ethlect.

An investigation into the Development of a Digital PR-STV Electoral Voting System.

Andrei Florian

andrei_florian@universumco.com

https://www.linkedin.com/in/andrei-florian

GitHub Paper

This paper is designed to accompany the project on GitHub

Date Released

6 January 2021

Table of Contents

Abstract

Elections are an integral part to a democratic system. They allow individuals to select representatives that share their goals and aspirations to run the polity. With roots in Athens and Ancient Rome, the ideal of empowering individuals to choose their leaders has developed into an important tradition, upheld in most countries around the world today.

The power and actualisation that elections provide people with is seen through countless organised protests and civil wars throughout history in the pursuit of universal suffrage (Wikipedia, 2022) and the fairness of elections.

Elections today look very different to what they once were but share the same founding principles and objectives once philosophised about in Aristotle's writings – the transfer of power and opinion to the individual and the reinforcement of people's trust and belief in democracy.

The way we voice our opinions naturally changes with time. Our vision for how elections should be carried out, who should be allowed to participate, and what defines a fair and just system shifts with advancements in knowledge and technology.

Ultimately one thing is certain – we will continue innovating how we carry out elections, enabling more people to voice their opinions with greater ease.

A Personal Note

My fascination with elections, what they represent, and how they are carried out began this summer while researching use cases of blockchain technologies. Almost all articles I read would discuss the potential applications in the electoral space and how recent breakthroughs in computing and cryptography allow for the running of more precise and efficient elections.

My life's ambition is to create impactful tools and systems that enhance the way

we interact with the world. In that sense, I was captivated by the potential of i-voting and wanted to dive deeper into the intersection between technology and elections and use the knowledge to create my own internet voting system.

After sleepless nights reading cryptography papers (more so trying to grasp them) and coding, I created a first-of-its-kind internet voting application tailored for Irish General Elections, that uses a mixture of existing cryptographic techniques, and some that I designed myself. I hope you enjoy the paper :).

Investigation Process

I started off my journey looking into the Irish Electoral System. I was interested in the inner workings of this system, namely the types of elections carried out, the electoral model used and what challenges elections in Ireland face.

After gaining an understanding of the system at present, I started researching how other countries have leveraged advancements in technology to improve the voting experience for both the voter and the government.

I focused on internet voting, and the technologies that enable the casting of digital ballots. I read about the benefits such systems bring together with criticisms and challenges they face.

I compiled all this knowledge together and developed the first end-to-end verifiable, internet voting application for PR-STV elections. The application proposes a novel hybrid approach to internet voting that provides elevated security in all layers of the system, something that has not yet been achieved by any solution. Finally, I rigorously tested the system's theory and actual app.

Paper Layout

This paper is split up into thirteen sections. I will start off by assessing the research conducted into elections in Ireland. I will then continue with research into internet voting as a concept and implementation. I will finish with fully presenting my application and delving into both the theory and its materialisation.

Links

The source code of the application and more details can be found at the GitHub link below:

Andrei-Florian/ethlect (github.com)

The Irish Electoral System

Ireland hosts direct elections by universal suffragette for four types of elections: presidential, Dáil Éireann, EU parliament and local government. All four of these elections are held using the PR-STV method (Wikipedia, 2022).

Ireland implements a traditional approach to ballot casting – voters can only cast their ballot in person, unless granted specific permission to vote otherwise. Ballots are not introduced into an electronic system at any point in the process. (Stuart-Mills, 2021)

Election Conduct and PR-STV

Ireland is unique in the sense that since the inauguration of the first Dáil, the Proportional Representation through Single Transferrable Vote method was used (PR-STV) (Wikipedia, 2022). Ireland is one of the only three countries in the world to use this system.

The country is split into multiple constituencies (3 for EU elections, 39 for Dáil elections as of 2020 (Houses of the Oireachtas, 2016)), in the case of all elections bar presidential, voters are presented with ballots consisting of candidates representing their constituency.

In the case of general elections, any number of candidates can express their intention to run. Each constituency defines the number of seats that are to be filled (around 1 for every 25,000 people (Citizens Information, 2021)).

When casting their vote, the voter can choose to vote for any number of the available candidates in preferential order by marking a number (1 to n, where n represents the number of candidates available) next to each candidate.

All paper ballots are then securely escorted to a centralised counting centre in each constituency. These ballots begin to be counted at 9AM the following day by count officials supervised by admin staff. The count begins by removing all spoilt ballots from the set and sorting the valid ballots by first preference.

The quota for each constituency is then calculated, this represents the number of votes a candidate needs to be elected.

$$Q = \frac{p}{s+1}$$

The equation above represents the calculation of a quota where p represents the valid poll count and s represents the number of seats in said constituency.

The first count commences for every constituency. In this count, all ballots are counted based on first preference. Should any candidate reach the quota for their constituency, the surplus votes are redistributed to the candidates that have not reached the quota in the first count.

Subsequent counts ensue until all seats are filled for every constituency. In the same fashion that the first count tallied the first preference on each ballot, subsequent counts will tally subsequent preferences respectively. At the end of each count, should a candidate exceed the quota, the surplus is redistributed

equally among the remaining candidates for the same constituency. (Citizens Information, 2021)

Most general elections have between six and ten counts. Counting is typically done the same day. (Stuart-Mills, 2021).

Analysis of 2020 General Election

The 2020 General Election was the most recent General Election in Ireland. The election saw a 62.9% voter turnout (expressed as a percentage of voting population (VAP) where over 2 million voters cast their vote (Houses of the Oireachtas, 2020).

An analysis of the cost breakdown of the election, reveals that the election cost a total of €14.4 million. This translates to approximately €6.54 per vote. (Department of Housing, Planning and the Local Government, 2020)

Country	Cost/Voter
Switzerland (2004, adj.)	€1.05
Spain (2004, adj.)	€5.32
Burkina Faso (2004, adj.)	€5.57
Ireland (2020)	€6.54
Australia (2004, adj.)	€12.12

Table 1 - Cost per Voter in Selected Countries

Table 1 shows how Ireland compares to other counties (all countries use paper ballots, data collected from general elections) (Fischer, 2004). The data was adjusted for inflation.

The 2020 General Election had a total of 17,986 employees. The total cost of labour with additional costs (food, training, etc.) was around €8.5 million, the greater portion of the total cost. (Department of Housing, Planning and the Local Government, 2020).

Postal and Special Voters

The design of the electoral system can present challenges to some voters that prevent them from being able to cast their vote in person. In most cases, people can request to vote via mail. In the 2020 GE, almost 20,000 votes were cast this way and 392 of these votes were spoilt (Houses of the Oireachtas, 2020). The total costs directly related to postal voting were around €38,000.

Some people may not be able to vote in person or via post, in these cases, special polling staff can assist the voter by bringing a portable ballot box to their residence and allowing them to cast their vote in private. In the 2020 GE, 15,000 voters cast their vote this way. The total associated cost was €150,300. (Department of Housing, Planning and the Local Government, 2020)

Challenges Facing the Irish Electoral System

Ireland's electoral system, albeit unique in many ways, is conducted in a similar

fashion as most systems around the world – relying primarily on in-person voting through paper ballots. After gaining an understanding of how the Irish electoral system works, I identified the following shortcomings and challenges it faces. Indeed, the majority of these challenges are shared among most countries.

Transparency and Auditability

The population's general impressions of the integrity of the Irish electoral system seem to be very positive. This is partially reflected in the high turnout rates at elections.

On further investigation, the reasons people most often cite as factors driving their trust in the election process are the large number of people involved in handling and tabulating ballots as well as the use of "tried and tested" in-person voting using paper ballots. (Stuart-Mills, 2021).

It is important that people's trust in the electoral system sources from quantitative factors which can be tested and audited to prove inarguably the election's integrity.

The question arises – what quantitative proof is there that the ballot one casts in an election is tabulated correctly?

In most implementations of the secret ballot[1], the only confirmation that the voter gets that their vote entered the system is that they put it in a ballot box. From there, the voter can no longer verify that their vote reached the count centre or was in fact tabulated (Stuart-Mills, 2021).

Although there are thousands of people involved in the process, and the interactions with ballots are thoroughly supervised, there is no way to prove with certainty that the final tally consists of only cast ballots, which were tabulated correctly (National Academies of Sciences, Engineering, and Medicine, 2018).

This makes the system vulnerable to claims and accusations that cannot be resolved with a definitive response. A great example of this is the 2020 US General Elections. The US does not have a national electoral body, it instead relies on states to organise their elections independently (National Academies of Sciences, Engineering, and Medicine, 2018). Although the US relies on electronic assistance when handling and tabulating votes[2], individual ballots cannot be verifiably traced.

This lack of quantitative proof of the correct operation of the system allowed Donald Trump and some of his supporters

[1] https://en.wikipedia.org/wiki/Secret_ballot

[2] https://en.wikipedia.org/wiki/DRE_voting_machine

to claim that the "election was stolen"[3] and to circulate baseless claims about election fraud[4]. The ordeal culminated in the storming of the Capitol[5].

These events could have been prevented should the electoral system have been able to prove with certainty that the votes cast were tabulated correctly. Such a situation could potentially develop in Ireland, a candidate may claim that the votes were replaced or disposed of and there would be no way for the authorities to disprove this claim definitively.

Precision

The counting process in Ireland is performed by trained, supervised counters in count centres throughout the country. Although the counting process is aided by specialised calculators, there can still be a small margin of error when tabulating votes (Stuart-Mills, 2021). Although this margin of error is typically insignificant, it may be enough to sway the election.

Financial Cost

Undoubtedly, the cost of organising a general election is significant. Arguably elections in Ireland are not as expensive to run as in other countries (Fischer, 2004), yet the €14.4 million cost incurred by the 2020 GE is not insignificant.

Human Resource Cost

The majority of the cost of the 2020 election was allocated towards employment of poll workers, counting staff, and related costs as outlined in the analysis of the 2020 GE above (Department of Housing, Planning and the Local Government, 2020).

Time and Disturbance

Setting up polling stations and vetting them takes considerable financial and time resources. A total of €1.2 million was spent on sourcing, preparing, and inspecting venues used as polling stations throughout the country. A total of 680 people were employed to assess these venues (Department of Housing, Planning and the Local Government, 2020).

The renting of these venues can also cause disturbances. Schools are closed as a result of school buildings being used as polling stations and traffic congestion may be amplified around the polling station.

[3] https://www.nytimes.com/2020/11/16/technology/trump-has-amplified-voting-falsehoods-in-over-300-tweets-since-election-night.html

[4] https://eu.usatoday.com/story/news/factcheck/2021/08/10/fact-check-8-million-excess-biden-votes-werent-counted-2020/5512962001/

[5] https://www.britannica.com/event/United-States-Capitol-attack-of-2021

Catering towards Special Voters

The electoral system, as it is implemented at present, needs to be adjusted significantly to cater towards special voters. This adjustment brings with it a significant financial and time cost to the election process.

COVID-19 Voter Safety

Should the next general election take place during the coronavirus pandemic, voters will be needed to cast their votes in communal areas among many people.

Some voters may be uncomfortable to vote under these circumstances which may lead to a reduction in turnout. The costs associated with ensuring the safety of the polling stations can be expected to be significant.

Convenience

In the 2016 GE, 30% of the surveyed population that did not vote stated a lack of time or difficulty getting to a polling station as the reason (Houses of the Oireachtas, 2016).

Not everyone has an hour on polling day to cast their vote.

Non-Intuitive Design

Voters do not receive feedback on the marking of their ballots. A voter is never notified if their ballot was marked correctly or was spoilt.

In the 2020 GE, there were 17,703 ballots spoilt. This represents 0.8% of the total poll. Although a small percentage, these votes could have altered the results of the election (Houses of the Oireachtas, 2020).

Electoral Innovation and Internet Voting

Throughout history, we have developed technologies to make interacting with the world around us more efficient and intuitive.

The past century was marked by countless innovations in computing which thoroughly changed the way we conduct business, socialise, and interact with our governments.

Elections are no exception to the changes that technological breakthroughs bring to aspects of our society.

Electronic voting has been adopted by over thirty countries in the world[6]. Electronic voting is an overarching term used to reference the use of technology to assist the casting and tabulating of votes.

Countries implement systems at various points along the election, for instance, over 20 countries are using Electronic Voting Machines (EVMs) to assist in the casting of ballots (NDI, 2013). Every country

[6] https://en.wikipedia.org/wiki/Electronic_voting_by_country

has a unique approach to electronic voting, some using it solely for marking ballots while others implement systems that allow for the recording and transmitting of ballots over the internet to be counted digitally[7].

Many states in the US have adopted the use of Direct Recording Electronic Voting Machines (DREs). These devices allow for voters to select their candidates digitally using a device at a polling station which then tallies the results locally and allows for the tally to be downloaded, printed, and/or transmitted to a central location[8]. For the US, this system offers many advantages, namely the significant decrease in time taken to tabulate votes, the reduction in expenses, provision of feedback to voters to alert them if they marked their ballot incorrectly, and ease of catering to special voters.

This implementation is far from perfect though. The system is not fully auditable, and there were numerous successful attacks identified in the 2016 Presidential Election where third parties gained access to the voting database and presumingly had the ability to alter votes (Office of the Director of US National Intelligence, 2017). Nevertheless, it represents a step forward in integrating technologies into the electoral process.

Internet Voting

Internet voting is a subset of electronic voting that encapsulates the casting of ballots by voters remotely using personal computing devices and the internet.

In abstract, internet voting allows voters to use their phones and computers to cast a vote in an election from the comfort of their home. Many countries have piloted internet voting schemes including Estonia, the Netherlands, the UK, and US (Wikipedia, 2022).

Objectives of Internet Voting

Internet voting is conceptualised as a means to make voting easier and more accessible to the population, but also as a way to increase the security and precision of elections. Internet voting can be offered as an alternative to in-person voting and be integrated into the existing electoral system, offering voters that wish to cast their vote digitally the option to do so.

Benefits of Internet Voting

Implementing well-designed internet voting solutions can bring many benefits to the election process.

Transparency and Auditability

Instead of attempting to provide integrity and ensure the secret ballot through

[7] https://en.wikipedia.org/wiki/Electronic_voting

[8] https://en.wikipedia.org/wiki/DRE_voting_machine

obscuring the ballots' paper trail as do many non-digital electoral systems, including Ireland's and America's, internet voting achieves this through precise, cryptographic transparency.

Well-designed internet voting systems use modern cryptographic techniques to allow for a paper trail to be created for each ballot cast in the system.

Ballot secrecy is an integral concept in fair elections. Voters' anonymities must be maintained to ensure that their vote cannot be coerced (Burson v. freeman, 1992). Well-designed i-voting systems allow for voters to individually verify that their ballot was recorded as intended and successfully included in the poll without compromising their anonymity (Park, Specter, Narula, & Rivest, 2021).

Such systems must also allow for any third party to independently verify the tabulation process to ensure that all ballots are counted correctly (Burson v. freeman, 1992).

If the entire process is made public and auditable by independent sources, claims about the malfunctioning of the system can be handled through the presentation of objective data.

Ultimately, if implemented correctly, these principles can ensure greater security for internet voting applications when compared with non-verifiable paper ballots, and an undeniable, quantitative proof of correctness of the election.

Precision

A study conducted in the US revealed that around one million more valid ballots were counted in the 2004 Presidential Elections that were missed by the lack of electronic tabulation technologies in the 2000 PE (Friel, 2005).

The tabulation process in internet voting applications is fully digital in most cases. This means that provided the application has no malicious intent and the code is fit for purpose, the tabulation process should have no errors.

If tabulation errors were to occur, these errors would be identified by independent auditors and the counting process is simply restarted.

Financial Cost

The financial cost associated with running a digital election is a fraction of the cost of running an in-person one.

Voting System	Average Cost
County Centres	€5.28
Polling Centres	€2.92
Internet Voting	€2.22

Table 2 - The average cost per ballot cast using different systems in Estonian 2017 Local Elections

Table 2 compares the cost per ballot cast in person in townhouses and polling centres with internet voting in the 2017 Local Elections in Estonia. There is a two-

fold difference between the cost of casting a ballot online and in person in county centres (Krimmer, Duenas-CId, & Krivonosova, 2020).

When implementing internet voting, a country should expect a significant upfront cost to develop the procedures and purchase the equipment needed (Batt, 2019). They will begin seeing a return on the investment as soon as the following election (Solvak & Vassil, 2016).

Human Resource Cost

The vast portion of costs associated with the running of non-digital, in-person elections are related to the employment of people for the preparation of the election, its running, and the tabulation of votes. (Department of Housing, Planning and the Local Government, 2020)

The vast majority of these costs would not be incurred by an internet voting solution as the casting and tabulation processes are automated. A small number of employees would be needed to oversee the system and ensure its operation.

Internet Voting in the 2011 parliamentary elections saved 1500 man-days for polling stations in Estonia (Tsahkna, 2013).

Time and Disturbance

Should a significant portion of the population choose to cast their vote online, a smaller number of polling stations would be needed. This would in turn cut down on costs and reduce the disturbance caused by the running of elections.

Catering towards Special Voters

Voters that are not able to cast their vote because they cannot get to the polling station can cast their vote online, from anywhere, using an internet voting application.

Computers and mobile phones can be equipped with assistive technologies that allow the entire population to interact with them. Such assistive technologies can aid people with casting their votes. The system does not require any alterations to allow these voters to cast their votes.

Convenience

Inarguably, it is significantly more convenient to cast a vote in seconds using your phone than it is to drive to your nearby polling station. Although added time convenience is unlikely to mobilise more voters and increase the turnout (Karp, Banducci, & Susan, 2000), studies show that voters that vote using internet voting are more likely to become recurring voters as a result (Solvak & Vassil, 2016).

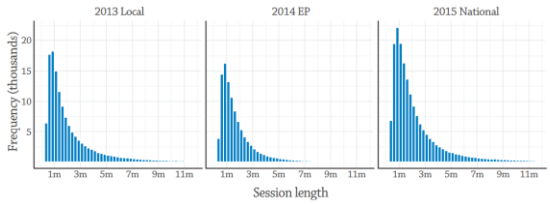


Figure 1 - Session length of casting a vote using the Estonian Voting Portal

Figure 1 illustrates the average time it took voters to cast their ballots in three elections in Estonia using the country's internet voting solution. The average time was 2 minutes and 36 seconds (Solvak & Vassil, 2016). In comparison, it takes a voter in Estonia 44 minutes on average to travel to and from a polling station and cast a vote in person (Anwar, 2020).

Voter Feedback and Intuitive Design

Internet voting applications can provide enhanced feedback to voters regarding the validity of their ballot. An application can only allow a voter to cast their ballot if it is correctly filled in. This prevents any spoilt ballots from being introduced in the system (Wikipedia, 2022).

Such implementations can also provide voters with a confirmation that the vote they cast was recorded correctly and remained unaltered throughout the system.

Requirements of Internet Voting

Internet voting is revered as the next frontier in voting – allowing for ballots to be cast in seconds and automatically tabulated in record time.

But casting ballots remotely creates a plethora of challenges, most of which are related to assuring the security of the system and integrity of an election held via the application.

Clear Threat Models and Open Source

Internet voting applications need to be thoroughly tested to ensure they uphold their security claims.

It is recommended that the providers of such applications open source their code and create threat models that assess the impacts a compromise at any point in the system would have on the system as a whole and on the election being run on the system (Halderman, 2020).

Internet voting applications should be developed considering all layers of the application to be hostile (infected with malware). A well-designed application is one where should all elements of the application be compromised, it would be known that the application is compromised and action can be taken to

mitigate the compromise (Halderman, 2020).

Verifiability and Auditability

The concept of end-to-end verifiable election systems (e2e for short) was developed to provide a set of guidelines that electoral systems should follow to provide definitive proof of the integrity, security, and ballot secrecy aspects of the implementation.

Such a system was first implemented in 2009 in Maryland. It allowed voters to cast their physical ballots in-person, and then receive a verification code that they could use to verify that their vote was recorded as intended. The digital tabulation process was also publicised allowing for independent auditing of the election (Carback, et al., 2010).

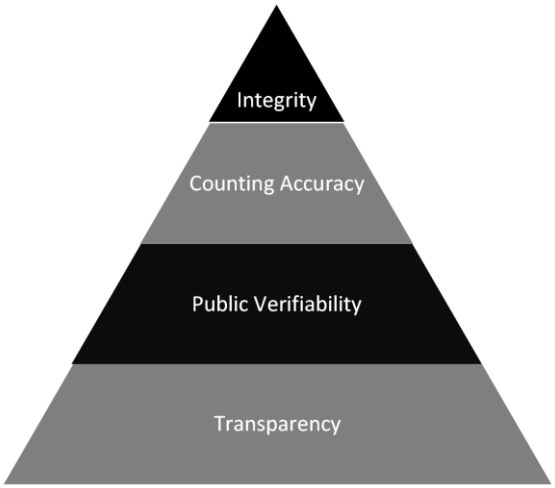


Figure 2 - Security Properties of e2e Systems

E2E implementations are not limited to internet voting as shown in the example above. All E2E systems share four characteristics:

- **Integrity:** once the voter successfully enters their ballot into the system, it cannot be undetectably altered or lost even if the system is to be compromised.
- **Counting Accuracy:** ballots cannot be miscounted without the detection of the miscount.
- **Public verifiability:** E2E systems publish sufficient verification data to permit any voter to verify that their ballot was not lost or modified and that votes were properly tabulated.
- **Transparency:** Mathematical principles underlying the application's security are open and public. The specifications for verification programs are publicly documented, and voters and observers are free to create and execute their own verification programs.

It is very difficult to implement these principles in practice and very few applications are e2e verifiable.

Trust with Voters

Because of the nature of internet voting, it may feel as though control over the process is removed from both the voter and electoral body. Pressing a button to cast a ballot may not provide the same assurance to a voter that their vote was

cast that placing their ballot in a physical ballot box does.

As a result, the application not only needs to be secure, but also give the impression that it is secure to the parties that interact with it (Solvak & Vassil, 2016). Voters should be able to confirm that their vote was cast successfully, and everyone needs to be undoubtfully assured that the ballots were tabulated correctly (Heiberg & J., 2014).

It is very important that the user interface that the application exposes to the end user is simple and easy to understand. The internet voting experience should be similar in process to the physical casting of ballots.

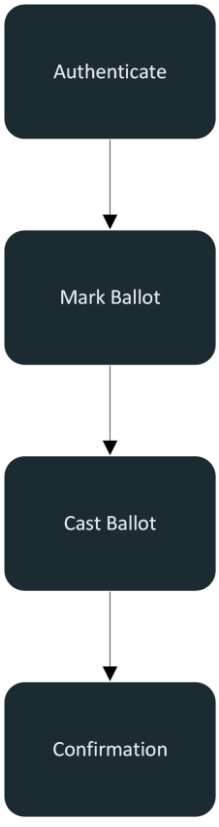


Figure 3 - Ballot Casting Workflow

Figure 3 illustrates this process in abstract. When casting a vote at a polling station, a voter first identifies themselves. This is done through the login process in the application.

After the voter successfully authenticated, they mark their ballot with their preference, they then cast the ballot. There finally needs to be a confirmation step where the voter receives confirmation of the casting of the ballot. This is achieved by placing the ballot in the ballot box in in-person voting and by receiving a success message in the application and having the ability to verify the vote thereafter in the case of internet voting.

The registration process that the voter undergoes to use the app should also be simple and quick. Ideally, as much of the process as possible would be done online. A cumbersome registration process may discourage the voter from registering to vote via the application.

Trust with Institutions

Ultimately, it is up to governmental institutions to adopt internet voting, hence the system must provide enough benefits, and certainty for governments to adopt it.

Ideally, the application would be developed in conjunction with the electoral management body and other stakeholders throughout governmental institutions. This ensures the integration of

the internet voting solution with existing election services.

It is instrumental that the full code of the application be shared with the election management body which is to review it integrally, and organise mock elections to prove its functionality (Solvak & Vassil, 2016).

Casting Multiple Votes

Many internet voting applications allow voters to cast multiple votes, where the last valid vote gets tabulated. The ability to cast multiple votes acts as a safeguard should the voter's vote be recorded incorrectly by the application or should the voter be coerced into casting a vote.

The ability to cast multiple votes also acts as a discouraging factor for bad actors that aim to manipulate the election. Should voters be able to view that their vote was recorded incorrectly and cast another ballot as a replacement, the attempt to interfere with the election would be identified and suppressed, provided of course the voter validates their vote (Solvak & Vassil, 2016).

It could be argued that allowing voters to re-cast votes allows them to change their minds and select other candidates. A study into the internet voting implementation in Estonia that allows voters to re-cast votes showed that 98% of voters cast a single vote using the system (Solvak & Vassil, 2016), hence this argument is disproven.

System Security

Elections are very high-stake targets for malicious actors. Organisations have tried to tamper with the conduct of elections since they were first held. In a sense, securing the digital casting of ballots is the opposite to securing the physical process involving paper ballots.

The advantage of physical ballots is that the ballots are distributed throughout multiple locations, making targeting enough ballots to alter the result difficult. Digital ballots are centralised in one location in most cases, this means that a successful attack on the one datastore could potentially compromise the entire election (Green, 2019).

Recent developments in cryptography have paved the way for the development of systems that allow for full transparency and third-party auditing. By making the entire system publicly verifiable, anybody can individually verify that the election was carried out correctly while maintaining a secret ballot (Park, Specter, Narula, & Rivest, 2021).

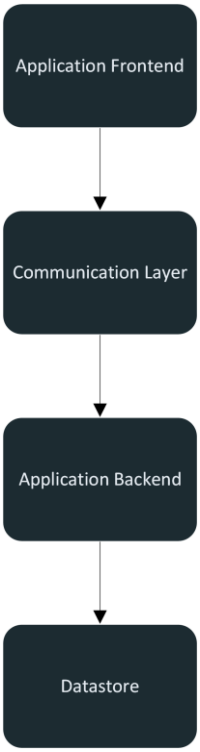


Figure 4 - Elements of an Internet Voting System

But it is not that simple. Figure 2 showcases an abstract of the different layers in a typical internet voting application.

Each of these layers can be attacked by a malicious actor. Cryptography comes in handy when securing the backend and datastore layers of the application, but the greatest problem with internet voting is the lack of security in the application's frontend layer (Park, Specter, Narula, & Rivest, 2021).

Voters use insecure hardware to cast their vote – the phones and computers used to cast the vote can be infected with various types of malwares that can either expose the voter's anonymity or alter their vote without their knowledge (Mugica, 2015).

Hostile communications infrastructure can also intercept votes between the frontend and backend, this compromises voter anonymity and may also allow for the mutation of the votes cast (Halderman, 2020).

Although the ability to verify that one's vote was recorded successfully, and to cast another vote to replace the old one should one wish to do so may offer a heuristic resolution to the latter challenge (Solvak & Vassil, 2016), the voter's choices could still be undetectably monitored by a third party, hence compromising their anonymity. No internet voting application in existence at present provides a solution for this problem.

The cost of developing such malware and releasing it would cost around $6,000,000 (Greenberg, 2012). This may seem like a steep price but the stakes are great and it can be assumed that malicious actors, such as foreign governments, have nearly unlimited resources available (Office of the Director of US National Intelligence, 2017).

Multiple reports, including a report commissioned by the US Government conclude that *internet voting should not be used in the future until and unless very robust guarantees of security and verifiability are developed and in place, as no known technology guarantees the secrecy, security, and verifiability of a*

marked ballot transmitted over the Internet (National Academies of Sciences, Engineering, and Medicine, 2018).

Implementations of Internet Voting

Internet voting is widely used in enterprise to allow shareholders to cast their votes remotely (Wikipedia, 2022). However, many countries conducted internet voting pilots to assess the feasibility of using such systems for official elections and some countries allow voters to cast their votes online today.

I will focus on assessing the implementation of internet voting in Estonia as well as two internet voting projects: Voatz and Helios and contrast them.

Internet Voting in Estonia

Internet voting was first introduced in Estonia in 2005. Estonia is famous for its policies regarding internet and computer technologies. Internet access is a human right in Estonia and computer literacy levels are the highest in the world (Borg Psaila, 2011). All Estonians have a digital ID card that links them to governmental institutions using an infrastructure called the x-road[9].

Naturally, Estonia was one of the first countries to test out internet voting. Internet voting was offered as an alternative to in-person voting for the entire Estonian population in 8 elections held between 2005 and 2016 (Solvak & Vassil, 2016).

Estonia uses party-list proportional representation in their elections. Parties propose candidates to run for elections and voters can vote for either a party or candidates representing a party. The seats are distributed to parties in respect to the total number of votes that candidates received in said party.

Voting Process

In order to vote online, voters are required to insert their digital ID card into a smart reader connected to an internet equipped computer (these card readers are broadly available to purchase). Next, they need to download a voting app which is a standalone program. Using their ID-card and a four-digit private pin, the user has to first identify themself to the system, after which the system checks whether the voter is eligible according to age and citizenship to vote in the election. If affirmative, the e-voting system displays the list of candidates in the voter's district.

Voters can then browse the list of candidates and decide for whom to vote

[9] https://e-estonia.com/solutions/interoperability-services/x-road/

for. In order to cast an e-vote, the voter has to choose a candidate and provide a separate five-digit pin to vote.

Provided the PINs inputted were correct, the downloaded e-voting app encrypts the vote using the 4-digit pin provided. After this the voter gives a digital signature to confirm their choice using their 6-digit pin. By digitally signing the vote, the voter's personal data is added to the encrypted vote. Before the ascertaining of voting results during the evening of the Election Day, the encrypted votes and the digital signatures are separated. This system is similar to two envelope postal voting in that regard[10].

To ensure that the voter is expressing their true will, they are allowed to change their electronic vote by voting repeatedly (electronically) or by voting at the polling station during the electoral period. (Solvak & Vassil, 2016)

After the voting process is complete, a QR-code is displayed in the voting application and using a smartphone with a QR-code reader a vote verification app allows the voter to verify their vote.

Adoption of Internet Voting

The share of e-voters in the first e-enabled elections in Estonia represented less than 2% of the total votes cast.

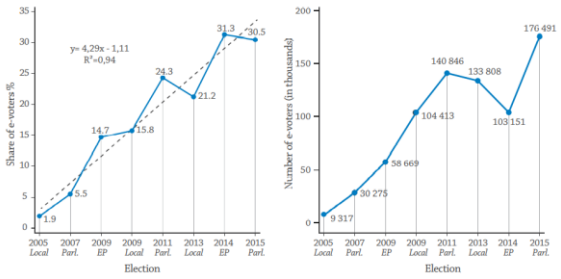


Figure 5 - Percentage and count of Votes Cast using the internet voting application in Estonia over time

As seen in figure 5, internet voting progressively increased in popularity over time. Internet voting seems to follow a typical adoption trend for technological products;

Early adopters, usually in younger demographics, start using the product first. Over time, as the technology matures, more people are willing to use it.

Once a distinct subgroup of the population has reached the adoption stage, subsequent spread to other groups, referred to as diffusion starts happening, where the number of people adopting the technology is dependent on the number of previous adopters (Rogers, 1962).

In Estonia's case, for the first three elections, multiple sociodemographic, attitudinal, and behavioural factors had a non-trivial association with being a first-time voter. However, from the fourth election onward, the importance of these factors gradually diminished, indicating the diffusion of e-voting among the Estonian electorate (Solvak & Vassil, 2016)

[10] https://en.wikipedia.org/wiki/Postal_voting

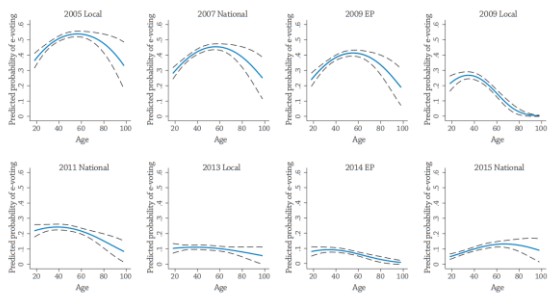


Figure 6 - Effect of age on the probabolity of e-voting for the first time over elections

As seen in Figure 6, there is a loss of relationship between age and the probability of casting a vote online. This is shown by the flattening of the curve over time.

Technological diffusion has traversed social boundaries and Estonian e-voters nowadays have become virtually indistinguishable from regular paper ballot voters. The evidence suggests that successful diffusion of e-voting has taken place. Over 30% of the population cast their vote online in the 2015 parliamentary elections. (Solvak & Vassil, 2016)

Voter Turnout and Internet Voting

Many studies have attempted to see if there is a positive relationship between the introduction of internet voting and voter turnout.

An Estonian paper refers to this investigation as the "bottleneck effect". Most people that do not vote choose not to do so for a multitude of reasons. These could include alienation from the political landscape or as a sign of protest. This demographic does not care about voting, hence why should a change in the way voting is conducted change their attitudes towards the idea of voting? (Solvak & Vassil, 2016)

The purpose of internet voting is not to mobilise people that would generally not vote, but to improve the electoral system and make voting more convenient for the portion of the demographic that chooses to vote.

Security Audit

The Estonian system underwent an independent security audit in 2014. The system was known not to be end-to-end verifiable and the audit succeeded in finding vulnerabilities in the system. The report concluded that the simplicity of the system's design, resulted in the system blindly trusting the integrity of the voter's hardware, system software, and staff.

Conclusion

In conclusion, Estonia's implementation of internet voting was a phased process. General adoption of the system took multiple elections to achieve but eventually, the relationship between certain demographics and the likelihood to cast a vote online diminished.

Estonia's i-voting system is not perfect, but the government is committed to incrementally improving it over time.

Voatz and Helios

Voatz[11] and Helios[12] are both internet voting application projects. They differ primarily in their business model – Voatz being a for-profit company while Helios is a non-profitable project, and their approach to solving the internet voting challenge.

Voatz Abstract

Voatz is an American internet voting application that allows voters to cast their votes remotely using a mobile application. The company was founded in 2015, raised $10.6 million through 7 rounds of funding, and is valued between $10-50 million. (Crunchbase, 2021)

The application was first used in an election in West Virginia in 2020 and was later used in a referendum in Venezuela. (Seletsky, 2020)

The company received a lot of contestations due to their reluctance to reveal the system's design and source code. Eventually, they released a short document outlining the abstract of the application while providing no information on the operation of the system (Moore, 2019).

This led MIT researchers to reverse engineer the app in an attempt to test its integrity (Halderman, 2020).

Unsurprisingly, the report found countless security flaws in the system and discovered that it used a very basic, non-verifiable process that provided no guarantee to the integrity of the election.

Helios Abstract

Helios can be regarded as the opposite to Voatz. It is a research project started by Ben Adida at Harvard University that aims to create an internet voting system that complies with end-to-end verifiability standards.

Helios has open sourced all code, which is available on GitHub[13], and hosts a running version of the application that can be used to organise small elections.

The system has been audited and independently confirmed to be end-to-end verifiable. It relies on homomorphic encryption and elliptic curve cryptography to achieve this. Helios is one of the few internet voting applications that achieve end-to-end verifiability. The application allows for individual voters to assert that their ballot was recorded successfully and allows the tabulation process to be verified.

But Helios fails, like all other internet voting applications, to achieve security in the frontend. The compromise of the voter's hardware can result in the compromise of

[11] https://voatz.com/
[12] https://vote.heliosvoting.org/
[13] https://github.com/benadida/helios-server

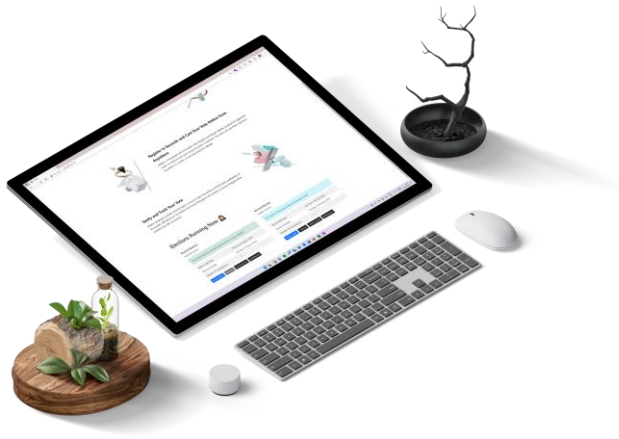
voter anonymity and/or mutation of the ballot.

Conclusion

In conclusion, Voatz and Helios differ significantly. Voatz approached internet voting conservatively, refusing to release any associated code or proof to uphold their claims about the system's integrity whereas Helios has an open-source approach allowing independent auditors to validate the code, and test the application. It is evident that an internet voting application needs to be as open as possible in order to build trust in the system (Appel, DeMillo, & Stark, 2019).

ethlect. Abstract

After researching electoral systems, how internet voting has the potential to revolutionise the way we cast votes, and how existing internet voting applications work, I designed ethlect.



ethlect. is the first end-to-end verifiable, internet voting application for elections run using the PR-STV model.

ethlect. employs a hybrid approach to internet voting that allows it to be the first internet voting implementation that can assure the security of the application's frontend.

A physical layer is added to the application, similar to postal voting, where voters introduce their votes into the application using paper ballots received. The voter never introduces their direct candidate selection but instead ciphertexts representing their candidate choices. This allows for an unprecedented level of security and anonymity where nor the application, nor any malicious actors can identify the voter's selections[14].

The application allows for voters to verify that their vote was cast successfully, and all ballots are traceable throughout the application once they are cast. This allows for third parties to audit the system and ensure the correct tabulation of votes.

ethlect. is designed specifically for Irish General Elections and integrates with existing public systems to validate voter identities and ensure the system's integrity.

[14] See security section of paper for details.

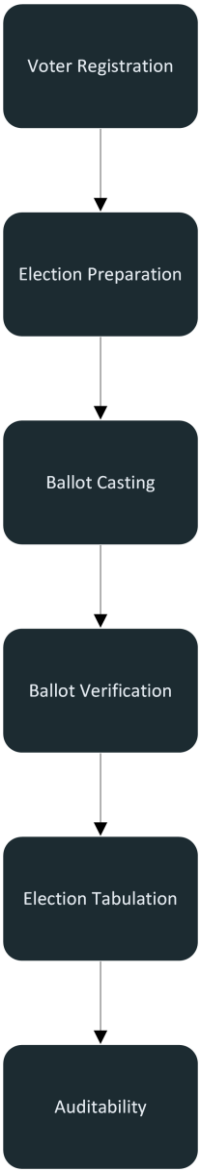


Figure 7 - ethlect. Abstract System Architecture

Figure 7 illustrates the system's architecture in abstract[15]. Voters that are already registered to vote can follow the online registration process to use the app to cast their ballot.

After the voter registration period, the election is prepared by the election authorities. In this process, an election is started using the application by inputting the candidates running for each constituency and distributing keys. Ballots are generated for all voters in a verifiable manner, the ballots are then printed at a secure location (again in a verifiable manner) and posted to the voters.

The electoral period is then started. Voters can use their physical ballots to introduce their candidate selection in preferential order. This is done by introducing the ciphertexts representing each candidate. The application will ensure that the ballot was marked correctly and will then add the ballot to the virtual ballot box.

The voter can verify that their ballot was cast correctly either through the application or by querying the public database themselves. This process is designed such that only the voter can confirm the correct casting of the ballot in a way that prevents third parties from identifying the voter's choices.

After the completion of the electoral period, the election administration body can start the tabulation process. This process involves the re-encryption and cryptographic shuffle of ballots in a verifiable fashion as to lose the correlation between a ballot and the voter that cast it.

Ultimately, after the ballots were shuffled a few times, ten keyholders that hold decryption keys are to combine their keys and decrypt the ballots verifiably.

[15] See Application Workflows section for more details

All stages of the tabulation process are made public together with cryptographic proofs that attest the correctness of each shuffle and decryption of the ballot set. This allows third parties to assert the integrity of the system.

Summary

In conclusion, ethlect. circumvents the impossibility of assuring security in the frontend of the application by adding a physical layer, similar to postal voting.

Voters never introduce their direct candidate selections into the system but a ciphertext representation which cannot be correlated to their candidate choice by any malicious actor, nor the application (without the input of the keys).

ethlect. is designed for the Irish electoral system in mind and is truly end-to-end verifiable.

Objectives of ethlect.

ethlect. aims to accomplish seven key concepts:

End-To-End Verifiability

The system is truly end-to-end verifiable. It complies with the criteria of an e2e verifiable system[16].

Figure 8 illustrates the process that a ballot takes from its creation to its decryption. All stages are verifiable to ensure that the ballot was cast and tabulated correctly. ethlect. encrypts ballots before they are cast, unlike most other applications. This process takes place when the election is prepared. Two ballots are generated per registered voter for their constituency. Each ballot is encrypted, and the encrypted candidates are shuffled before assigning the ballot to a specific user. The application generates a zero-knowledge, non-interactive proof of encryption that can be verified by third parties.

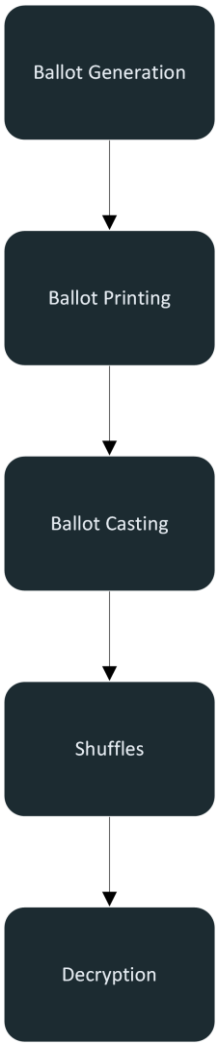


Figure 8 - Ballot Handling Layers in ethlect.

[16] Outlined in Requirements of Internet Voting Section

The ballot printing process involves the printing of the ballots generated. This process involves the decryption of the ballots by an airtight computer at a safe location in order to reveal which candidate each ciphertext on a ballot represents. The ballots are printed, the sensitive information is covered in tamper-proof scratch off ink[17], and they are posted to the voters. This process is verifiable through the introduction of placebo votes in the system by auditors during the printing process (this is discussed later in the paper).

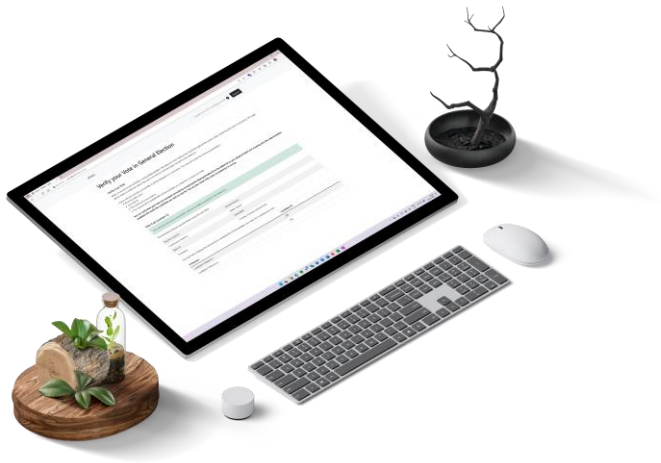
Upon receipt of the ballot, the voter reveals the ciphertexts representing each candidate and identifies the candidates they wish to vote for. The voter logs into the application and inputs the ID assigned to their ballot and their selection of candidates in preferential order using the ciphertexts.

Should the vote be cast successfully, the application will allow the voter to verify that their ballot was recorded successfully in the virtual ballot box. This datastore is read-only and data entered cannot be modified.

From this stage onwards, the application no longer tracks individual ballots through the system but instead ensures that from one stage to another, the outputted ballot set contains the same ballots as the inputted, which indirectly verifies the integrity of individual ballots.

All cryptographic shuffles produce an indirect zero-knowledge, non-interactive proof similar to the generation process that can be verified by third party auditors. A somewhat similar proof is provided by the decryption process.

All the data needed for an auditor to trace ballots throughout the system without the compromise of voter anonymity is made public on the website.



Frontend Security

As mentioned before, the application circumvents the inevitable lack of security in the hardware voters use to cast their votes by allowing voters to input ciphertexts representing candidates into their devices when casting a vote.

The vast majority of credible studies into internet voting agree that internet voting solutions should not be implemented

[17] https://en.wikipedia.org/wiki/Scratchcard

because of their inability to ensure security in voter hardware (National Academies of Sciences, Engineering, and Medicine, 2018).

In existing internet voting applications, the voters input their candidate selection into the application directly. In this sense, there is a correlation created between the voter and the cast vote before it is encrypted. This correlation occurs in the application's frontend. Hence should a bad actor have access to the voter's device, they can compromise the voter's anonymity and/or change the vote selection without their knowledge. As a result, it is important to note that although some internet voting applications claim to be end-to-end verifiable, these claims are related to the application's backend only as it is impossible to definitively secure the frontend (Mugica, 2015).

ethlect. encrypts all ballots before they are cast and assigned to voters. Instead of encrypting the vote cast by a voter after casting it, the application encrypts all candidates on the ballots when they are generated.

As a result, the only way voter anonymity can be compromised, or the vote mutated is if the attacker has access to the voter's physical ballot and their login credentials or control over their computer.

Because of the added physical requirement, it becomes very infeasible for a successful attack of a scale large enough to change the outcome of the election as this would require having access to a large number of physical ballots and digital credentials.

In a sense, the security guarantee of the frontend is similar to that of postal voting, which is implemented by over 18 countries (Wikipedia, 2021).

Integrity in Case of an Attack

ethlect. is designed such that should all data stored by the application be leaked, the worst-case scenario is the compromise of voter anonymity[18].

Because all aspects of the application are publicly verifiable, a successful attack on the application would be detectable by auditors. The source of the attack can be identified, and the application can discard all damaged ballot processes and process the ballots again.

The system does not make any assumptions about the correct operation of any of its parts, instead allowing for the independent auditing of all its components.

Ease of Registration

In order to encourage voters to cast their vote using the system, the registration process is designed to be fully digital, and

[18] See section on security analysis for more.

integrated with state-of-the-art security tools to transition a registered voter to the application in minutes.

The application is integrated with the Dublin Voter Registry[19] via a public API. When the voter registers to use the application, their details will be checked against the registry to ensure they are registered to vote.

The voter must verify their identity before casting a vote to ensure they are who they claim. Stripe's Identity[20] service has been integrated into the application for this process. The voter must provide a proof of ID (through a picture of their passport) and a video selfie (multiple photos while the face is in motion to prove that the person is real). If all security checks pass, the application will allow the voter to cast their vote.

This process is complete in a few minutes.

Integration with The Electoral System

Figure 9 illustrates the integration of ethlect. with the existing Irish electoral system. Internet voting should be provided as an alternative to in-person voting and not a replacement (Solvak & Vassil, 2016). The election held digitally using ethlect. should allow for an election window of around a week prior to Election Day in person (Solvak & Vassil, 2016), this is to ensure voter convenience and allow for the recasting of votes should a voter deem their vote to be mutated.

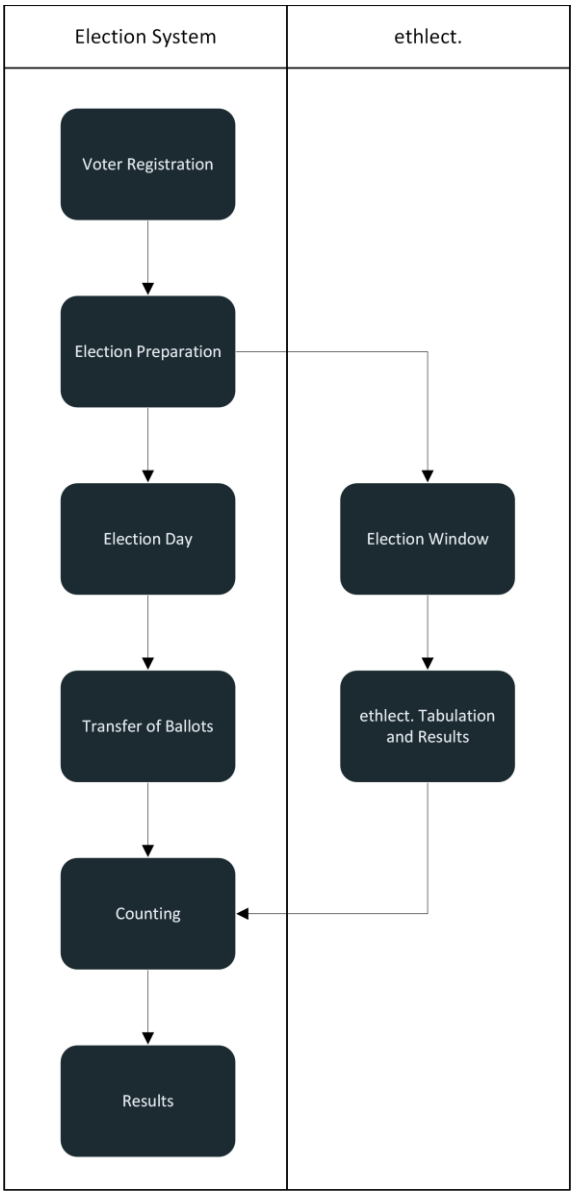


Figure 9 - Diagram of Integration of ethlect. with Electoral System

At the end of the election window (referred to as electoral period), the application will allow for all ballots cast

[19] https://www.voter.ie/

[20] https://stripe.com/ie/identity * note that the service is in private BETA in Ireland.

using the system to be decrypted. The decrypted ballots contain a list of candidates in preferential order.

This list can be embedded on a memory element (memory stick, etc.) and the ballots entered into the official count (mixed with the paper ballots).

This can be done by printing the ballots cast online and adding them to the existing ballots in the system, or simply by counting the digitally cast ballots together with the physical ones.

Alternatively of course, the votes can be counted digitally by the application. This prevents the need to mix the digital ballots with the physical ones and ensures full end-to-end verifiability.

Transparency and Ease of Access

The application aims to make access to the information needed to verify the integrity of the system easily available to anyone. Anyone can download JSON files representing the data processed at different stages of the system from the application website.

The source code of the application is also made public and easily accessible[21].

Minimal UI

The application aims to make the process of organising elections and casting votes easy and straight forward. A minimal, and well-designed user interface is essential to achieve this.

It is important for complex applications, such as internet voting ones, to reveal as little complexity as possible to the end user, as to not overwhelm them.

In this sense, the application uses plenty of user feedback, beautiful UI packs, and thought-out UX strategy to make interacting with it easy.

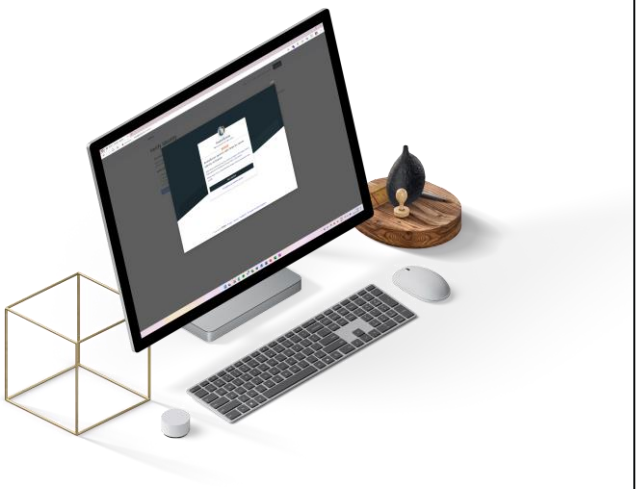
ethlect. Workflows

This section of the paper walks through the inner workings of the application explaining all workflows. The workflows will be explained conceptually, their implementations will then be shown using screenshots of the application, and the models behind them will be explained in appendices. Please reference figure 7 for an outline of the workflows.

I conceptualised, designed, and developed all aspects of the application over the past months myself.

[21] See Links Section in Abstract

Voter Registration



The voter registration process represents the first workflow in the application. Because the application generates ballots for specific voters[22], voters that wish to partake in the election must register online before the ballots are generated. The registration process can be completed in around 6 minutes (including Stripe Identity).

This process can be split into two parts: the registration process (where the voter inputs their details and the voter registry is checked), and the identity verification process (where the voter verifies their identity using Stripe Identity).

Registration Process

Figure 10 illustrates the voter registration process. From here on out, the frontend will reference the web application served to the user and the backend will reference the servers of the application.

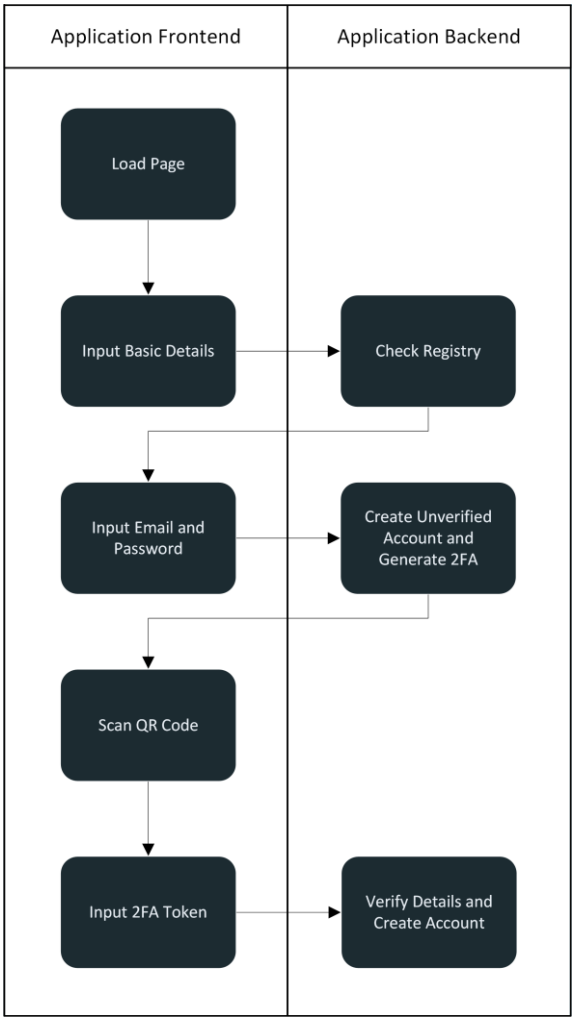


Figure 10 - Voter Registration Process Diagram

A user can register to use the application by heading to `/register`. They will be see a form asking them to input their first

[22] Security implications explained in security analysis

name, last name and Eircode, as seen in figure 11.

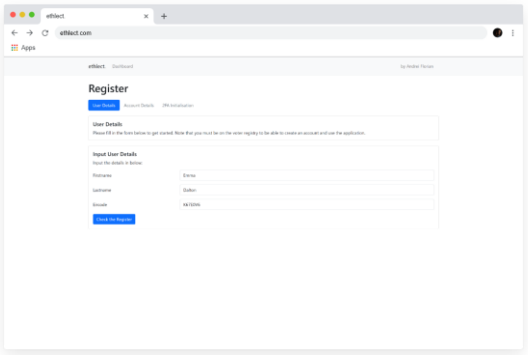


Figure 11 - Voter Registration Basic Details

Upon submitting the form, the application will submit the details to `/api/register/check`. This API will send an API request to the URL below, where the queries are filled in with the details provided by the voter.

```
https://www.voter.ie/api/search/name/${firstName}/surname/${lastName}/eircode/${eircode}/lang/en
```

This is the public API that the voter registry exposes. If the voter with the provided details is found in the registry, the call will return a package of details about the voter including their constituency. The app's API will resolve with a `{match: true}` if the voter is found.

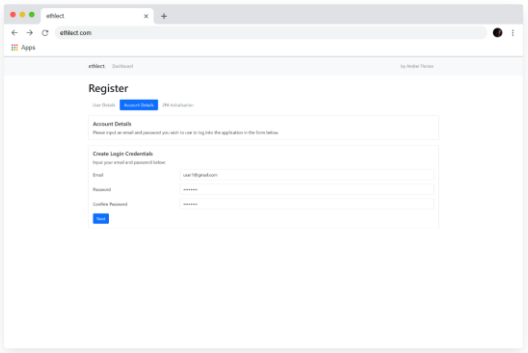


Figure 12 - User provides email and password

If the application receives a positive match, it will move to the next tab in the form.

This can be seen in figure 12. The voter is asked to input an email and password they will use to log in.

After submitting the form, the application will send these details to the API `/api/register/create`. This route will check the register again (to prevent data mutation in the frontend) and will generate a 2FA secret.

The application uses 2 factor authentication for logins. When logging in, a voter must provide their email, password, and a time sensitive token provided by their 2FA application (such as Microsoft Authenticator). See appendix 1 for details.

The backend will then salt and hash the passcode and create a user in the authentication database. This database is a MongoDB read/write DB. The application will set the `{accountVerified: false}` property on the user document.

If this process was successful, the website will progress to the last page where the voter is asked to scan a QR code using their authenticator app. This code allows the application to generate time sensitive tokens. The voter must input the time sensitive token outputted by the authenticator app to complete their registration as seen in figure 13.

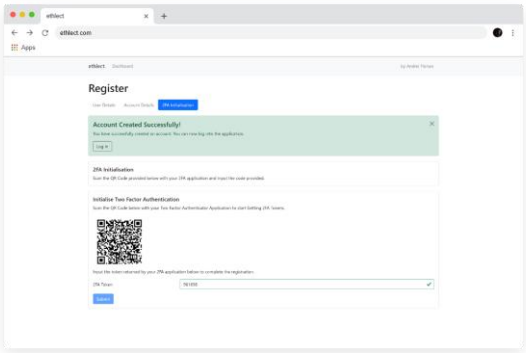


Figure 13 - User 2FA Initialisation

The voter will then be offered the option to log into the application.

Voter Login

The application uses the Next Auth[23] framework to handle user login. This is a minimal JavaScript framework that uses server side JWT tokens and integrates perfectly with Next.js (the framework used for the app).

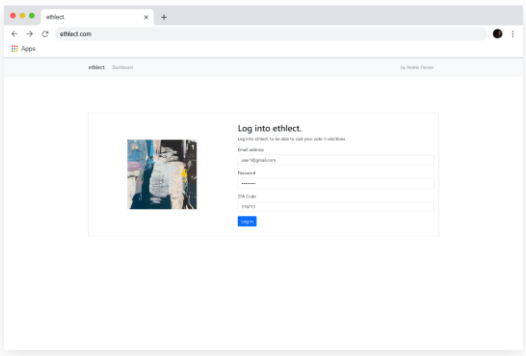


Figure 14 - Login Page

Voters can log in by accessing the `/login` page of the application. Here, they are asked to input their email, password, and a 2FA token released by their auth app.

A successful form submittal is submitted to an API route handled by Next Auth. This in turn sends the login details to a custom route `/api/auth/endpoint` that verifies the 2FA token using a JavaScript library[24] against the secret stored in the database, salts and hashes the passcode, and checks if the hash matches the one in the DB. If the verification is successful, the API will return a success status to the Next Auth route which will in turn handle the login. The voter will receive an error message should they fail to log in.

Identity Verification Process

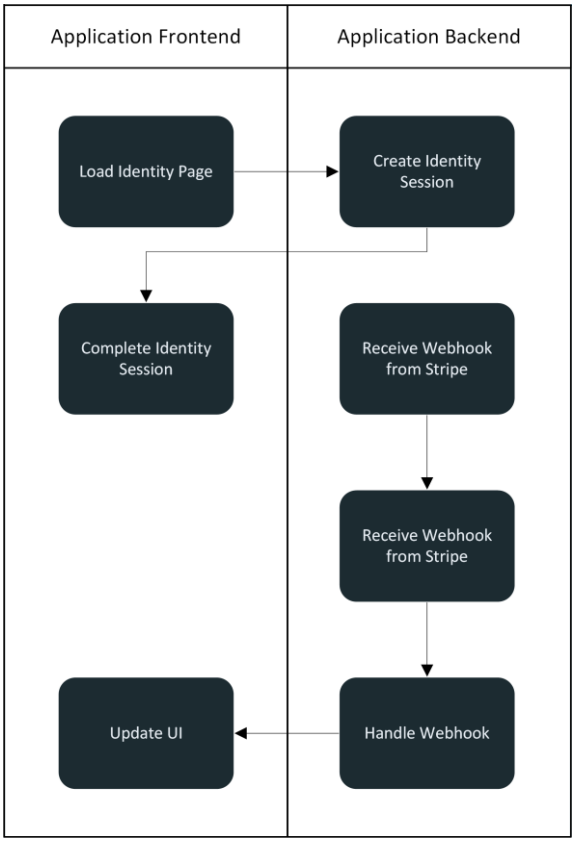


Figure 15 - Identity Verification Process Diagram

[23] https://next-auth.js.org/

[24] https://www.npmjs.com/package/speakeasy

Figure 15 shows the process for verifying the voter's identity. After logging in successfully, the voter will be alerted to verify their ID to be able to cast votes.

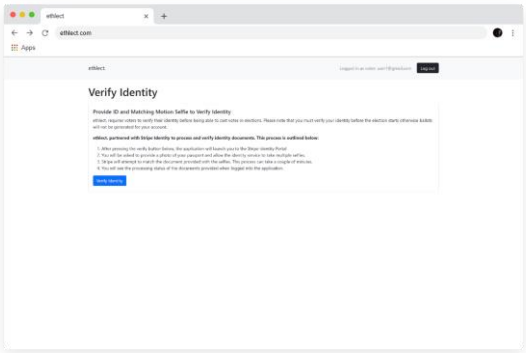


Figure 16: Identity Verification Page

By following the link in the alert, the voter goes to `/verifyID`. From this page, as seen in figure 16, the voter can press a button that loads the Stripe Identity pop-up. When the button is pressed, the frontend sends a request to `/api/register/stripeSession`. Stripe Identity works using "verification sessions". These are secure sessions that the application backend can create for users that act as containers. The user can input verification documents in this container and submit them for Stripe to check.

The API will resolve a client key that the frontend will use to initiate a launch the session.

Stripe Identity is very flexible in the sense that it allows a user to provide the documents using their phone; the user can generate a QR code and scan it with their phone to provide these documents.

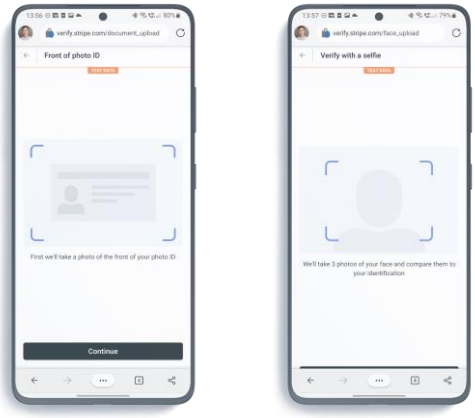


Figure 17 - Stripe Idenity Process

The voters are required to scan their passport and then allow the application to record their face for a few seconds. When complete, they can confirm the submittal of documents for verification using the web portal.

The documents are now sent over to Stripe. The verification process should take a few minutes. Stripe sends events related to user ID verification to ethlect. via a webhook at `/stripe-webhook`. Stripe sends three types of events:

1. `identity.verification_session.processing`
2. `identity.verification_session.requires_input`
3. `identity.verification_session.verified`

If the route receives the first event, it means that a verification session has been submitted successfully and the documents are being processed. Every user document in the DB has an `idVerified` attribute. This property on a user document is set to `pending` if the documents for the respective user are being processed.

If the second event is received, it means that the verification failed. In this case, the app will set `{idVerified: 'rejected'}`.

Finally, if the session returns the third type, it means that the document and selfie matched. The backend will now extract the first name and second name from the document (provided by Stripe Identity) and compare it to the name the voter signed up with. If they match, the app will set `{idVerified: 'true'}`, otherwise `{idVerified: 'rejected'}`.

The voter will receive feedback about the state of the session. If their documents are pending, a yellow warning will be displayed in the frontend informing them of this. If the session is rejected, they will be asked to try again, and finally if the documents match, a tick will be added next to their email in the navbar.

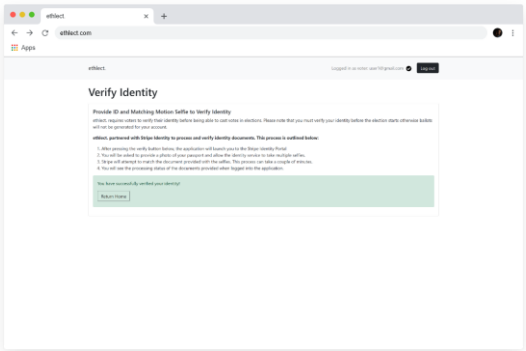


Figure 18 - Voter Verified Identity Successfully

The voter is now introduced in the verified voter pool and ballots will be generated for them.

Election Preparation

The election preparation workflow consists of four processes. Two of these processes are done using the ethlect. application and the remaining two are related to the printing and distribution of ballots.

The election is to be prepared by the electoral management body, in this case a subset of the Dept. of Housing, etc. The implementation is designed to have multiple keyholders. A keyholder is an entity entitled with a fraction of the decryption key that can be used to decrypt ballots.

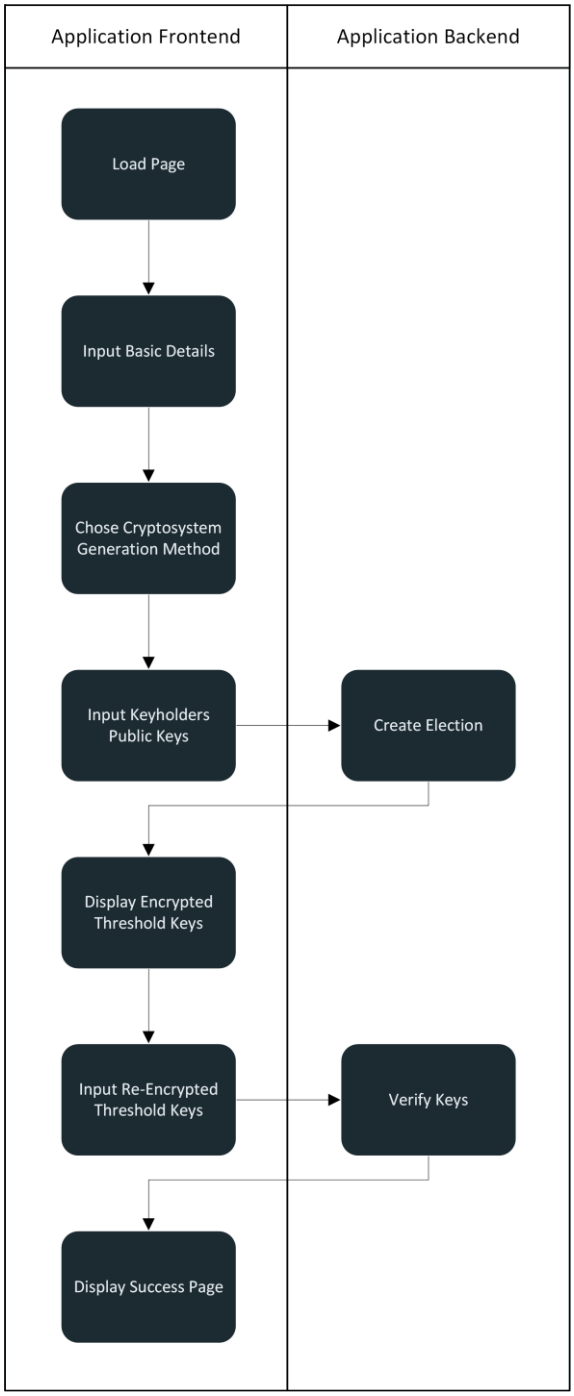
The application uses the Shamir Threshold Scheme (appendix 2) to share the key used to decrypt the ballots into ten keys. In this case, eight of these ten keys are needed to reconstruct the original key, although this threshold is changeable.

These keys should be distributed to different stakeholders involved in the process (president, Taoiseach, etc.).

Election Creation Process

The election creation process is completed digitally using the application. One or multiple admin type accounts can be added to the application's authentication database by the election management body. These accounts can create and manage elections.

A new election can be created by accessing `/admin/new` using an admin account.



The admin is asked to input a name for the election, a description and a start and end date (note that these dates are only for public information, only the admin can start and end the electoral process).

The application also requires the input of a csv file containing all the constituencies in the election being created and a list of candidates running for each constituency.

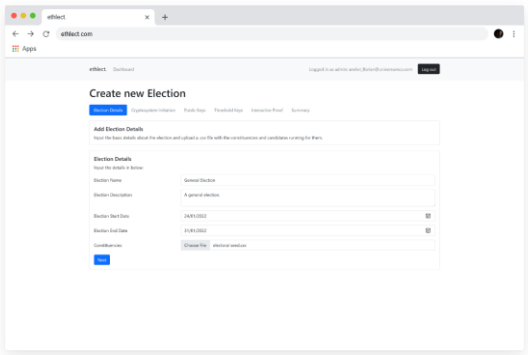


Figure 19 - New Election First Page

Consituency	Seats	Candidates
Dublin Central	2	Candidate 1, candidate 2...

Table 3 - csv file format example

Table 3 shows an example of this format. The file should be comma delimited. Upon submitting the form, the voter will be directed to the next tab, here they can select the cryptosystem initiation method.

The application uses the El Gamal Elliptic Curve Cryptosystem using multiplicative notation (appendix 3). This involves generating large primes which can take hours to compute. A library is used for testing that provides these ready-made values from a specialised server (this should be done locally for official elections). The admin can choose which method to use.

After this, the admin is asked to input 10 public RSA keys. These keys should be generated by the stakeholders who are to create the keys and sent to the admin. I developed a small app that handles the cryptography on the keyholders side, this application is included in the GitHub page.

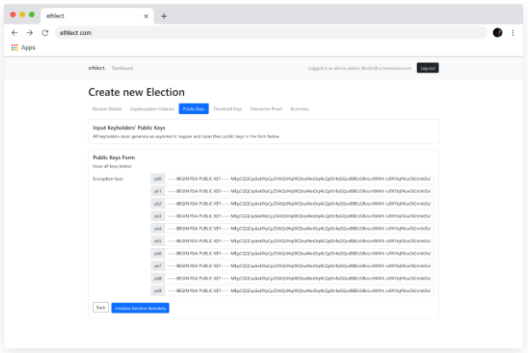


Figure 20 - Keyholders public keys (note that the same key was used here for all keyholders for simplicity)

Upon submitting this form, the frontend will send all the inputted data to `/api/election/create`. This route will process the csv file inputted and initiate the cryptosystem by generating the numbers (remotely or on the server depending on the admin's choice). The backend will then share the El Gamal private key (x) into ten shares with a threshold of eight.

Finally, the app will hash the El Gamal private key, encrypt the key shares using the public RSA keys provided by the admin, generate another RSA keypair (this will be referenced as the election private and public key E_{pk} and E_{prk} respectively), and create a new election document with the following data:

1. The El Gamal values g, p, y and $SHA256(x)$ (the hashed private key)
2. The election RSA keypair (public key E_{pk} and private key E_{prk})
3. An election ID where $ID_E = ID_{E-1} + 1$ (an increment of the ID of the last election)
4. The election details (name, description, dates)
5. Election constituencies as an array of objects. Each object has a `constituency` property and a `candidates array` where the candidates are listed.

If this process is successful, the admin will be directed to the next tab. Here they will be able to copy the encrypted threshold keys and distribute them to the respective keyholders.

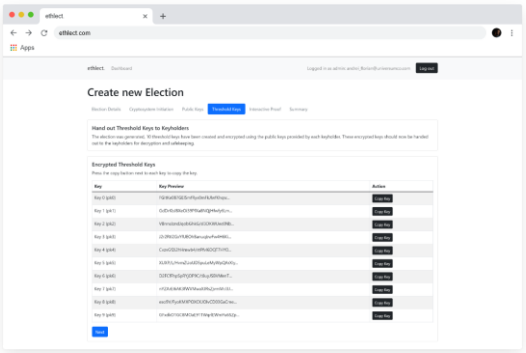


Figure 21 - Key Distribution

The keyholders should use their generated private key (linked to the public key provided) to decrypt the keys and store them securely. The admin can press next to reveal the verification tab. The frontend will send a request to

`/api/election/getElectionKey` to get the public election key (E_{pk}) which the admin can copy and distribute to the keyholders. The keyholders are to encrypt their decrypted threshold keys with E_{pk} and send the encrypted keys to the admin which should input them into the application.

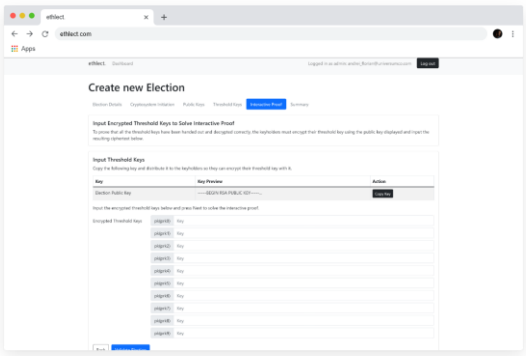


Figure 22 - Input Threshold Keys Encrypted with E_{pk}

When submitting this form, the app sends a request to `/api/election/verifyKeys`. This API will receive the encrypted keys and will decrypt them, attempt to combine them, hash the combination, and compare it against $SHA256(x)$ in the database. If the keys match, the backend will set the property `{electionVerified: true}`.

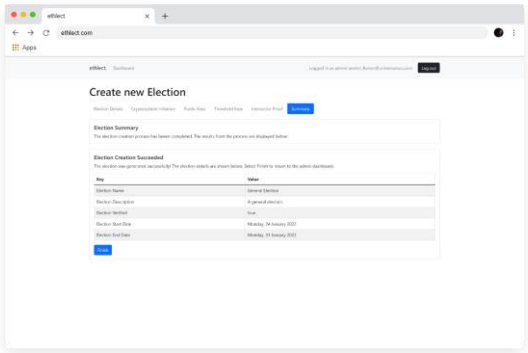


Figure 23 - Election Creation Results Table

The admin will receive a success or error message as a result. The verification process is added to ensure that the admin successfully distributed the keys to the keyholders, and they managed to decrypt them. The keys cannot be intercepted as they are always encrypted when communicated between the keyholders and application. The threshold keys are never exposed to the admin or other entities in the process. In this sense, the admin solely functions as an intermediary.

Ballot Generation

The admin can now generate ballots for the created election. This can be done by going to `/election/${electionID}/generate` where `electionID` is the ID of the election to generate ballots for.

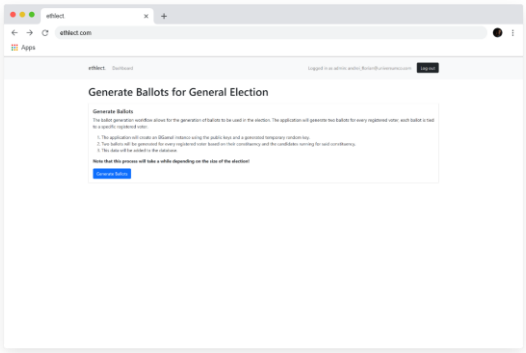


Figure 24 - Ballot Generation Page

The admin can press the button on the page to start the process. This process can take a while depending on the number of ballots that need to be generated.

Figure 25 shows an abstract of the process. The application backend will first determine the number of ballots n to generate for each constituency C_i by calculating $V * 2$ where V is the number of voters registered for constituency C_i. Note that more ballots can be generated at a later date should the need arise.

n number of candidate IDs ID_n (6-digit, random numbers) are generated for each candidate c of constituency C_i. The logic behind this is explained in appendix 4.

n number of ballots are then generated by appending the candidate ID ID_n of each candidate c representing the constituency the ballot is intended for to each ballot.

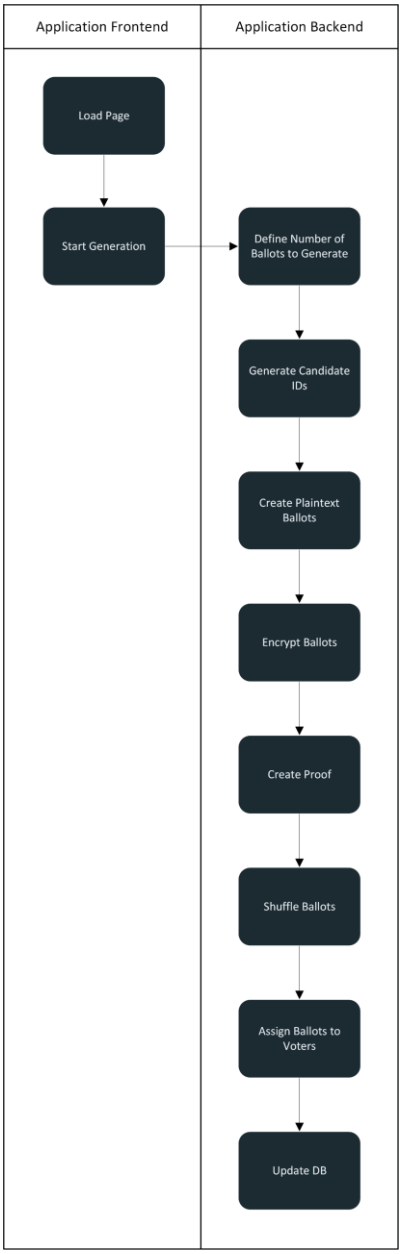


Figure 25 - Ballot Generation Process Diagram

The result should be n ballots, each containing different candidate IDs of the same candidates.

Ballot	Candidate 1	Candidate 2
1	204549	853903
2	429040	489372

Table 4 - Ballot IDs example

Table 4 shows an example of this. The numeric values representing candidate 1

in ballots 1 and 2 are different yet represent the same candidate. The same is the case for candidate 2

These ballots are then encrypted (appendix 5) by encrypting every candidate ID in every ballot n.

Proof is created to attest the correctness of the encryption and then the encrypted candidates within each ballot set are shuffled and finally the ballot set itself is shuffled.

Now the application will assign 2 ballots to every voter V_s that is registered for constituency C_i. The ballots assigned are designed for the voter's constituency. Note that at this stage, the application will no longer know which candidate ID each encrypted candidate ID relates to because the order of the candidates in each ballot was permuted randomly. This is key in achieving the extra layer of security unique to ethlect.

For voter convenience, a random 3-digit number is generated (unique to every ballot) for every encrypted candidate ID for every ballot. One cannot expect the voter to input over 50 characters representing a candidate when casting their vote, hence these pins can be inputted instead which are stored alongside the encrypted candidate ID they represent.

Finally, a random unique 9-digit number is generated as a ballot ID for every ballot.

the election document in the database is updated with the following fields:

1. The constituencies array created when creating the election is updated such that the array of candidate IDs representing each candidate that was generated during the ballot generation process is added to each candidate.
2. The generated ballots are added in the database as an array of objects. Each ballot object contains the ballot ID generated, user ID of the voter the ballot is assigned to, the constituency it is designed for and an array of objects containing each candidate's 3-digit number and related ciphertext.
3. The proof is added to the shuffles array in the database. This object contains the plaintext ballots, encrypted and permuted ballots, and an accompanying indirect zero-knowledge proof of the encryption.

If the generation process succeeded, the admin will receive a success message in the frontend.

Ballot Decryption and Printing

After the ballots were successfully generated using the application, they

must be printed and distributed to the voters.

This process needs to be done in a verifiable manner, in a safe location using airtight computers. It is critical that this process happens integrally as to not compromise voter anonymity.

1. The ballots generated by the application should be burned to a memory element and transferred to the safe printing facility.
2. An airtight computer running a decryption program (the same program used in the tabulation process), must share an RSA key with the keyholders that they can use to encrypt their threshold keys and input them into the system.
3. This computer should then decrypt and combine the inputted keys and use the resulting key to decrypt all candidate IDs of all ballots.
4. The computer should then print these ballots. Every printed ballot consists of the ballot ID and list of candidates for the respective constituency. Every candidate should be represented by their name, party, etc. together with the 3-digit number that is to be introduced into the system when selecting the candidate. It is recommended that this ID be covered with scratch off ink.

These ballots can then be sealed in envelopes and sent out to the respective voters.

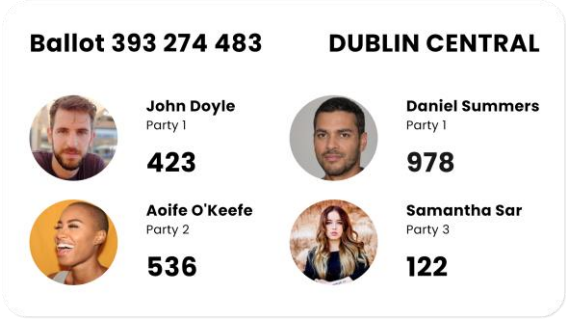


Figure 26 - Example Ballot Design (main elements included only)

This seems great, but how can one verify that the ballots were decrypted and printed correctly? In this case, auditors should be allowed to add auditable ballots to the list of ballots to be printed. These ballots are generated in a similar fashion to the real ballots: for each candidate on every ballot: a random 6-digit number (representing one of the candidates running for the constituency chosen from the generated candidate IDs stored alongside the respective candidate in the DB) is encrypted using a random key chosen by the auditor.

Random 3-digit numbers are then assigned to each encrypted candidate ID. The auditor can choose a ballot ID for the ballot and must be the only one that knows which candidate IDs each 3-digit pin relate to.

After all ballots are printed, the auditable ballots are extracted by the auditors which can then reveal the 3-digit pins

representing each candidate and check if the candidates are represented correctly.

Should enough ballots be successfully audited in this manner, it proves probabilistically that all ballots printed are printed correctly.

Ballot Casting

The ballot casting process allows voters to cast their ballots in an election. This process can only happen once the admin has started the electoral period. This can be done by accessing `/admin/${electionID}/start`. By pressing the button on the page, the app will send a request to `/api/election/startElection`. This route will set `{electoralPeriod: true}` on the election document, which will allow voters to cast their votes.

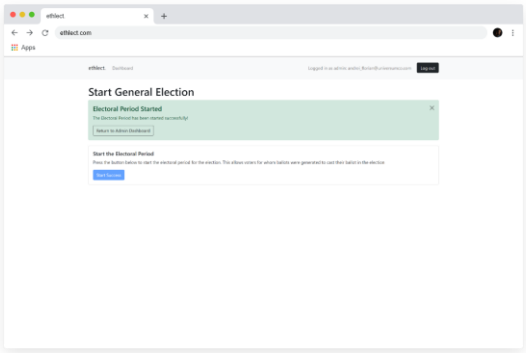


Figure 27 - Start Electoral Period

Casting a Ballot

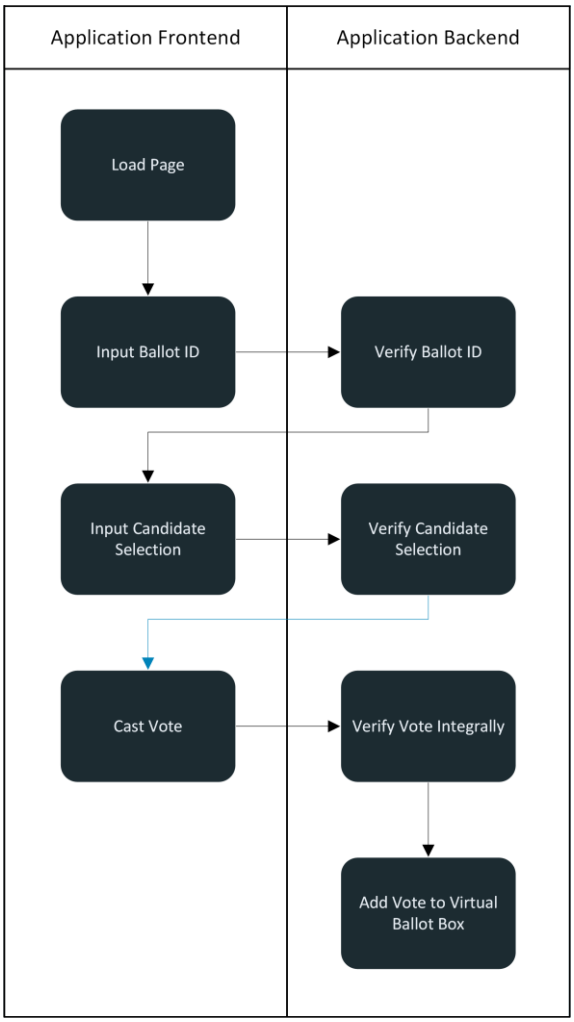


Figure 28 - Ballot Casting Process

Upon receipt of a ballot, the voter can scratch the ink off the ballots to reveal the 3-digit representative candidate IDs, log into the application, and visit `/${electionID}/vote`. The app will firstly check if the election generated any ballots for the voter by calling `/api/vote/checkAccount`. This API will search through the generated ballots for a user with the logged in user's ID. If one if found, the website will ask the voter to input their ballot ID.

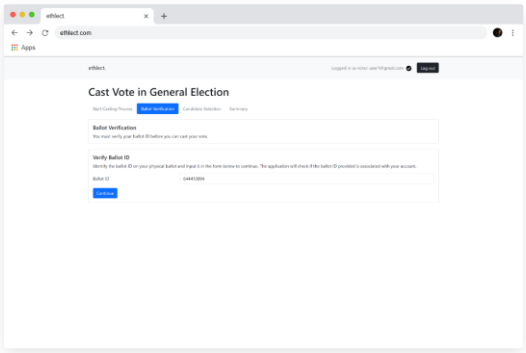


Figure 29 - Ballot ID Verification

The frontend will then send a request to `/api/vote/checkBallotID` which will search for a ballot with the introduced ballot ID in the ballots database that is assigned to the logged in user, if one is found, a success message is returned to the frontend which then allows the voter to input the 3-digit representative candidate IDs of the candidates they wish to vote for in preference order.

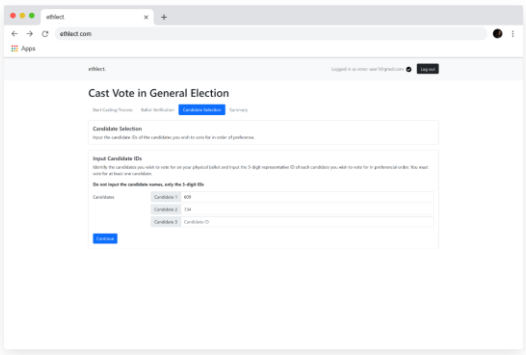


Figure 30 - Input of Candidate Selection

When submitting this page, the application sends a request to `/api/vote/checkCandidateIDs`. This API will check that the 3-digit IDs introduced are in fact present on the ballot with the introduced ballot ID. If this is the case, the voter is directed to the confirmation tab where a summary of the inputted information is provided. They can then cast the ballot.

This sends a request to `/api/vote/cast` which will run the same verifications on the cast ballot again. If they check out, the app will ensure the ballot was not cast already and then add it to the ballot box array. This should be a read-only part of the database that does not allow mutation after the initial data entry. The ballot added contains the user ID of the voter that cast it, the ballot ID, the constituency of the ballot, and the 3-digit representative IDs and encrypted candidate IDs of the IDs voted for in order of preference. If the process is successful, the voter will receive a success message.

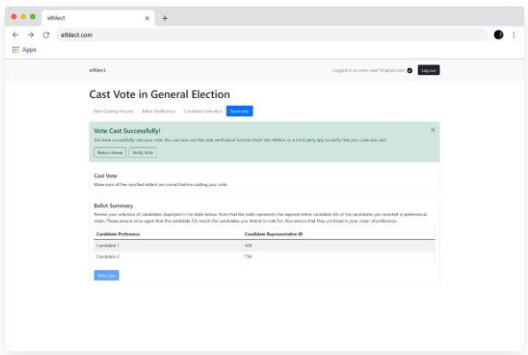


Figure 31 - Ballot Casting Successful

Ballot Verification

Voters can verify that their ballots were recorded successfully in the ballot box. This can be done directly, using the application, or independently by

downloading the ballot box and finding their ballot in it.

Direct Verification

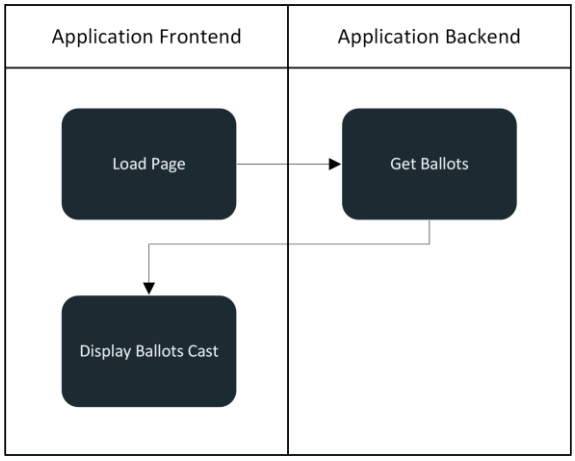


Figure 32 - Direct Ballot Verification Process

The voter can access `/${electionID}/verify` to verify the ballot they cast using the application. When loading this page, the frontend will send a request to `/api/election/verifyBallot`. This route will index through the ballot box, searching for a ballot cast with the user ID of the user logged in. The API will resolve all the ballots cast by the respective user and the frontend will display them on the page.

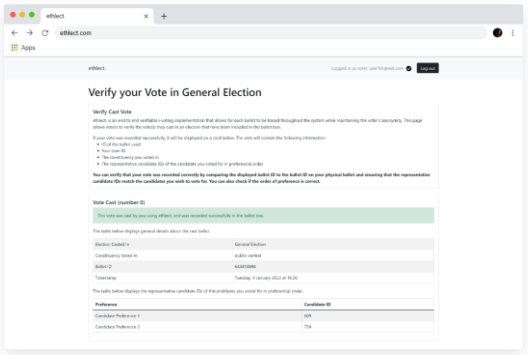


Figure 33 - Ballot Verification Page

The page will display when the ballot was cast, the election it was cast in, its constituency, and finally the recorded list of 3-digit candidate IDs in the order of preference they were recorded in.

From here, the voter can ensure the recorded candidate IDs match their desired vote by identifying the candidates on their physical ballot with the displayed IDs.

All the votes cast by the voter are displayed here (should they cast multiple votes). In this case, the voter will be informed that only their last vote will be tabulated.

Indirect Verification

Ballots can also be verified independently by a voter should they wish to do so. This is easily done by accessing /${electionID}/audit. The audit page[25] allows for all public information related to the election to be downloaded by anyone. The ballot box is made public through this page and the voter can download it (the ballot box will update during the election period; the download is timestamped to help identify the time at which the state of the ballot box reflects the downloaded file).

The voter can open the downloaded JSON file in an editor and search for `ballotID: ${myBallotID}` where `myBallotID` is the ID of the ballot cast (the lookup can also be

[25] see auditability in this section

done by userID). The voter can then verify that the 3-digit IDs in the ballot object represent their intended vote.

Anonymity

Ultimately, anybody can download the ballot box and search through it, but unless they have access to the physical ballot used to cast a vote, they will not be able to identify what candidates were voted for (they will only see the 3-digit representations which cannot be traced back to the candidates).

It is instrumental that the voter securely disposes of the ballot thereafter to prevent this kind of attack.

Election Tabulation

The election tabulation process is the final process in the election. This process is done digitally using the application and involves the cryptographic shuffling and decrypting of ballots.

The process is started by the admin by visiting `/admin/${electionID}/tabulate`. The page will check the attribute `electionTabulating` on the election document. If this is set to false (which it is by default), the page will display the option to start the tabulation process.

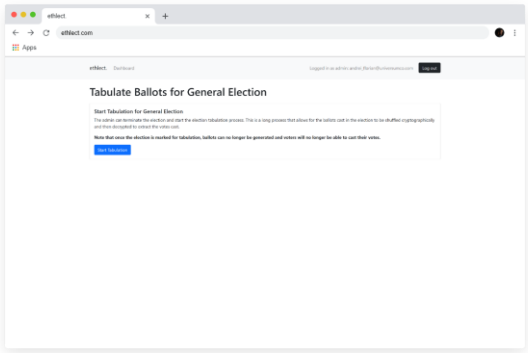


Figure 34 - Start Tabulation Process

By pressing the button, the app will send a request to `/api/election/startTabulation`. This route essentially transfers all ballots from the ballot box into the shuffles array in the database.

All identifying information is removed from the ballots and only the last ballot cast by every voter is transferred. Ultimately, each transferred ballot only contains the list of encrypted candidate IDs in preferential order. If the transfer is successful, the backend will add the ballots to the shuffle array as an object and will then set `{electionTabulating: true}`.

The frontend will display a success message and the admin can refresh the page. They will be presented with a user interface split into three sections:

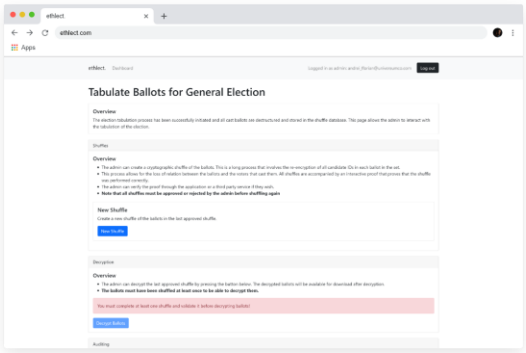


Figure 35 - Tabulation Page

The first section titled “Shuffles” displays all the shuffles performed on the ballot set together with the option to create a new shuffle. The second section titled “Decryption” allows for the decryption of the ballot set (provided at least one shuffle was performed and accepted). Finally, there is a link to the audit page too.

The admin can now shuffle the ballots and then decrypt them using the options made available.

Shuffle Ballots

By pressing the “new shuffle” button on the tabulation page, the admin is directed to `/admin/${electionID}/tabulate/shuffle`. Here they are presented with a button that they can press to create a new shuffle.

Unless this is the first shuffle, the previous shuffle must be approved by the admin before they can create a new shuffle (this will be explained in a bit).

Figure 34 illustrates the abstract of this process.

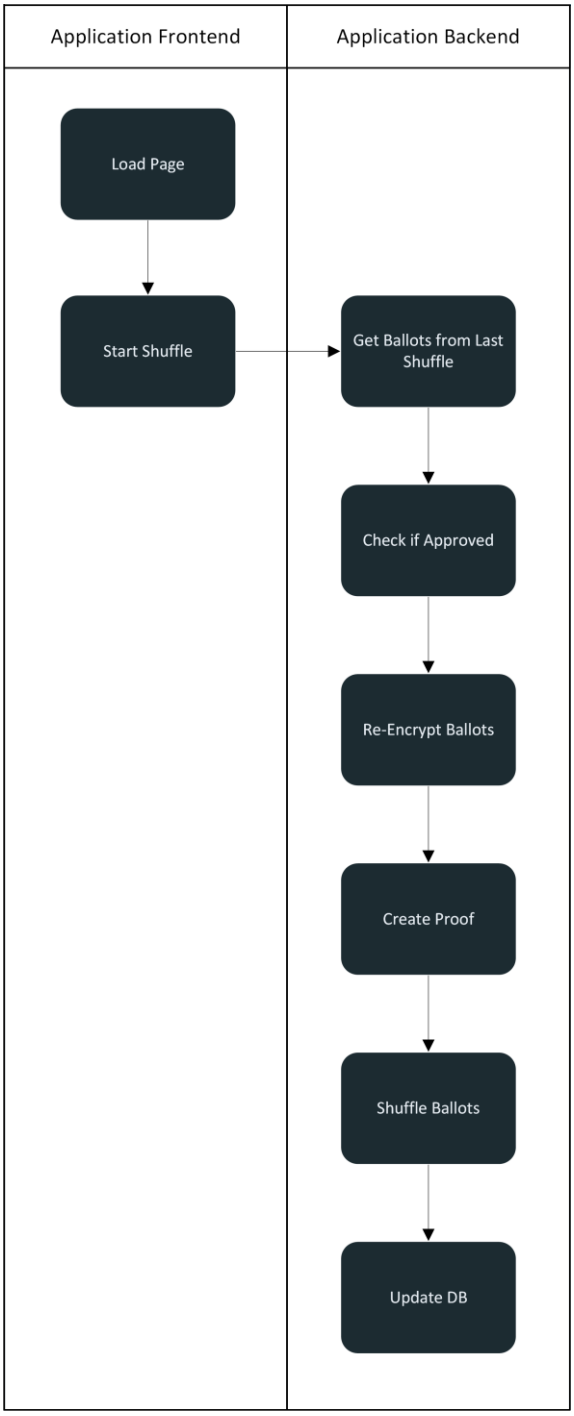


Figure 36 - Ballots Shuffle Process Diagram

After starting a new shuffle, the frontend will send a request to `/api/election/shuffle`.

The backend will first check if this is the first shuffle, if not, it will check if the admin

approved the last shuffle and only continue if they did.

The application will then get the outputted ballot set from the previous shuffle and re-encrypt all candidate IDs of all ballots. The app will then create an indirect zero-knowledge proof (in a similar fashion as when generating the ballots) (appendix 5).

The resulting re-encrypted ballot set will be shuffled and then the shuffle object will be added to the database as follows:

1. A shuffle ID will be generated (all shuffle proofs have a numeric ID which is an incrementation of the previous object's ID in the shuffles array)
2. The output ballots of the previous shuffle (or transfer) will be added to the `inputBallots` array.
3. The output ballots of this shuffle will be added to the `outputBallots` array.
4. A timestamp will be added
5. The proof will be added together with the property `{approved: false}`

If the shuffle is successful, the admin will receive a success message and can return to the tabulation page. They now need to either confirm or reject the shuffle.

Ideally the shuffle would be audited independently before it is approved or rejected based on the verification.

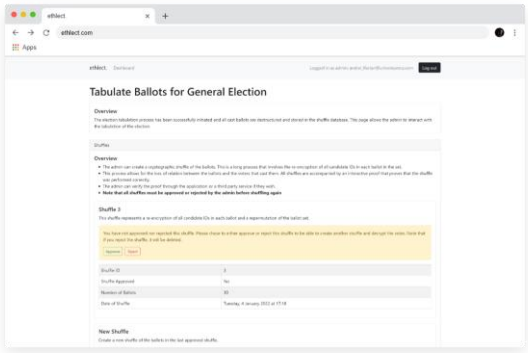


Figure 37 - Verify Shuffle

Should the admin approve the shuffle, a request will be sent to `/api/election/validateShuffle`. This API will mark `{approved: true}` on the shuffle in question. Should the admin choose to reject the shuffle, a request will be sent to `/api/election/deleteShuffle` which will in turn delete the shuffle from the database.

Decrypt Ballots

After shuffling the ballots and approving the shuffle(s), the admin can decrypt the ballot set and reveal the votes. The admin can press the "Decrypt" button on the tabulate page, this will direct them to `/admin/${electionID}/tabulate/decrypt`.

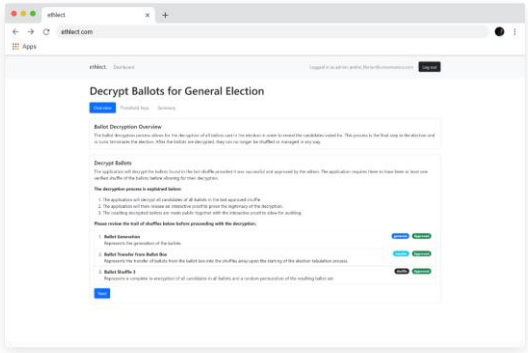


Figure 38 - Ballot Decryption Page

Figure 40 shows this process in abstract. After loading the page, the admin will be presented with an overview of the processes that the ballots underwent until present. The admin can press next and will then be asked to share the election's public encryption key with the keyholders to allow them to encrypt their threshold keys and return them to the admin to be inputted in the form (in the same way as in the ballot generation process).

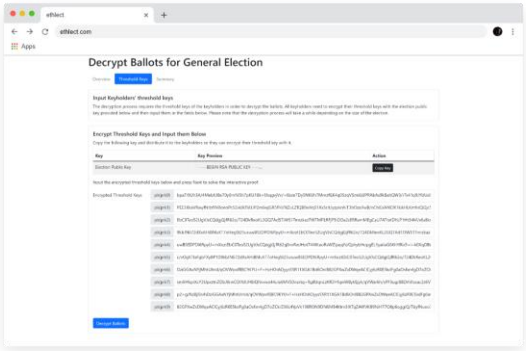


Figure 39 - Input Encrypted Threshold Keys

The admin can finally press the decrypt button on the decryption page, this will send a request to `/api/election/decrypt`.

The backend will first decrypt the threshold keys, combine them, hash the resulting key and compare it to the hashed private key in the database. If the keys match, the backend will check if the ballots were shuffled at least once and if the last shuffle was approved. If so, it will decrypt all encrypted candidate IDs of all ballots resulting in the ballots consisting of plaintext 6-digit candidate IDs arranged in order of preference (appendix 6).

The application will create a direct zero-knowledge proof of the decryption.

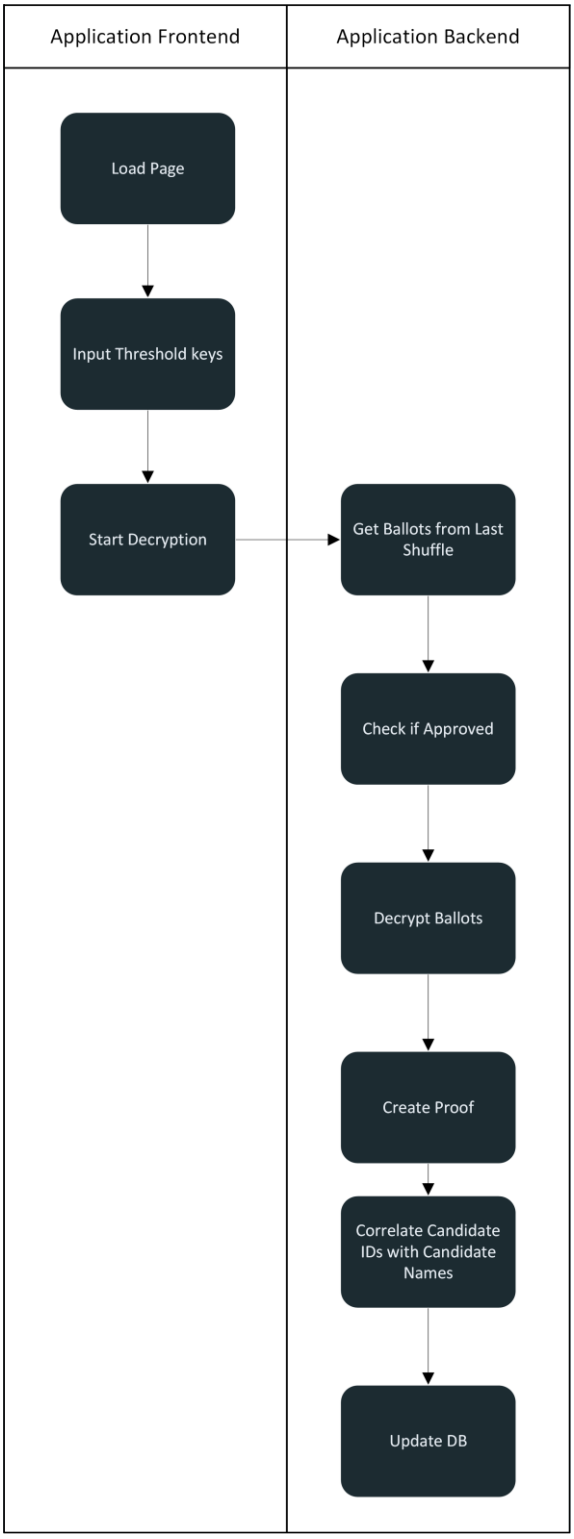


Figure 40 - Ballots Decryption Process Diagram

The application will then reference the candidates array in the database and will correlate each candidate ID with the candidate name they represent. All this data will finally be added to the election document in the database as follows:

1. A shuffle object will be added to the shuffles array containing the shuffle ID, encrypted ballots inputted, decrypted ballots (containing the candidate IDs) outputted and a timestamp.
2. The ballots containing the candidate names will be added to the database in the `electionResults` array.
3. The property `{electionComplete: true}` will be set.

The election is now over. The resulting decrypted ballots can be merged with the paper ballots cast during the election and then counted[26] or digitally counted.

Auditability

Auditability is crucial to ethlect. as it allows for stages of the tabulation process to be verified by third parties, hence confirming the correctness of the process.

Every election has an audit page publicly accessible at `/${electionID}/audit`. This provides access to all public information about the election.

It is very important to ensure that the data downloaded represents the actual data being used by the application. To ensure this, the application must use a secure file store such as solutions provided by AWS[27]. These files should be updated whenever new entries get added to the database.

For simplicity when developing, whenever a download request is sent to the application, it will compile a file in the frontend from the requested data from the election document.

Database Structure

Table 5 shows all the fields in an election document in the database, provides a short description and notes if they are publicly available.

Field	Data
electionID (public)	The unique ID of the election.
electionVerified (public)	Marked as true once the keys are verified.
electionName (public)	Name of the election.
electionDescription (public)	Description of the election.
electionStart (public)	Date of election start.
electionEnd (public)	Date of election end.
electoralPeriod (public)	True if the election is accepting ballots.
electionTabulating (public)	True if the electoral period is over and the election is tabulating.
elGamal (public)	The $g, y, p, SHA256(x)$ values

[26] As outlined in Objectives of ethlect.

[27] https://aws.amazon.com/s3/

rsaKeypair (private)	The RSA keypair used by the app to secure transmission of data
constituencies (public)	An array of objects. Each constituency has a name, the number of seats, and an array of candidates running for it. Each candidate object has a candidate name, and the IDs representing that candidate.
electionResults (public)	An array of the decrypted ballots where the candidate IDs are represented as the candidates' names.
ballots (private)	The generated ballots. Each ballot object contains the ballot ID, user ID of the user associated with the ballot, the constituency and the candidates voted for. Each candidate object has a 3-digit representative value and the encrypted candidate ID.
ballotBox (public)	The ballot box contains all cast ballots. The format is the same as the ballots array.
shuffles (public)	The shuffles array contains records of all processes involving ballots in the system. The results and accompanying proofs of the ballot generation, transfer, shuffle, and decryption processes are found here.
electionComplete (public)	True if the ballots are decrypted and the election is complete.

Table 5 – Election Document Structure

Accessing Election Information

A visitor can download the election constituencies array from the database, the ballot box, and the election results array. Auditors can use the constituencies database to ensure that the correlation between the election results and candidates is correct. It can also be used to verify the decryption process to ensure that the decrypted candidate IDs are present in the constituencies array.

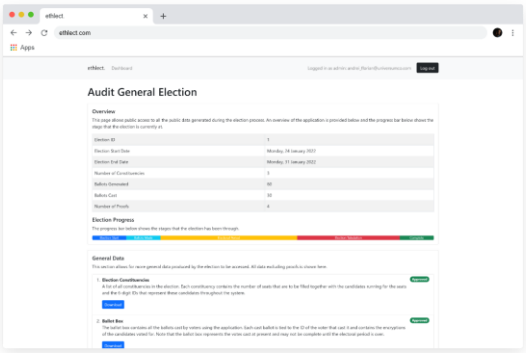


Figure 41 – Election Audit Page

The ballot box array can be used to independently verify a ballot was recorded as intended. Finally, the election results can be accessed.

Accessing Proofs

Four types of proofs are created by the application:

1. A ballot generation proof is created to attest the correctness of the ballot generation process.
2. A transfer proof is created which proves the correct transfer of ballots from the ballot box into the shuffles array.
3. Proofs are generated for every shuffle created by the system.
4. A proof of decryption is created by the system to certify the correctness of the decryption.

All of these proofs are incorporated in a shuffle object in the shuffles array. The

shuffles array contains entries with the following structure:

Field	Data
shuffleID	A unique ID for the shuffle
shuffleType	Either one of the 4 proof types listed earlier
inputBallots	The ballots inputted into the workflow that were processed. In the case of a ballot generation, these are the plaintext ballots, for all others they are the ballots in the previous shuffle/transfer.
Approved	Denotes if a shuffle is approved.
Timestamp	The time that the operation was completed at.
Proof	The proof of correctness of the process.

Table 6 - Shuffle Object Format

The proof of correctness contains all information needed (elGamal keys, etc.) to validate the specific operation performed on the ballots (generation, shuffle, decryption). The appendices discussing these workflows explain mathematically how these proofs are generated and verified.

All shuffles in the shuffles array are downloadable by the visitor. In conclusion, ethlect. exposes all parts of the application needed for a complete independent audit of an election's integrity. This allows for independent parties to prove an election's integrity.

Proving Proofs

I created a small application that allows for an auditor to upload the proof file outputted by the ethlect. application and check it. This allows for the proving of correctness of the generation, shuffle, and decryption processes with the click of a button requiring no technical know-how.

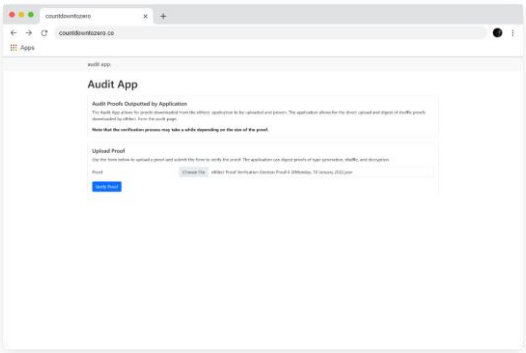


Figure 42 - Upload File to Audit Page

The auditor simply uploads the proof file as provided by the application to the form on the audit app and submits it. The application will run multiple checks and attempt to prove the proofs provided and will return a success/failure message respectively.

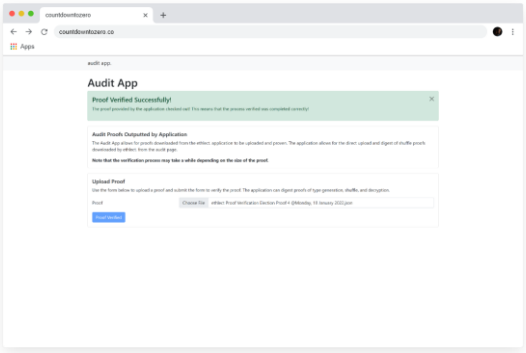


Figure 43 - Verification Successful

Security Analysis

In an internet voting application, security is the greatest concern. Because of the great stakes involved, it is essential that the integrity of the election is upheld

regardless of the type of attack on the application.

I have thoroughly tested the theory behind the processes and the application to ensure that it upholds its claims.

In the end, I am a 17-year-old with a limited budget and limited time, I cannot guarantee that I assessed every possible attack scenario, which in fairness, no product team can.

Verification Chain

Figure 40 illustrates all the stages that ballots go through in the application. All these stages are verified to ensure the integrity of the ballots throughout the system.

By verifying each stage, the ballots can effectively by traced throughout the system while maintaining the secret ballot. Should a ballot be mutated, removed or added to the system at any stage, this mutation of the ballot set would be detected by the verifier.

In this sense, ethlect. is truly end-to-end verifiable. ethlect. also implements workflows that allow for action to be taken in the case of any abnormality identified.

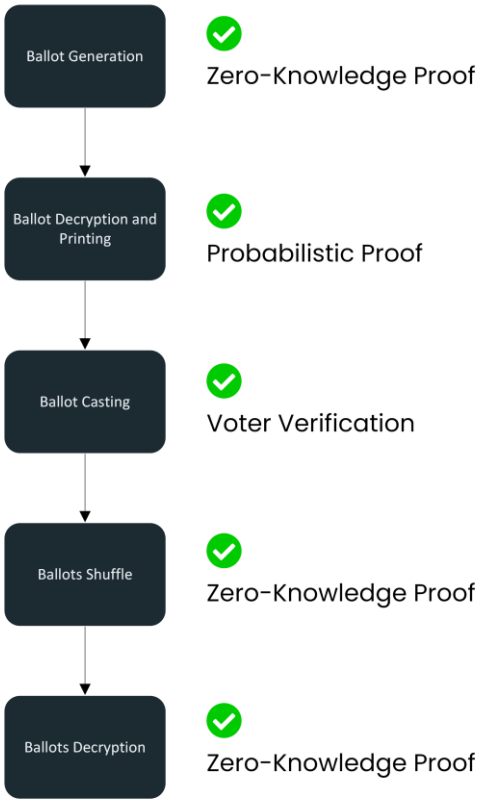


Figure 44 - Verification Chain Diagram

Should the proof provided by the generation algorithm not check out, the application can always generate the ballots again. The same goes for the printing, shuffle, and decryption workflows.

Should a voter consider that their ballot was mutated or should they have been coerced into casting a vote, they can cast another one. 2 ballots are generated for every voter and one of these is sent out to the voter initially. The voter can request the second ballot be sent out to them should they wish to cast another vote. Thereafter more ballots can be generated and distributed.

The application does not rely on any trusted hardware or software as the whole system is verifiable. In the case of a successful attack on any aspect of the system, it would be detected through verification.

General Security

The general security of the application was tested. Specifically, all API routes were pen tested to ensure that no malicious actor can successfully send requests that execute functions in the app.

All pages and APIs were checked to ensure that only authorised logged in users (voters and/or admins) can access the pages and APIs.

The pages served to the client were thoroughly inspected to ensure no secret information is shared.

Access to information in all methods and routes in the application was assessed to ensure that no aspect of the application has access to information it does not have authority to access.

In summary, a general black-box, and white-box test was conducted on the application.

Documented Attacks

This section walks through the attacks that could be carried out on the various workflows of the application and the proposed resolution for these attacks.

Voter Registration

- Forcefully skipping to third tab and submitting request. The application backend checks the register during the registration process and after it.
- Entering duplicate email address. Unique email address is needed.
- Providing false proof of identity. The identity verification process is handled by Stripe Identity which uses industry standards identification techniques, the webhook Stripe posts to ensures that the data sent is signed by the Stripe Server.

Voter Login

- Brute force login attacks. All passcodes are salted, and the attacker needs to have access to the voter's 2FA application to successfully log in.

Election Creation Process

- Overload of csv file or wrong format. The application checks for correct formatting before processing data and documents that are too long are discarded.
- Provision of incorrect public keys by the admin. The verification stage of the process will fail.
- Failure of admin to distribute encrypted threshold keys or collect and input re-encrypted threshold

keys. The verification stage of the process will fail.

Ballot Generation Process

- Error in format of constituencies array. The process will error out.
- Election is tabulating. Process will error out.
- Error in the ballot generation process. If process does not error out, the proofs provided will not work out.

Ballot Printing

- Exposure of threshold keys. The election can be restarted, and new keys can be generated.
- Error decrypting and/or printing ballots. If the process does not report an error, it will be detected in the auditable ballots.
- Exposure of sensitive ballot data (candidate representative IDs). The ballots can be generated again. The facility used to print the ballots should be monitored and all code and machines used should be checked to ensure they do not compromise voter anonymity.

Ballot Casting

- Interception of ballot. An attacker would also need control over the voter's computer and/or login credentials to cast a ballot in their place. If an attacker were to reveal the candidate reps on the ballot, the voter would notice the tampering and will request a new ballot.
- Access user login credentials. The attacker still needs access to the user's phone running the 2FA app and their physical ballot (the ballots are tied to users so that an attacker cannot pick up a ballot that is valid for all voters in a constituency and cast it should they have access to the voter's credentials).
- Spyware on voter hardware. The voter at no point introduces information digitally that can be traced back to identify the candidate selection.
- Malware on voter hardware. Should an attacker be able to change the vote cast by a voter, this action would be futile as the attacker cannot organise a strategic attack where they aim to vote for a specific candidate because they do not know which 3-digit number is correlated to which candidate. If such an attack were to go ahead for the sake of interfering with the election in a random manner, the voter can notice this modification and cast another ballot.

Ballot Verification

- Should the voter choose to verify their ballot through the website and have malware that changes the values displayed. If the new values do not match the intended selection, the voter will naturally cast another ballot. If, however, the values are modified to make the voter think that their intended selection was recorded when in fact it was not, this attack would have to be carried out on a large portion of voters to sway the election. Some of these voters may choose to verify their ballots independently. If a ballot verified independently has a different verification outcome to one verified directly, the attack will have been identified and voters will be encouraged to verify their ballot independently or using a third party.

Ballot Shuffle

- Error in the shuffle process. If no error is displayed, the proof released by the shuffle will not check out.
- Failure to verify last shuffle. If the last shuffle was not verified, or contains malformed data, the shuffle process will return an error to the admin.

Ballot Decryption

- Incorrect keys inputted. If the admin failed to input the keys correctly or the keys are incorrect, the application will return an error to the admin.
- Failure to verify last shuffle. If the last shuffle was not verified, or contains malformed data, the shuffle process will return an error to the admin.
- Error in the decryption process. If no error is displayed, the proof released by the process will not check out.

Election Audit Page

- Private independent auditors should be given access to the election's database directly (excluding all private data). This is to ensure that should the information displayed on the audit page be incorrect, these auditors can notice the misrepresentation.
- The admin of the election should only investigate a reported issue in the proof provided if multiple sources spotted it and can prove the error mathematically.

Election Keys

- Should the election's RSA keypair be leaked, a new one can be generated to use for future communications.

- Should the El Gamal private key, or all keys involved in the shuffling of the ballots be leaked, voter anonymity would be compromised but the integrity of the ballots and thereby results would not be.
- Access to the admin account needs access to the admin's 2FA application. Ideally a 2FA airtight device would be used in this case. This device would be stored safely to prevent access to it. The compromise of an admin account allows for the interference with elections and creation of new ones. Ultimately, all actions that admins perform produce verifiable material, therefore should an admin account be compromised, this compromise would be easily identifiable. Importantly, access to an admin account cannot compromise the integrity of an existing election nor compromise voter anonymity.

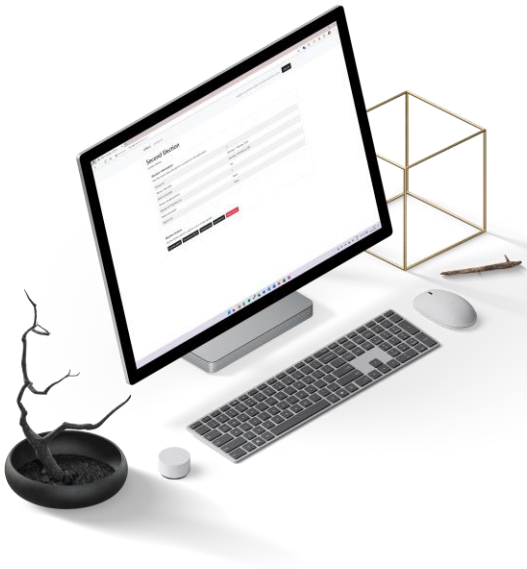
Mock Election

As part of the testing conducted for the application, I conducted a complete mock election using the system. In summary, a voter pool size of 35 voters spread across 3 constituencies attempted to cast a vote in an election running on the platform.

The test was a complete success. The ballots were cast as intended, the tabulation process successfully shuffled and decrypted the ballots, and the proofs checked out.

Details about the election are in the project's GitHub repo in the links section of the paper.

Conclusion



I really enjoyed working on ethlect. I've always searched for impact in my projects, and I believe that I managed to create something truly impactful.

I started off this summer researching internet voting, and election systems. I was determined to try making my own i-voting application, one that was unique. I delved into cryptographic techniques and then theorised the application. After a while I started developing and testing it.

In conclusion, I built the world's first truly end-to-end verifiable internet voting

system and designed it specifically for Irish General Elections running on PR-STV.

ethlect. is an innovation in the internet voting space as it is the first and only application that guarantees end to end verifiability and frontend security. ethlect. is also meticulously designed for integration with the Irish Electoral system, being the first internet voting application for PR-STV type elections and having one of the fastest voter registration workflows of all apps thanks to its strategic integration with the Dublin Voter Registry and Stripe Identity.

I hope you enjoyed the paper 😊.

Appendices

In-depth explanations of the processes outlined in the paper.

Appendix 1

The 2-factor authentication system implemented in the project is referring to time-based one-time passwords. This protocol allows for the generation and verification of one-time passcodes that are time sensitive.

Two parties are involved in this system: the authenticator who verifies the token inputted by the authenticatee who generates them for authentication.

This system requires both parties to agree on an epoch time to use and a secret value k.

In most cases, including ethlect, the authenticator (application) will generate this value and share it with the authenticatee via a QR code.

$$C = \frac{T}{T_x}$$

Unique, TOTP tokens are generated using the formula above where T represents the current epoch time and T_x represents the length of one-time duration (this is the period of time that the token refreshes at, normally 30 seconds).

After the TOTP token is generated by the authenticatee, the time based token is concatted with the agreed secret and used as inputs to a one-way HMAC function[28].

$$HMAC(k \,||\, C)$$

This function returns a 40-character unique value. The process then uses dynamic truncation. This involves taking the binary representation of the 20th character of the hash and converting the first 4 bits to a decimal.

The resulting decimal d is outputted where $0 > d > 40$. The consecutive four characters after d are taken from the hash and converted to base 10 decimals.

[28] https://en.wikipedia.org/wiki/HMAC

This will result in a 10-digit long decimal which is modulo one million to cut it down to 6.

When authenticating, both the authenticator and authenticatee process the token in the same way. If the authenticator's calculated token is the same as the one inputted by the authenticatee, the verification succeeds.

Appendix 2

The Shamir Threshold Encryption Scheme allows for a key to be split into multiple shares to be shared out to different parties. The system uses polygon interpolation to allow for the original key to be reconstructed out of a defined fraction of the split threshold keys.

A random polynomial $P(x) = a_{k-1}x^{k-1} + a_k - 2x^{k-2} + \cdots + a_2x^2 + a_1x + s$ is selected where k is the desired threshold number of shareholders required to compose the shared key, the a_i are randomly selected integers in the range $0 < ai < p$ where p is the prime of the El Gamal System (appendix 3), and s is the secret value, in this case the El Gamal private key. Parties $P_1, P_2, \ldots P_n$ (with $n \geq k$) each hold a point on this polynomial (P_i holds the value $P(i)$, this point is represented by the key they hold).

It can be shown that any k of these points allow interpolation of the polynomial and discovery of the secret value $s = P(0)$.

Appendix 3

The El Gamal cryptosystem is a cryptosystem based on elliptic curve cryptography and an application of the Diffie-Hellman system. The system is initiated through the generation of a large prime p and a generator g based on the multiplicative subgroup of integers in the space $modp$ are generated. El Gamal also works with multiplicative and exponential groups, but these do not produce the desired workflows needed to satisfy end-to-end verifiability standards. The application will also generate a private key x where $0 < x < p$ and a derived public key y where $y = g^x$. The values g, p, y are made public.

A party can encrypt a message by only having access to the values g, p, y, by generating a random key r where $0 < r < p$, the pair $(a, b) = (My^r modp, g^r modp)$ is created where M represents the message being encrypted as an integer.

A message encrypted as such can be re-encrypted by a party that only has access to the values g, p, y and the ciphertext (a, b) by generating another random value r' and forming the pair $(a', b') = (My^{r'} modp, g^{r'} modp)$. This re-encryption can be decrypted by the same private key x.

The decryption of an El Gamal pair is achieved by computing:

$$\frac{a}{b^x} mod\ p\ = \frac{My^r}{g^{rx}} mod\ p\ = \frac{Mg^{rx}}{g^{rx}} mod\ p$$
$$= M\ mod\ p\ =\ M$$

Appendix 4

During the ballot generation process, the application will have already generated the plaintext ballots B as described in the workflow before encrypting the ballots.

All candidates of all ballots in the ballot set are encrypted to form a new ballot set B' using the encryption technique outlined in appendix 3. This process is quite straight forward.

The application then needs to provide a zero-knowledge proof that $B \equiv B'$. The application uses the method described in a Microsoft research paper to achieve this[29].

The challenge for the application is that it cannot simply release the key used to encrypt the ballots as this would prove direct equivalence between the two ballot sets which would defeat the purpose of a shuffle and ballot generation (which is to ensure that ballots cannot be individually traced through the shuffles).

A probabilistic zero-knowledge non-interactive proof is provided by the system.

The application encrypts n ballot sets $B'_1, B'_2 \dots, B'_n$ using different keys $k_1, k_2 \dots, k_n$.

These ballot sets are encrypted in the same way as ballot set B'.

The collection of ballot sets $B_1, B_2 \dots, B_n$ are fed into a SHA256 cryptographic hashing function that produces a unique value. The first n bits from the output of the function are used as challenge bits (ones or zeros), this is the Fiat-Shamir Heuristic[30]. This results in bits $c, c_2 \dots, c_n$ being generated.

This is done to keep the proof non-interactive. Proofs of knowledge are typically interactive where two parties are involved: the prover (which proves their knowledge) and the verifier (which ensure the prover has the knowledge). In this interactive model, the challenge bits would be provided by the verifier. This is not implemented here out of practicality as it would be difficult for the application to continuously have to respond to challenges.

The application will now index through the generated ballot sets.

For each i such that $c_i = 0$, the plaintext ballot set B_i is proven to be equivalent to B'_i by releasing the encryption key k_i.

For each i such that $c_i = 1$, the key $(r - k_i) mod (p-1)$ is calculated where r is the

[29] https://www.usenix.org/legacy/event/evt06/tech/full_papers/benaloh/benaloh.pdf

[30] https://en.wikipedia.org/wiki/Fiat%E2%80%93Shamir_heuristic

key used to encrypt the plaintext ballots initially.

This is repeated n times by the application. Should the proofs provided check out, the encryption is proven to be correct through probability. An attacker would have to produce 2^n ballot sets in order to defeat the proof. Setting $n \approx 100$ should be sufficient to prevent this type of attack. The proofs are shuffled together with the encrypted ballot set B' and made publicly available alongside the ballots. This shuffle is done by alphabetically sorting the encrypted ballot set.

An auditor can verify a type 1 proof ($B \equiv {B'}_n$) by encrypting the plaintext ballot set B with the provided key k_n and permuting the resulting ballot set alphabetically (note that in the case of ballot generation, the permutation is done for candidates within a ballot and for the ballot set as a whole).

A proof of type 2 ${B'}_n = B'$ can be proven using the provided value $r - k_i$ by proving that all encryptions of ballot set ${B'}_n$ (as (a, b)) and B' (as (a', b')) are encryptions of ballot set B and can be decrypted using the same key. This is done by verifying the following for each candidate ID.

$$\frac{b'}{b} = g^{r-k_i}$$

$$\frac{a'}{a} = y^{r-k_i}$$

If all proofs check out, the auditor can be certain of the correctness of the encryption. This proof is an application of homomorphic encryption.

Appendix 5

The shuffle process is very similar to the generation process. Here, all candidates of all ballots in the inputted ballot set B' (either from the ballot box transfer or the previous shuffle) are re-encrypted using the method in appendix 3.

The same indirect zero-knowledge proof of equivalence is used in this case as the one described in appendix 4. The only difference is that when permuting the resulting sets, the candidates within each ballot are not permuted (as this would change the candidate preferences).

Appendix 6

The ballot decryption process consists of the application decrypting all encrypted candidate IDs in all ballots using the key x as described in appendix 3.

After decrypting the ballots, the application provides a discrete logarithm equivalence proof (DLEQ). This proof aims to show that should the decrypted ballot set be encrypted with the key used to generate the ballots; the resulting encryption would be the same. This is sufficient to attest the correctness of the decryption

A random value k is selected by the application which will be used for all candidates in all ballots.

The application calculates $c = SHA256(g^k \parallel b^k)$ where SHA256 is the SHA256 hash function and (a, b) is the ciphertext representing the encrypted candidate ID being verified. The app also calculates $r = k - cx$ where x is the El Gamal private key.

This proof is released for all candidate IDs of all ballots. The application releases the pair $(c.r)$ for each candidate to be verified publicly.

In order to verify these proofs, the verifier checks if $c = SHA256(g^r * y^c \parallel b^r * P^c)$ where $P = \frac{a}{M'}$, where M is the decrypted candidate ID being verified. If the check passes for all ballots, the decryption correct.

Bibliography

Burson v. freeman, 504 U.S. 191 (U.S. Supreme Court 1992).

Anwar, N. K. (2020). Advantages and Disadvantages of e-Voting. *Nanyang Tenhnological University*, 1-12.

Appel, A., DeMillo, R., & Stark, P. (2019). *Ballot-Marking Devices (Bmds) Cannot Assure the Will of the Voters.* Appel, AW.

Batt, S. (2019, November 14). *How Electronic Voting Works: Pros and Cons vs. Paper Voting*. Retrieved from Makeuseof: https://www.makeuseof.com/tag/how-electronic-voting-works/

Borg Psaila, S. (2011, May 2). *Right to access the Internet: the countries and the laws that proclaim it*. Retrieved from Diplomacy.edu: https://www.diplomacy.edu/blog/right-access-internet-countries-and-laws-proclaim-it/

Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P. S., . . . Vora, P. L. (2010). *Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy.* Massachusetts: Caltech/MIT Voting Technology Project.

Citizens Information. (2021, October 27). *General Elections*. Retrieved from Citizens Information: https://www.citizensinformation.ie/en/government_in_ireland/elections_and_referenda/national_elections/the_general_election.html#l89735

Crunchbase. (2021, October 26). *Crunchbase Report on Voatz*. Retrieved from Crunchbase: https://www.crunchbase.com/organization/voatz/company_financials

Department of Housing, Planning and the Local Government. (2020).

Costings Breakdown GE 2020. Dublin: Not publically available, most enquire for copy.

e-estonia. (2022, January 4). *Interoperability services*. Retrieved from e-estonia: https://e-estonia.com/solutions/interoperability-services/x-road/

Fischer, J. (2004). *Election Cost Survey Results.* Ace Project.

Friel, B. (2005). Let the Recounts Begin. *National Journal*.

Green, M. (2019, December 3). Professor of Cryptography at Johns Hopkins University. (V. News, Interviewer)

Greenberg, A. (2012, March 23). *Shopping For Zero-Days: A Price List For Hackers' Secret Software Exploits*. Retrieved from Forbes: https://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/?sh=41193aa52660

Halderman, A. (2020, February 13). security expert and professor of computer science at the University of Michigan. (V. News, Interviewer)

Heiberg, S., & J., W. (2014). Verifiable internet voting in estonia In R. Krimmer and M. Volkamerm. *Proceedings of Electronic Voting 2014 (EVOTE2014)*, 7-13.

Houses of the Oireachtas. (2016, February 9). *Constituency Profiles*. Retrieved from Oireachtas: https://www.oireachtas.ie/en/how-parliament-is-run/houses-of-the-oireachtas-service/library-and-research-service/use-our-research/constituency-profiles/

Houses of the Oireachtas. (2016). *Election Turnout in Ireland: measurement, trends and policy implications.* Dublin: Oireachtas.

Houses of the Oireachtas. (2020). *33rd General Election Results.* Dublin: Library of the Oireachtas.

Houses of the Oireachtas. (2020). *General Election 2020: A Statistical Profile.* Dublin: Oireachtas Library and Research Service.

IDEA. (2022, January 4). *Voter Turnout Database*. Retrieved from IDEA International: https://www.idea.int/data-tools/data/voter-turnout

Karp, J., Banducci, A., & Susan, A. (2000). Going Postal: How All-Mail Elections Influence Turnout. *Political Behavior*, 223-239.

Krimmer, R., Duenas-CId, D., & Krivonosova, I. (2020). New methodology for calculating cost-efficiency of different ways of voting: is internet

voting cheaper? *Public Money & Management*, 17-26.

Moore, L. (2019). *Under the Hood: The West Virginia Mobile Voting Pilot.* US: Voatz.

Mugica, A. (2015, February 26). CEO of Smartmatic. (T. Guardian, Interviewer)

National Academies of Sciences, Engineering, and Medicine. (2018). *Securing the Vote: Protecting American Democracy.* Washington, DC: The National Academies Press.

NDI. (2013, December 17). *Electronic Voting and Counting around the World.* Retrieved from NDI: https://www.ndi.org/e-voting-guide/electronic-voting-and-counting-around-the-world

Office of the Director of US National Intelligence. (2017). *Assessing Russian Activities and Intentions in Recent US.* New York: Office of the Director of US National Intelligence.

Park, S., Specter, M., Narula, N., & Rivest, R. L. (2021). Going from bad to worse: from Internet voting to blockchain voting. *Journal of Cybersecurity*, 1-15.

Rogers, E. (1962). *Diffusion of Innovations.* New York: Free Press, New York.

Seletsky, H. (2020, December 20). *Venezuela Holds Unofficial Referendum Using Blockchain, Millions of Votes Cast.* Retrieved from beincrypto.com: https://beincrypto.com/venezuela-holds-unofficial-referendum-using-blockchain-millions-of-votes-cast/

Solvak, M., & Vassil, K. (2016). E-voting in Estonia: Technological Diffusion and Other Developments Over Ten Years (2005 - 2015). *University of Tartu*, all.

Stuart-Mills, I. (2021, November 26). Dept. of Housing, Planning and the Local Government. (A. Florian, Interviewer)

Tsahkna, A. (2013). E-voting: lessons from Estonia. European View. 59-66.

Wikipedia. (2021, November 9). *Postal Voting.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Postal_voting

Wikipedia. (2022, January 4). *Elections in the Republic of Ireland.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Elections_in_the_Republic_of_Ireland

Wikipedia. (2022, January 4). *Electronic Voting.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Electronic_voting

Wikipedia. (2022, January 4). *Single Transferable Vote*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Single_transferable_vote

Wikipedia. (2022, January 4). *Universal Suffrage*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Universal_suffrage