blank	SK7R-Q920-TKVH-H54P	C9SE1JJ8

-----  
**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/DPK89J28T2CRJ2EW7PYR4GMQH6R4/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 24: Ballot 109 - Part A (page 4/4)

**Serial number: 109**

**Security code: 15532681**

**Election Name:** National Elections

**Election ID:** 1E

-----  
**Party:** Pink

**Vote-code:** 5RS6 -XEYB -6E57 -2WPM

**Receipt:** FZHT5XZ5

Candidate	Vote-code	Receipt
Candidate 1	2MRT -AASW -3147 -BWDF	R60PTGFX
Candidate 2	AXVA -V17E -YC34 -XMAA	TYDH4QFB
Candidate 3	Y61H -B2R9 -JA93 -P99M	518HKZKP
Blank	KJMH -EM50 -VNHS -1V3D	K5BE6207
Blank	WG1T -RDYD -H022 -CZP6	05CT0136
Blank	413Y -365G -729F -EG2H	R1WBP098

**Party:** Blue

**Vote-code:** XP34 -TY43 -CTTV -SVED

**Receipt:** J4GZ8J58

Candidate	Vote-code	Receipt
Candidate 4	P0EN -GTJ8 -QJP9 -9PDV	6W5V43NB
Candidate 5	0M5Q -366M -PJZG -8RHE	M4E65NFQ
Candidate 6	VJSV -Y9YX -69TR -T4CX	PGC0KAD6
Candidate 7	B6EB -A2EX -E8S2 -W6E2	XZAVGJ5E
Candidate 8	7AG0 -N96N -6Y7F -91P9	PWCEAV4T
Blank	WR33 -E71E -2S4X -8NH3	55P7V6YC

-----  
**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/3PWK0JVSNSKB0X0P4MV3XP1CS36Z/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 25: Ballot 109 - Part B (page 1/4)

**Serial number: 109**

**Security code: 15532681**

**Election Name:** National Elections

**Election ID:** 1E

**Party:** Green

**Vote-code:** WP9X-6G12-54E9-7T8P

**Receipt:** RY9A1WRP

Candidate	Vote-code	Receipt
Candidate 9	6HTY-3XAH-CEPM-4GTE	WTF98BWG
Candidate 10	PBM8-GFZ0-SFB1-S2RM	V9G6DES6
Candidate 11	36K2-CSZR-10ER-M324	0FBJQDJH
Candidate 12	GVVC-HE8S-BBGR-B4DJ	R577N27R
Blank	9M2M-AXB2-WYJW-22VS	3RP86CTP
Blank	F6X3-6RYG-8514-SKC6	GASC159C

**Party:** Red

**Vote-code:** 2TXY-HZ68-RP35-DG94

**Receipt:** 2REG5TF6

Candidate	Vote-code	Receipt
Candidate 13	W4EX-H709-YJF3-3C84	R3AVM35K
Candidate 14	FEPX-GQAC-3KCM-4BWT	DK253SNN
Candidate 15	R8HD-NB8Y-Y0D1-0752	TGZR5SSD
Candidate 16	02V4-Q300-N7BH-A0T4	7G9PXZMR
Candidate 17	HB5X-X7RT-2WJ9-1WKF	HNW5FE78
Blank	EQDP-VSC6-5B1H-3GRQ	9ZPJXPTW

**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/3PWK0JVSNSKB0X0P4MV3XP1CS36Z/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 26: Ballot 109 - Part B (page 2/4)



**Serial number: 109**

**Security code: 15532681**

**Election Name:** National Elections

**Election ID:** 1E

**Party:** Brown

**Vote-code:** T8XT-FRSD-TN8V-W2V8

**Receipt:** 5JJ92Y7H

Candidate	Vote-code	Receipt
Candidate 18	V9JB-HBJ9-G114-RRCQ	K97FPG9M
Candidate 19	9ZKW-SWZE-T2AH-DRQQ	ME85RDGS
Candidate 20	Q46C-PJVC-P941-VE3J	6DEMYE0Y
Blank	CEX6-CW4T-PZ4J-8EHY	0RKPK7QY
Blank	J20S-JNVQ-YAN1-WKE6	F2XN0MX7
Blank	6PRM-JEDW-GC0W-60ZH	D4JGPZJP

**Party:** Cyan

**Vote-code:** FHTF-C5GM-S1KG-KS2W

**Receipt:** BV5MCT15

Candidate	Vote-code	Receipt
Candidate 21	E864-J2XH-1BMV-4D52	0JN2NFZT
Candidate 22	5E98-8TQT-SWZE-K11F	8F55HXS6
Candidate 23	TMVD-WFH4-J8FB-JH66	P26V75GT
Candidate 24	GGDN-KGPQ-APB3-95ND	2KR1H15W
Candidate 25	PSH9-7S1C-F9A0-GVHE	WTGGGH4E
Blank	QQNQ-370Q-002R-YBGN	A5M2TV0A

**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/3PWK0JVSNSKB0X0P4MV3XP1CS36Z/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 27: Ballot 109 - Part B (page 3/4)

**Serial number: 109**

**Security code: 15532681**

**Election Name:** National Elections

**Election ID:** 1E

**Party:** Orange

**Vote-code:** 8NWF-PFYF-X1JA-SRD8

**Receipt:** 0HXJ7T9V

Candidate	Vote-code	Receipt
Candidate 26	0BBF-TJ8T-BMXS-4QQD	6SRBFF52
Candidate 27	YWRE-SYFB-64XE-KVAA	WYDEB0JP
Candidate 28	EH7B-2BSE-2ZFY-HJ76	WS2CKHS6
Blank	HXDA-RMF4-KFXT-R2EG	1K2AN7M4
Blank	ZW19-623Q-TMPK-T3XS	EE7FTQ39
Blank	XX7M-5TKX-QE5A-VMRV	2HC2TYK7

**Party:** Blank

**Vote-code:** R3N8-YNVK-VYNY-JJJN

**Receipt:** ZQSKKDVH

Candidate	Vote-code	Receipt
Blank	W37R-Y7B3-T024-A1QQ	7W8TSK3M
Blank	F027-FSTS-7DCZ-X7D7	5FZW6GNW
Blank	V87F-FA73-T1KE-BPX2	F7AM5CY8
Blank	R1R3-E47J-Z60N-3EX6	E0C6G26H
Blank	6EMX-RD7E-QF4A-MWHG	BV6SP8RA
Blank	7J19-2H3G-02EZ-822E	HC1SGE73

**Virtual Ballot Box:**

<https://elections.uoa.gr/demos-voting/vbb/1E/3PWK0JVSNSKB0X0P4MV3XP1CS36Z/>

**Audit and Results:**

<https://elections.uoa.gr/demos-voting/abb/1E/>



Every ballot consists of two parts, A and B. Please use one of them to vote and keep the other one for the verification process.

Image 28: Ballot 109 - Part B (page 4/4)

## BIBLIOGRAPHY

- [1] N. Chondros, A. Delis, D. Gavatha, A. Kiayias, C. Koutalakis, I. Nicolacopoulos, L. Paschos, M. Roussopoulos, G. Sotirelis, P. Stathopoulos, P. Vasilopoulos, T. Zacharias, B. Zhang, and F. Zygoulis, “Electronic voting systems - from theory to implementation,” in *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*, pp. 113–122, 2013.
- [2] R. Küsters and T. Truderung, “An epistemic approach to coercion-resistance for electronic voting protocols,” in *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*, pp. 251–266, 2009.
- [3] A. Juels, D. Catalano, and M. Jakobsson, “Coercion-resistant electronic elections,” *IACR Cryptology ePrint Archive*, vol. 2002, p. 165, 2002.
- [4] S. Delaune, S. Kremer, and M. Ryan, “Coercion-resistance and receipt-freeness in electronic voting,” in *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), 5-7 July 2006, Venice, Italy*, pp. 28–42, 2006.
- [5] A. Escala, S. Guasch, J. Herranz, and P. Morillo, “Universal cast-as-intended verifiability,” in *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, pp. 233–250, 2016.
- [6] S. Popoveniuc, J. Kelsey, A. Regenscheid, and P. L. Vora, “Performance requirements for end-to-end verifiable elections,” in *2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '10, Washington, D.C., USA, August 9-10, 2010*, 2010.
- [7] C. A. Neff, “Practical high certainty intent verification for encrypted votes,” 2004.
- [8] J. Benaloh, R. Rivest, P. Y. Ryan, P. Stark, V. Teague, and P. Vora, “End-to-end verifiability,” *arXiv preprint arXiv:1504.03778*, 2015.
- [9] S. Kremer, M. Ryan, and B. Smyth, “Election verifiability in electronic voting protocols,” in *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, pp. 389–404, 2010.
- [10] R. Küsters, T. Truderung, and A. 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*, pp. 538–553, 2011.

- [11] A. Kiayias, T. Zacharias, and B. Zhang, “End-to-end verifiable elections in the standard model,” *IACR Cryptology ePrint Archive*, vol. 2015, p. 346, 2015.
- [12] D. Chaum, “Surevote: technical overview,” in *Proceedings of the workshop on trustworthy elections (WOTE’01)*, 2001.
- [13] D. Gritzali, “Principles and requirements for a secure e-voting system,” *Computers & Security*, vol. 21, no. 6, pp. 539–556, 2002.
- [14] S. Delaune, S. Kremer, and M. Ryan, “Verifying privacy-type properties of electronic voting protocols,” *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009.
- [15] J. C. Benaloh and D. Tuinstra, “Receipt-free secret-ballot elections (extended abstract),” in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pp. 544–553, 1994.
- [16] H. L. Jonker and E. P. de Vink, “Formalising receipt-freeness,” in *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, pp. 476–488, 2006.
- [17] T. Moran and M. Naor, “Receipt-free universally-verifiable voting with everlasting privacy,” in *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pp. 373–392, 2006.
- [18] Python Software Foundation, “Python.” <https://www.python.org/>.
- [19] Django Software Foundation, “Django.” <https://www.djangoproject.com/>.
- [20] B. Peterson, “Six.” <https://pythonhosted.org/six/>.
- [21] A. Solem and contributors, “Celery.” <http://www.celeryproject.org/>.
- [22] PyCA, “pyOpenSSL.” <http://www.pyopenssl.org/>.
- [23] G. Danezis, “petlib.” <https://github.com/gdanezis/petlib/>.
- [24] T. Christie, “Django REST framework.” <http://www.django-rest-framework.org/>.
- [25] ReportLab, “ReportLab PDF Library.” <https://www.reportlab.com/>.
- [26] Bootstrap Core Team, “Bootstrap.” <https://getbootstrap.com/>.
- [27] jQuery Team, “jQuery.” <https://jquery.com/>.
- [28] Stanford Computer Security Lab, “Stanford Javascript Crypto Library.” <https://crypto.stanford.edu/sjcl/>.
- [29] D. Crockford, “Base32 Encoding.” <http://www.crockford.com/wrmg/base32.html>.

- [30] Passlib, “Modular Crypt Format.” [https://pythonhosted.org/passlib/modular\\_crypt\\_format.html](https://pythonhosted.org/passlib/modular_crypt_format.html).
- [31] N. Chondros, B. Zhang, T. Zacharias, P. Diamantopoulos, S. Maneas, C. Patsonakis, A. Delis, A. Kiayias, and M. Rousopoulos, “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*, pp. 711–720, 2016.