ethlect.

An Investigation into Worldwide Electoral Systems and the Development of a Novel Internet Voting System.

Andrei Florian

andrei@andreiflorian.com

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

Project Paper

Intended to provide a full analysis of the project

Date Released

18 April 2022

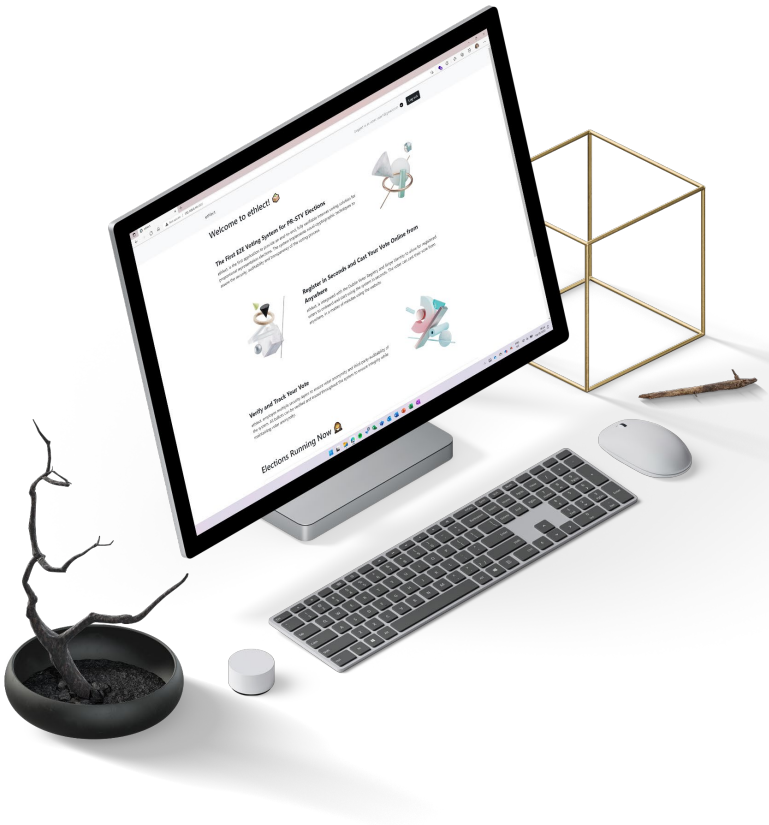

Table of Contents

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 how they once did 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 nurturing 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 sieving through papers on internet voting and accustoming myself to the cryptographic systems involved, I created a first-of-its-kind internet voting application tailored for PR-STV type elections, that uses a mixture of existing cryptographic techniques, and some that I designed myself.

Investigation Process

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

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

I found internet voting, and the technologies that enable the casting of digital ballots, particularly interesting. 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, exceeding that of the existing Irish Electoral System (Benaloh, Internet Voting, 2022). I finally conducted extensive testing and an election using the system to prove its functionality.

Paper Layout

This paper is split up into thirteen sections. I will start off by assessing the research conducted into the Irish Electoral System. I will then continue with research into challenges electoral systems face worldwide and introduce internet voting as a concept and implementation. I will finish with fully presenting my application and delving into both the theory and application.

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 suffrage 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 (PR-STV) mechanism was used (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 – typically around 1 for every 25,000 people (Citizens Information, 2021)).

When casting their ballot, 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. Ballot counting begins at 9AM the following day by count officials supervised by admin staff (Stuart-Mills, 2021). 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.

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

The equation above represents the calculation of a quota where p represents the valid poll count and s represents the number of seats in the 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 ballots (chosen randomly from the ballot set) are redistributed based on second preference

to the candidates that have not reached the quota in the first count.

Should no candidate reach the quota, the candidate with the smallest number of votes will be eliminated and their ballots will be redistributed to the other running candidates based on the second preference.

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, their surplus ballots are redistributed among the remaining candidates for the same constituency and should no candidate reach the quota, the ballots of the candidate with the smallest number of votes are redistributed (Department of Housing, Planning and Local Government Ireland, 2018).

Most general elections have between six and ten counts. Counting is typically complete in one day. (Stuart-Mills, 2021).

Analysis of the 2020 Irish 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, adj.)	€6.81
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 as of January 2022.

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

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).

Surplus Redistribution

The PR-STV model relies on the redistribution of surplus ballots to ensure all seats are filled in a constituency. This involves the redistribution of the surplus ballots of a candidate that exceeded the quota to other running candidates based on the next preferences on the ballots (Department of Housing, Planning and Local Government Ireland, 2018).

The Irish PR-STV system has a fundamental fault in the sense that the surplus ballots are defined by all ballots counted after the ballot that allowed a candidate to reach their quota (Department of Housing, Planning and Local Government Ireland, 2018).

Hence, the allocation of surpluses is not conducted in a deterministic manner but rather the ballots redistributed are determined by the order in which the ballots are counted.

Take for instance a situation where four candidates – John, Aisling, Eoin, and Sarah – are competing for two seats. If John exceeds the quota of two votes by another two votes, two of his ballots are to be redistributed to the other running candidates.

Take for instance that every ballot has any one of the remaining candidates marked down as a second preference: Aisling is the second preference on one ballot, Eoin is the second preference on another, and Sarah is the second preference on two ballots.

The question now is how to decide which ballots to redistribute (only 2 can be redistributed). The Irish System will redistribute all ballots that are counted after the second ballot (which allowed John to reach the quota). In that sense, the order in which the ballots were counted determines which two of the four ballots are redistributed.

Although Sarah is a second preference on two ballots, these ballots could be the first two counted, hence she would not receive any of the surplus ballots. Alternatively, it

is equally likely that she receives both ballots.

The worrying truth is that this systemic fault has the potential to change the election's outcome based on the order ballots are counted in. The electoral system, through its lack of use of technology and sole reliance on human counters, cannot facilitate the deterministic redistribution of ballots based on the frequency of second preferences (Martin, 2022).

Challenges Facing Electoral Systems Worldwide

Ireland's electoral system, albeit unique in many ways, is foundationally analogous to 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 investigated other systems implemented throughout the world and identified a list of challenges all systems commonly face:

Transparency and Auditability

In Ireland, the population's general impressions of the integrity of the Irish electoral system seems to be very positive (Stuart-Mills, 2021). This is 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 the election's integrity inarguably.

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] around the world, the only confirmation that the voter gets that their vote entered the system is their putting 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 electoral processes around the world, and the interactions with ballots are typically thoroughly supervised, there is no way to prove with certainty that the final tally consists of only cast ballots, which were tabulated correctly (National

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

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], ballots cannot be verifiably traced throughout the system.

This lack of quantitative proof of the correct operation of the system allowed Donald Trump and some of his supporters 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]. It is estimated that 30% to 40% of Republicans today believe that the 2020 election was unfairly conducted (Benaloh, Internet Voting, 2022).

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 any country: 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 margin of error when tabulating votes (Stuart-Mills, 2021).

Such is the case for all electoral systems that rely on the manual counting of votes.

Resource Efficiency

The organisation of elections and assurance of their integrity requires a lot of resources.

Elections have a financial cost. 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 (Department of Housing, Planning and the Local Government, 2020).

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

Elections also require a lot of people to operate. In the 2020 General Elections in Ireland, 17,986 people were employed in various jobs. This results in an employee to votes cast ratio of around 1:110 (Department of Housing, Planning and the Local Government, 2020).

Elections are time consuming; counts took days in the Irish 2020 General Election (Stuart-Mills, 2021) and weeks in the US 2020 Presidential Election (Wikipedia, 2022).

Catering towards Special Voters

Electoral systems in Ireland and other countries around the world often need to be adjusted significantly to cater towards special voters that cannot vote in person. This adjustment brings with it a financial and time cost to the election process and may discourage a portion of the population from voting.

Some voters may not be able to cast a vote in person due to sickness or inability to get to their nearest polling centre. Catering to voters abroad is also a problem many electoral systems face today; to facilitate foreign voters, countries would typically set up a limited number of polling stations abroad – although usually these polling stations are not sufficient to cater to the entire population abroad. Furthermore, it is often infeasible to organise polling stations for smaller scale elections abroad (Zagórski, et al., 2013).

Convenience

In the 2016 Irish General Election, 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).

It is widely known that many seemingly unrelated factors can have a direct impact on voter turnout. For instance, poor weather was stated as the reason for not voting by 5% of absentees from the 2016 Irish GE (Houses of the Oireachtas, 2016) and the impact of poor weather on elections proves to be a global issue (Gomez, Hansford, & Krause, 2007).

Voter Feedback

Systems that employ pen and paper ballot marking offer no means of providing voters with feedback regarding their ballot marking – a voter receives no confirmation that they correctly marked their ballot and must assume this.

Often, voters that ask for assistance for marking their ballots compromise their anonymity by exposing their candidate selection to the helper (Herrnson, 2022).

In the 2020 Irish 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 in certain constituencies (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.

Electoral systems have themselves experienced the changes technological breakthroughs bring to all 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 digital 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 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 for integrating technologies into the electoral process.

Internet voting is a subset of electronic voting that is defined as the casting of

[6] https://en.wikipedia.org/wiki/Electronic_voting_by_country
[7] https://en.wikipedia.org/wiki/Electronic_voting
[8] https://en.wikipedia.org/wiki/DRE_voting_machine

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 votes in elections from the comfort of their homes. Many countries have piloted internet voting schemes including Estonia, the Netherlands, the UK, and US (Wikipedia, 2022).

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 existing electoral systems, offering voters that wish to cast their vote online 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

Whereas most non-digital electoral systems attempt to provide integrity and ensure the secret ballot through obscuring the ballots' paper trail, as do Ireland's and America's systems, internet voting achieves these requirements 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. This allows for all ballots to be verifiably tracked throughout all stages of the system while persisting voter anonymity.

Ballot secrecy is an integral concept in fair elections. Voters' anonymities must be maintained to ensure that they cannot be coerced (Burson v. freeman, 1992). Well-designed i-voting systems allow for voters to individually verify that their ballots were recorded as intended and successfully included in the poll without compromising their anonymities (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 elections employing non-verifiable paper ballots (Benaloh, Internet Voting, 2022).

Precision

It is estimated that in the years following the introduction of EVMs (Electronic Voting Machines) in the US, as many as one million ballots that would have been missed by manual counting were included in the poll (Friel, 2005).

The tabulation process in internet voting applications is fully digital in most cases. This means that provided the application is operating as intended, the tabulation process should have a zero margin of error.

If tabulation errors were to occur or should the application not work as intended, these problems would be identified by independent auditors and the counting process can simply be restarted.

Resource Efficiency

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

Table 2 compares the cost per ballot cast in person in townhouses with internet voting in the 2017 Local Elections in Estonia. A ballot cast over the internet costs 42% of the cost of a ballot cast in a county centre (Krimmer, Duenas-Cid, & Krivonosova, 2020).

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

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

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

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. Only a small number of employees would be needed to oversee the system and ensure its correct operation.

It is estimated that the use of internet voting in the 2011 Estonian parliamentary elections saved 1500 man-days (Tsahkna, 2013).

Catering towards Special Voters

Voters that are not able to cast their votes because they cannot get to a polling station can cast their votes 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 without compromising their anonymities. The system does not require any alterations to allow these voters to cast their votes.

Should people not have access to the technology needed to cast their ballots, local public buildings can be equipped with devices running the application that allow 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 do so at a 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).

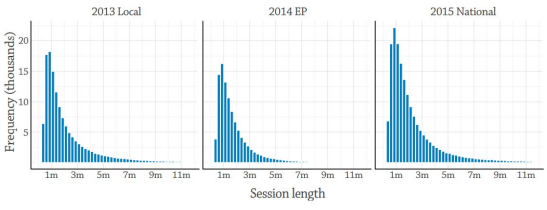

Figure 1 - Session length of casting a vote using the Estonian voting portal (Solvak & Vassil, 2016)

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 in the 2015 National Election 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

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 into the system (Wikipedia, 2022).

Such implementations can also provide voters with a confirmation that the vote they cast was recorded and counted correctly.

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 ensuring the security of the system and integrity of elections 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 elections being run using it (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 any or all elements of the application be compromised, auditors and election officials are aware of the compromise and action can be taken to mitigate it (Halderman, 2020).

Applications should provide threat models for all layers, discussing how issues such as Denial of Service (DoS) are mitigated.

Verifiability and Auditability

Internet voting has an analogous relationship with end-to-end verifiability, a concept first proposed by David Chaum in the early 2000s (Carback, Internet Voting, 2022).

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

Such a system was first implemented in 2009 in Maryland; Scantegrity II 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).

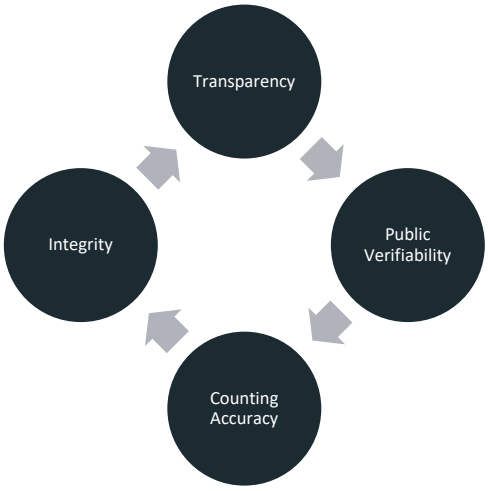

Figure 2 - Security properties of e2e systems

E2E implementations are not limited to internet voting. 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 abstracted 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 & Willemson, 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.

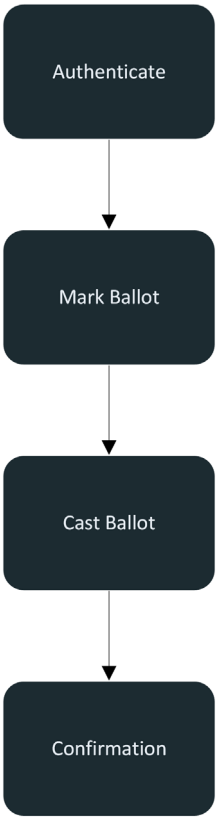

Figure 3 - Ballot casting workflow

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

After the voter successfully authenticated, they mark their ballot with their preference and then cast it. 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 and having the ability to verify the vote thereafter in 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 is shared with the election management body, which is to review it integrally, and organise mock elections to prove its functionality (Solvak & Vassil, 2016).

Voter Coercion Mitigation

Many internet voting applications allow voters to cast multiple votes, where the last valid vote gets tabulated and all others are spoilt. 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 (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, which allows voters to re-cast votes, showed that 98% of voters cast a single vote using the system (Solvak & Vassil, 2016), the paper states that consequently 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 very different 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 the secret ballot (Park, Specter, Narula, & Rivest, 2021).

But it is not that simple. Figure 4 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).

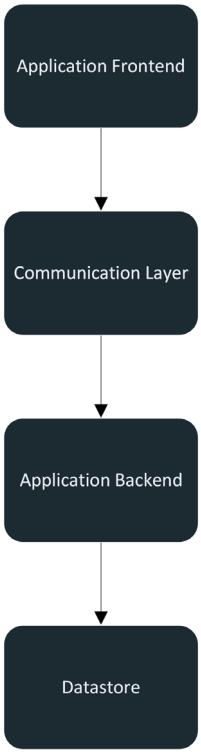

Figure 4 - Elements of an internet voting system

Voters use insecure hardware to cast their vote; the phones and computers used to cast votes can be infected with various types of malwares that can expose the voter's anonymity and/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 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 may offer a heuristic to the latter

challenge (Solvak & Vassil, 2016), the voter's choices could still be undetectably monitored by a third party, hence compromising their anonymity.

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 and number of parties interested in destabilising an election are great, and it can be assumed that malicious actors, such as foreign governments, have nearly unlimited resources available to them (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).

Internet Voting Implementations

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 internet voting implementation in Estonia as well as discussing some other internet voting applications developed recently.

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 as well as all elections thereafter (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

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

candidates representing a party. The seats are distributed to parties in respect to the total number of votes that candidates received in said party (Solvak & Vassil, 2016).

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 first identifies themself to the system, after which the system checks whether the voter is eligible, according to their age and citizenship, to vote in the election. If so, the i-voting system displays the list of candidates in the voter's district that can be voted for.

Voters can then browse the list of candidates and decide whom to vote 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 e-voting app encrypts the vote using the 4-digit pin provided. After this, the voter provides their 6-digit pin as a digital signature to confirm their choice. 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 Election Day, the encrypted votes and the digital signatures are separated. This system is similar to two-envelope postal voting in that regard[10].

After the voting process is complete, a QR-code is displayed in the voting application. Using a smartphone with a QR-code reader, voters can use a vote verification app to verify that their vote was recorded as intended.

To ensure that the voter is expressing their true will, the system allows for them to change their electronic vote by casting another ballot digitally or by voting at a polling station during the electoral period (Solvak & Vassil, 2016). In this manner, should a voter be coerced into casting a vote, or should their vote be recorded incorrectly, they can cast another vote to replace it.

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 (Solvak & Vassil, 2016).

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

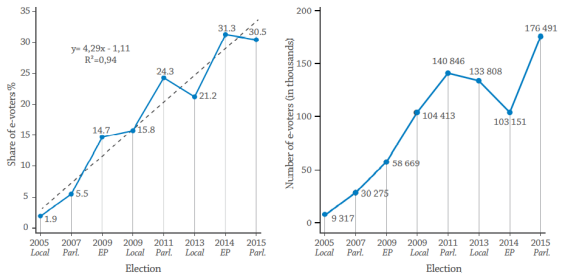

Figure 5 - Percentage and count of votes cast using the internet voting application in Estonia over time (Solvak & Vassil, 2016)

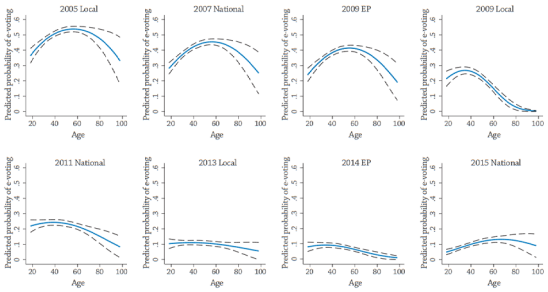

Figure 6 - Effect of age on the probabolity of e-voting for the first time over elections (Solvak & Vassil, 2016)

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, and 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)

As seen in Figure 6, there is a loss of relationship between age and the probability of casting a vote online.

Technological diffusion has traversed social boundaries in Estonia 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.

The study also identified a relationship between the distance to a polling station of a voter and their likelihood to cast a digital vote. It can be deduced from figure 7 that the distance of a voter to the nearest polling station increases the probability that they will choose to cast an e-vote.

Over 30% of the population cast their vote online in the 2015 parliamentary elections (Solvak & Vassil, 2016).

© Andrei Florian 2022

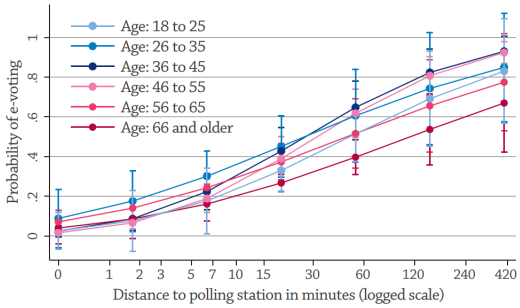

Figure 7 - Relationship between the distance to a polling station and probability of e-voting in Estonia for different age groups (Solvak & Vassil, 2016)

Voter Turnout and Internet Voting

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

An Estonian paper refers to this 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 concept 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 security audit revealed that the system relied heavily on misplaced assumptions about the security of voter hardware and system hardware and software (Springall, et al., 2014).

The members of the audit were invited at the vote counting event after the election where they observed the proceedings. Multiple protocol breaches were identified during the process: the "airtight" computer that was used to sign the election showed clear signs of having been connected to the internet, software was downloaded in an insecure manner, and the USB stick containing all the casted ballots was repetitively rejected by the system leading to the staff needing to use a personal USB stick to transfer the electronic votes (MacAlpine, 2022).

The report recommended that the Estonian authorities to cease using the internet voting system (Springall, et al., 2014).

Conclusion

Estonia was the first country in the world to allow voters to cast their ballots over the internet in an official, country-wide election. The insight gained from the project proves invaluable to understanding the relationship between voters and internet voting.

© Andrei Florian 2022

As discussed, many problems were identified with the system and the Estonian Government guaranteed the improving of the system based on feedback from auditors (Springall, et al., 2014).

Today, Estonia still provides internet voting as an option in all elections.

Internet Voting Application

A small community of experts are working on several internet voting applications. In this section, I will highlight three internet voting applications and provide a summary of their records.

Voatz

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 of electoral integrity.

The Voatz team chose to remain secretive and disallow security audits of the system. This resulted in an internet voting application with no security guarantees that should never be used in official elections.

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).

Helios

Helios 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[11], and hosts a running version of the application that can be used to organise small elections. Many elections have been hosted using the

[11] https://github.com/benadida/helios-server

platform to this day and millions of votes were cast (Benaloh, Internet Voting, 2022).

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 verify that their ballot was recorded successfully and allows the tabulation process to be verified (Adida, 2017). Helios is designed for first-past-the-post[12] type elections.

Although Helios achieves end-to-end verifiability, it is not recommended to be used in high-stakes elections as it relies on the digital marking of ballots.

Remotegrity

Remotegrity is an internet voting extension to Scantegrity II[13] – the first e2e-v system used in an official election.

The application uses a hybrid approach consisting of the provision of ballots via mail and the return of ballots digitally over the internet. The application uses an implementation of code voting[14] where the voter receives a paper ballot via mail containing codes representing each candidate they can vote for (Zagórski, et al., 2013).

The voter then inputs the code of the candidate they wish to vote for into the digital application and can then verify that the vote was recorded as intended.

The application is end-to-end verifiable and allows for the independent auditing of ballots. This hybrid approach to internet voting circumvents vulnerabilities in voter hardware as an attacker is not able to mutate a voter's ballot in a targeted way or compromise their anonymity as they do not know which candidates the ciphertexts inputted correlate to (Zagórski, et al., 2013).

Remotegrity overcame most of the challenges associated with internet voting and appears to be suitable for use in high-stakes elections.

[12] https://en.wikipedia.org/wiki/First-past-the-post_voting
[13] http://scantegrity.com/
[14] Surevote: Technical Overview by David Chaum.

ethlect. Abstract

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

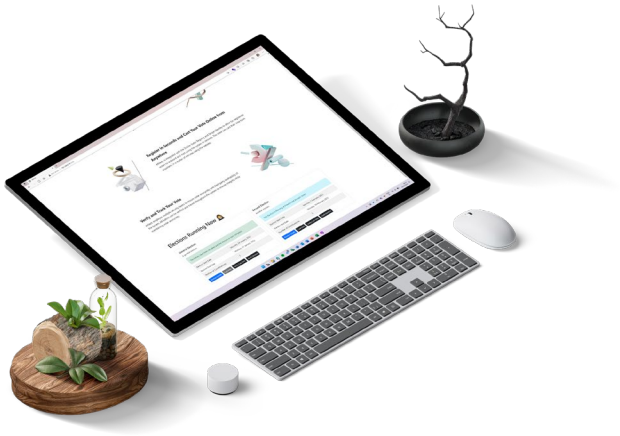

Figure 8 - ethlect. application homepage.

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.

System Overview

ethlect. is an end-to-end verifiable, internet voting system that employs a hybrid approach to remote voting and various cryptographic techniques guaranteeing complete transparency and auditability while ensuring the secret ballot.

The system employs code voting for the marking of ballots: the electoral authority verifiably generates and distributes physical ballots via mail to voters. Ballots consist of the candidates a voter can vote for and a three-digit numeric representing each candidate.

Upon the receipt of a ballot, a voter can log into the ethlect. application on their personal computing device and cast their ballot digitally by inputting the unique ID of the ballot and the codes representing each candidate they wish to vote for in order of preference.

The implementing of code voting circumvents the unavoidable security vulnerabilities in voter hardware such that an attacker cannot modify a ballot with intent nor compromise voter anonymity by monitoring voter selections.

The application allows voters to verify that their vote was recorded as intended without compromising their anonymity. Once a vote is recorded in the system, a series of verifiable cryptographic processes and shuffles anonymise ballots while allowing for the independent auditing of the election to ensure its integrity at all stages.

ethlect. is designed specifically for the Irish Electoral System running on PR-STV but can be easily adjusted to fit other systems such as first-past-the-post.

The application's integration with the Dublin Voter Registry[15] and the Stripe Identity[16] service allows for the digital

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

[16] https://stripe.com/ie/identity

onboarding of registered voters in seconds.

Architecture Overview

Figure 9 illustrates the system's architecture in abstract[17]. Voters that are already registered to vote on the registry can follow the online registration process to use the app to cast their ballot.

A registered voter is required to provide their full name and address allowing the application to check if they are on the voter registry. If the voter is found on the registry, they are asked to create an account on the application by inputting an email, passcode, and generating a 2FA token.

In order to complete the registration process, voters must verify their identity via Stripe Identity by providing an image of their passport and a video selfie.

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 election keys. Ballots are generated for all voters in a verifiable manner, the ballots are then verifiably decrypted and printed at a secure location and posted to the voters.

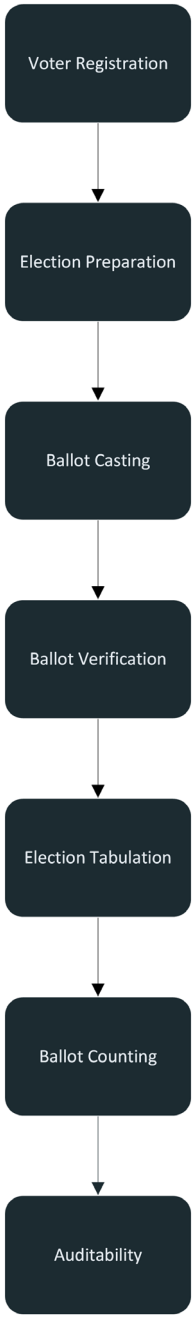

Figure 9 - ethlect. abstract system architecture

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 codes representing each candidate into the application. The system will ensure that the ballot was marked correctly and

[17] See Application Workflows section for more details

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 digital ballot box themselves. The voter must have access to their physical ballot to verify their vote; this process is designed such that only the voter can confirm the correct recording of their vote preventing third parties from identifying the voter's choices and compromising their anonymity.

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, the election's keyholders that hold decryption keys are to combine their keys and decrypt the ballots verifiably. The decrypted ballots are made public and can then be counted by any interested party.

All stages of the tabulation process are made public together with cryptographic proofs that attest the correctness of each stage. Independent third parties can hence verify each stage of the application and quantifiably confirm the election's integrity. Should the application misbehave, or should an attacker compromise any element of the system, auditors are able to identify these issues which can then be mitigated.

Objectives of ethlect.

ethlect. aims to accomplish eight key concepts:

End-To-End Verifiability

ethlect. complies with the criteria of an e2e verifiable system[18]. Figure 10 illustrates the process that a ballot takes from its creation to its decryption. Each stage of the process is completed in a verifiable way that allows independent third parties to certify the correctness of the process.

ethlect. generates and encrypts ballots before they are cast; this process takes place when the election is prepared. Two ballots are generated per registered voter. Each ballot is encrypted, and the encrypted candidates are shuffled before assigning the ballot to a specific voter. The application generates a zero-knowledge, non-interactive proof of encryption to accompany the ballot generation process that can be verified by third parties. This allows auditors to ensure that ballots were generated correctly.

[18] Outlined in Requirements of Internet Voting Section

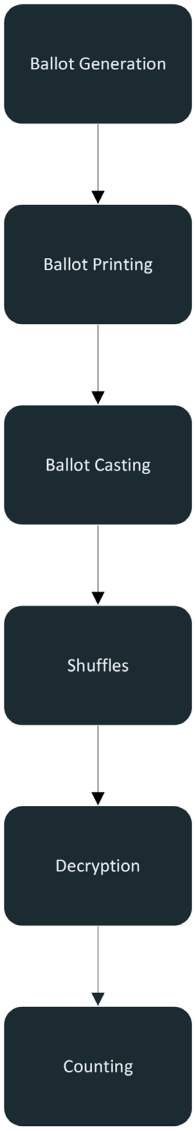

Figure 10 - Ballot handling layers in ethlect.

The ballot printing process involves the decryption and 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 code on a ballot represents. The ballots are printed, the codes representing each candidate are concealed using tamper-proof scratch off ink[19], and the ballots are posted to the voters. This process is verifiable through the introduction of auditable votes in the system and subsequent verification of these ballots by auditors during the printing process.

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 provides their selection of candidates in preferential order using the codes on the ballot.

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.

The application mitigates the issue of voter coercion by providing voters with the option to cast a new ballot which replaces their old one should they believe their ballot was recorded incorrectly or should they have been coerced into casting it.

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 of an operation contains the same ballots as the inputted set, which

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

ultimately verifies the integrity of individual ballots.

The electoral period is terminated, and the ballots are anonymised through shuffling and finally decrypted. All cryptographic shuffles produce a 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.

The application also publicly provides the decrypted ballots so that any interested party can independently count the ballots and identify the winning candidates.

Deterministic Surplus Redistribution

As mentioned in the analysis of the Irish Electoral System section of the paper, the redistribution of surplus ballots from a winning candidate to remaining candidates is conducted in a non-deterministic manner, relying on the order the ballots were counted in.

The Irish Electoral System does not have the capability to formularise the redistribution of ballots due to its lack of reliance on technologies that can assist with the counting process; it would prove very challenging to devise a strategy to redistribute ballots deterministically for PR-STV using human counters (Martin, 2022).

ethlect. allows for the deterministic redistribution of ballots based on the frequency candidates appear as the next choice on the winning candidate's ballots.

To better explain the redistribution of ballots, take for instance a situation where four candidates – John, Aoife, Sarah, and Leah – are competing for two seats in a constituency. Let's assume that the quota for the constituency is 3 votes.

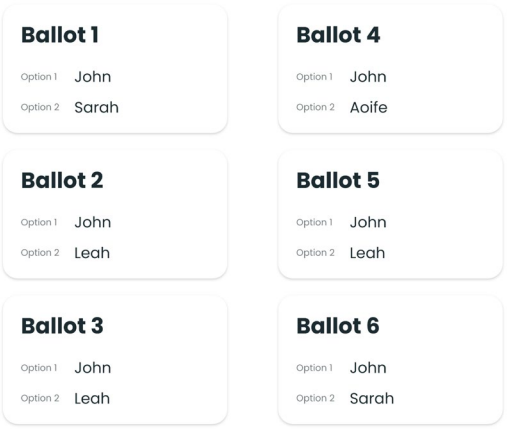

Figure 11 - Ballots with John as first preference

Figure 11 illustrates all six ballots casted in the election where John was selected as the first preference. In this situation, we need to redistribute 3 of the 6 votes to other candidates remaining in the contest.

The Irish PR-STV implementation would choose all ballots counted after the third and redistribute them. Using this system,

Aoife, Sarah, and Leah would receive one ballot each.

This is unfair for Leah who was marked as a second preference 3 of the 6 times yet receives the same number of redistributed ballots as Sarah and Leah.

ethlect. approaches this problem by sorting the winner's ballot set as shown in figure 12.

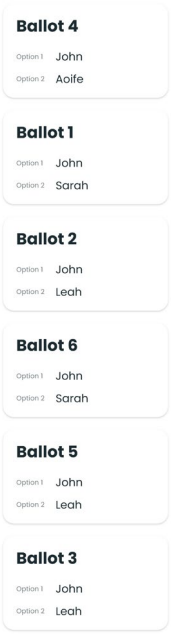

Figure 12 - ethlect. sorted ballot set

By looping through each next choice, effectively orderly mixing the ballots, the candidates who appear more frequently as the next preference will be more concentrated at the bottom of the pile than candidates that appear less frequently.

The application will then select n number of ballots from the bottom of the pile and redistribute them (where n is the number of surplus ballots, 3 in this case).

With the ethlect. system, Leah would receive 2 of the redistributed ballots and Sarah would receive one which is a fair representation of the frequency each respective candidate appeared as a next choice.

Frontend Security

As mentioned before, the application circumvents the inevitable lack of security in the hardware voters use to cast their votes through the use of physical mailed ballots and code voting. This approach is similar to that used in Remotegrity (Zagórski, et al., 2013).

Consensus in the electoral design community is that internet voting solutions that only involve digital elements do not offer the security guarantees required for use in high stakes elections (National Academies of Sciences, Engineering, and Medicine, 2018).

In pure internet voting applications, the voters input their candidate selection into the application directly. In this sense, a voter exposes both their identity (through logging in) and ballot (through selecting the candidates they wish to vote for) to the application's frontend. Hence should a bad actor have access to the voter's device, they can compromise the voter's anonymity by monitoring their selections

and/or intently mutate their ballot without their knowledge.

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 as well as 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 to be carried out as this would require having access to a large number of physical ballots as well as digital credentials.

In a sense, the security guarantee of the frontend is analogous 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[20].

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 security assumptions about the correct operation of any element (hardware or software), 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 integrated with identification tools to transition a registered voter to the application in minutes.

The application is integrated with the Dublin Voter Registry 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 to be. The Stripe's Identity 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

[20] See section on security analysis for more.

motion to prove that the person is real). If all security checks pass, the application will allow the voter to cast their vote.

Integration with The Electoral System

Figure 13 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 at the start (Solvak & Vassil, 2016). Elections held digitally using ethlect. should allow for an election window of around a week where voters choosing to vote online can do so prior to Election Day in person (Solvak & Vassil, 2016), this is to ensure voter convenience, reduce the impact of potential DoS (Denial of Service) attacks and allow for the recasting of votes should a voter deem their vote to be mutated.

At the end of the election window, the application will allow for all ballots cast using the system to be shuffled and decrypted. The decrypted ballots contain a list of candidates in preferential order.

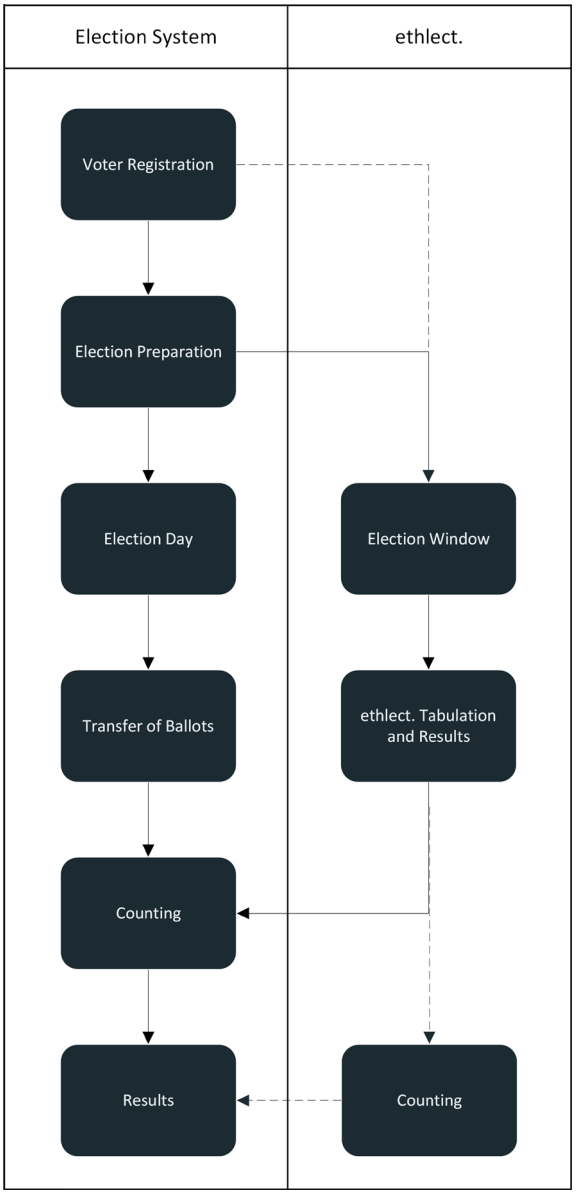

Figure 13 - Diagram of integration of ethlect. with the electoral system

The electoral authority has two options to count the votes cast online: the ballots can be counted digitally using a provided application or they can be printed and mixed with ballots cast traditionally and counted together.

A digital count offers many security benefits and ensures the end-to-end verifiability of an election, but it is

important to note that to count ballots in a PR-STV type system, all ballots must be counted collectively.

If the ballots are to be counted digitally, the election results can be downloaded from the ethlect. application by any party and automatically counted using the votecounter application (or manually if desired).

If it is decided that the ballots cast digitally are to be counted with the physical ballots, they can be embedded on a memory element (memory stick, etc.) and then 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.

A solution to counting all ballots digitally would be to digitise paper ballots cast and add them to the list of ballots cast over the internet. The votecounter application can then be used to count the ballots autonomously and identify winning candidates.

Although ethlect. is designed to be used with Ireland's specific PR-STV system, it can be easily adapted to other systems such as first-past-the-post as implemented in many elections in America while maintaining the same security guarantees and operation.

Open Source, 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 ballot processes and accompanying proofs from the application website.

The source code of the ethlect. application is made public together with the source code of other accompanying applications used for processes such as vote counting or election auditing[21]. All cryptosystems used in the project are publicly documented allowing independent auditors to develop their own auditing solutions.

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 nurture trust in voters and the electoral authority (Herrnson, 2022).

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

[21] See Links Section in Abstract

In this sense, the application uses plenty of user feedback, beautiful UI packs, and thought-out UX strategy to make interacting 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 8 for an outline of the workflows.

Voter Registration

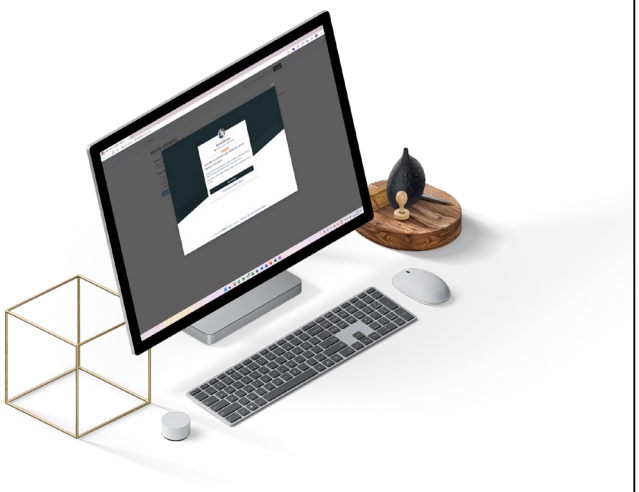

Figure 14 - ethlect. voter registration process

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.

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

Registration Process

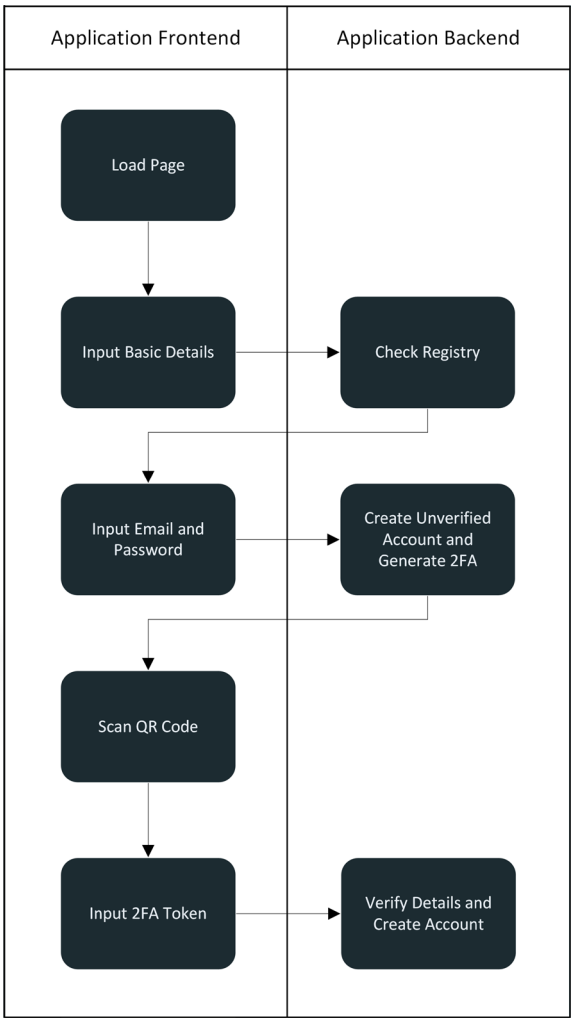

Figure 15 - Voter registration process diagram

Figure 15 illustrates the voter registration process. From here on out, the frontend will reference the web application served

[22] Security implications explained in security analysis

to the user and the backend will reference the servers of the application.

A user can register to use the application by heading to `/register`. They will see a form requiring them to input their first name, last name and Eircode[23], as seen in figure 16.

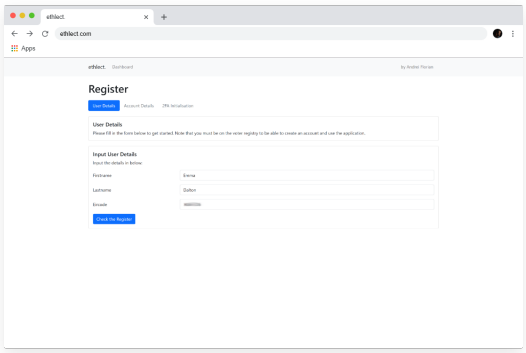

Figure 16 - Voter provides general details

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/${firstName}/surna
me/${lastName}/eircode/${eircode}/lang/en
```

This is the public API that the Dublin 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. Note that the election admin can disable the voter register connection via the app's settings.

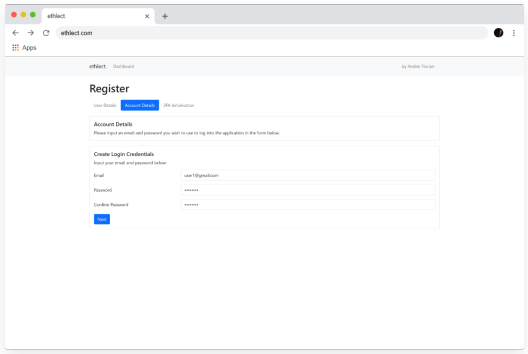

Figure 17 - User provides email and password

If the application receives a positive match, it will move to the next tab in the form. This can be seen in figure 17. The voter is asked to input the 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

[23] https://www.eircode.ie/

will set the `{accountVerified: false}` property on the user document.

If this process was successful, the form will progress to the last page where the voter is asked to scan the provided 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 18.

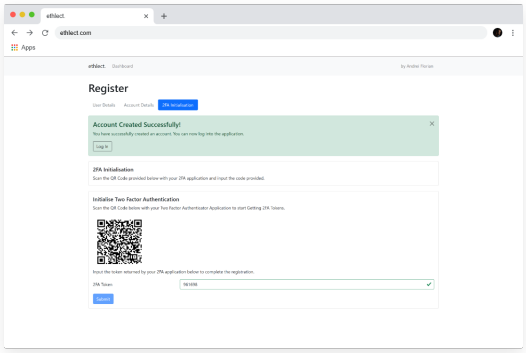

Figure 18 - User 2FA initialisation

The token submitted will be sent to `/api/register/validate`. This route will search for the user that submitted the token in the database and they are found, it will check if the provided token is correct. If the token is correct, the application will set the following property on the user account `{accountVerified: false}` completing the registration process.

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

Voter Login

The application uses the Next Auth[24] 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).

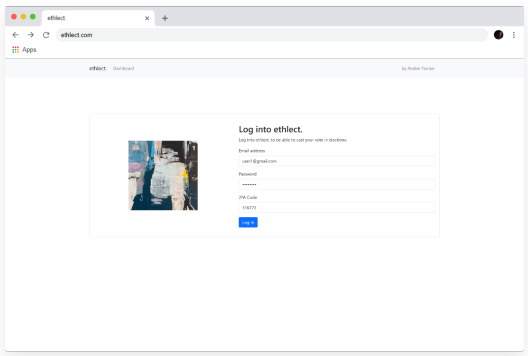

Figure 19 - Login page

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[25] 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

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

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

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

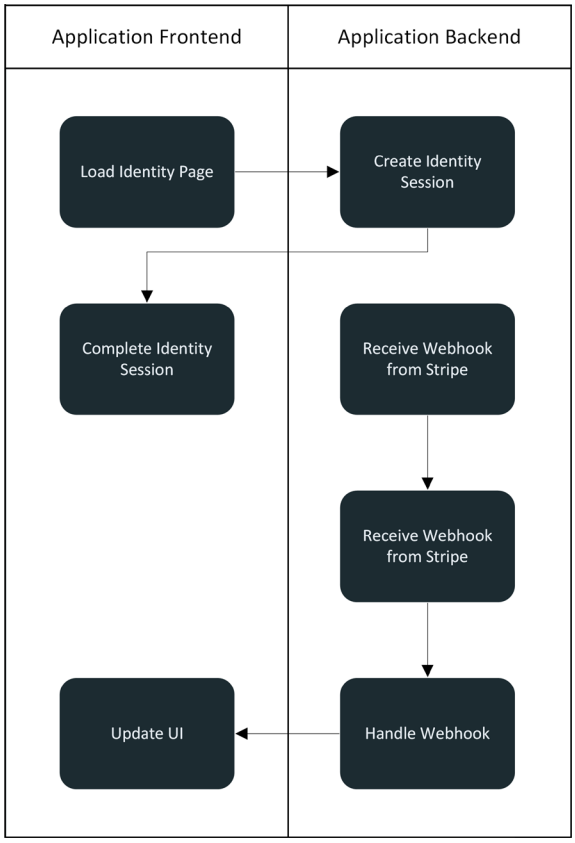

Figure 20 - Identity verification process diagram

Figure 20 shows the process for verifying the voter's identity. After logging in successfully the first time, the voter will be alerted to verify their ID to be able to cast votes.

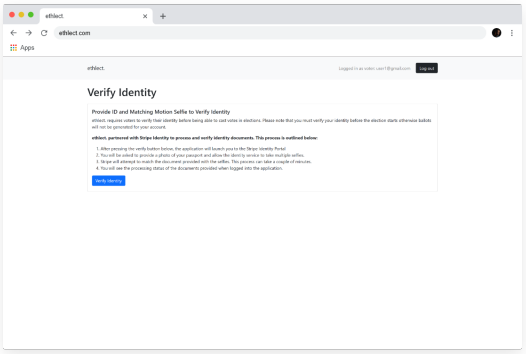

Figure 21 - Identity verification page

By following the link in the alert, the voter is directed to `/verifyID`. From this page, as seen in figure 19, 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.

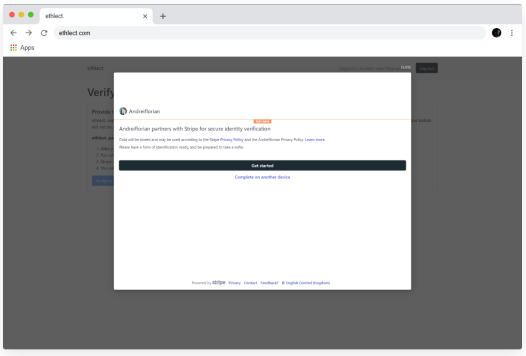

Figure 22 - Stripe verification session pop-up

The API will resolve a client key that the frontend will use to initiate a launch the session as shown in figure 22.

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 as shown in figure 23.

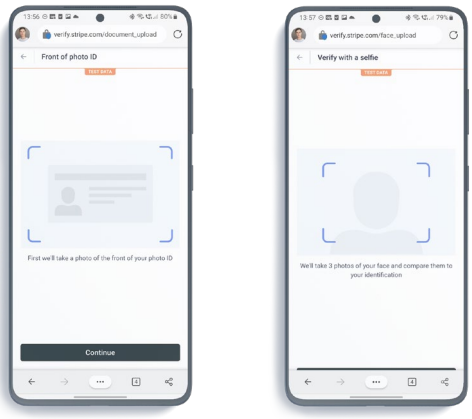

Figure 23 - Stripe Idenity process

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 the following 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 in this case extract the first and second names 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'}`.

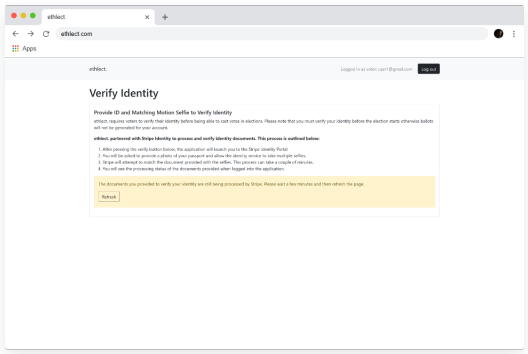

Figure 24 - Stripe Identity processing documents

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 as seen in figure 24. 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.

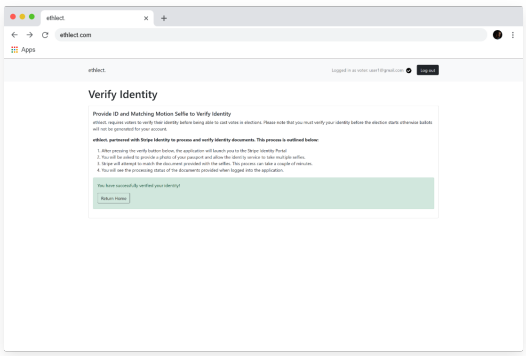

Figure 25 - Voter verified identity successfully

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 Ireland's case a subset of the Dept. of Housing. The implementation is designed to have multiple keyholders. A keyholder is a person 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. This threshold system allows for the reconstruction of the original key using a fraction of the generated shares. 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, heads of parties, etc.).

Election Creation Process

The election creation process is completed digitally using the application and is summarised in figure 26. 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). This can be seen in figure 27.

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.

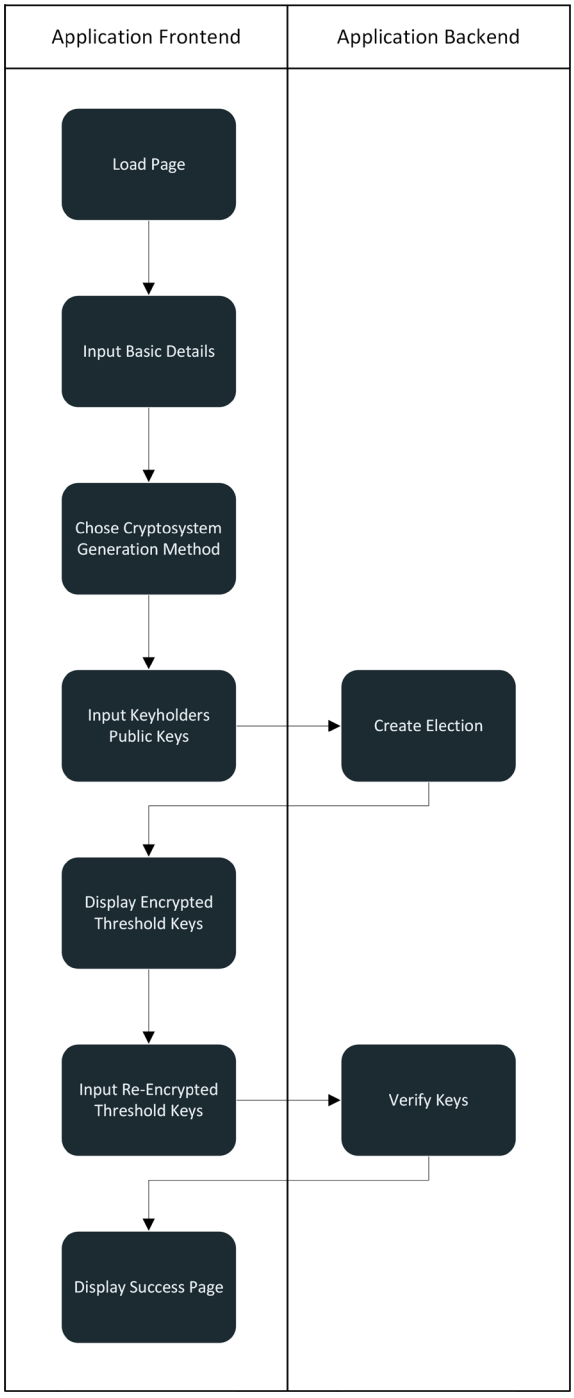

Figure 26 - Election creation process diagram

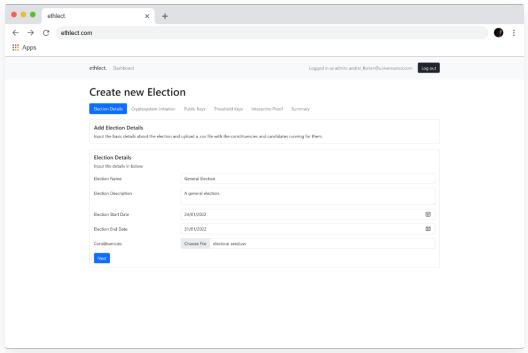

Figure 27 - New election form first page

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

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

Table 3 - csv file format example

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[26] 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.

The application will then ask the admin to ask all keyholders to generate an RSA keypair and provide the public key to the admin.

[26] https://www.npmjs.com/package/basic_simple_elgamal

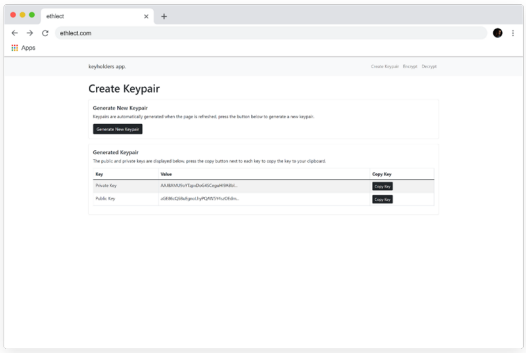

Figure 28 - keyholders app generating an RSA keypair

ethlect. includes the keyholder app which is an application that allows keyholders to manage their RSA keys. A keyholder can locally run this application and visit `/create` to receive an RSA keypair as shown in figure 28. The keyholders can copy the keypair through the app and send their public keys to the admin. The admin will then input these keys into the form as shown in figure 29.

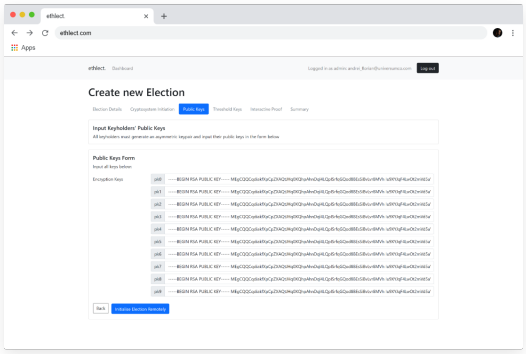

Figure 29 - Keyholders public keys (note that the same key was used here for all keyholders for simplicity)

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 keys 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 $SHA384(x)$ (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 running for said constituency are listed.
6. The election will be marked as not verified by setting the following property: `{electionVerified: false}`

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 as shown in figure 30.

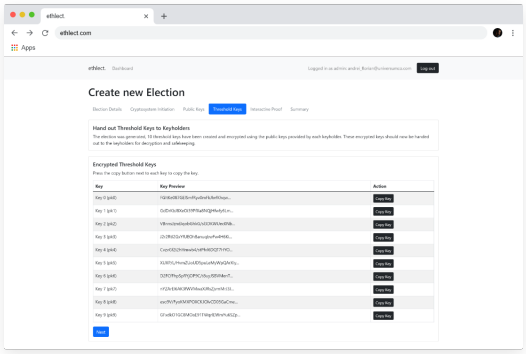

Figure 30 - Key distribution

The keyholders should use their generated private key (linked to the public key provided) to decrypt the keys and store them securely. This can be done using the keyholders app.

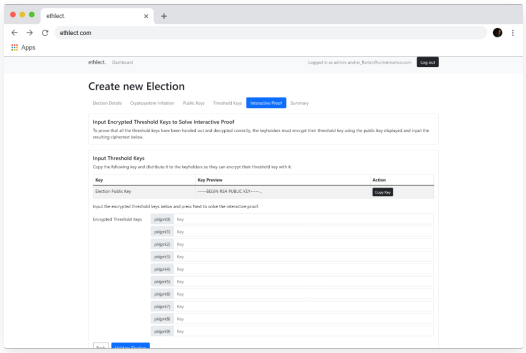

Figure 31 - Input threshold keys encrypted with E_{pk}

The admin can press next to reveal the verification tab as seen in figure 31. 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.

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}`.

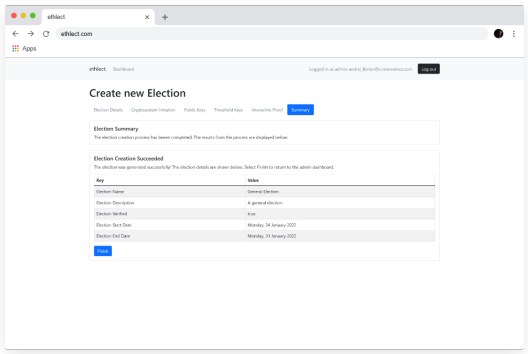

Figure 32 - Election creation results table

The admin will receive a success or error message as a result as shown in figure 32. The verification process is added to ensure that the admin successfully distributed the keys to the keyholders, and that the keyholders managed to decrypt them correctly. 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.

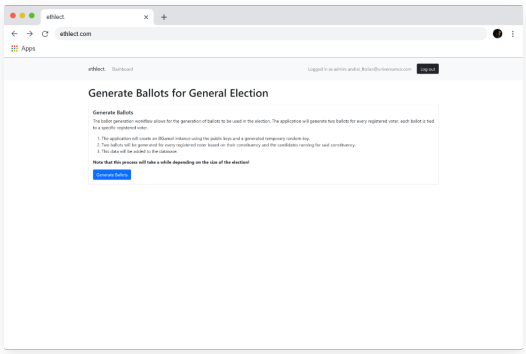

Figure 33 - Ballot generation page

The admin can press the button on the page to start the process as shown in figure 33. This process can take a while depending on the number of ballots that need to be generated.

Figure 34 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 $V_{C_i} * 2$ where V is the number of voters registered for constituency C_i. Note that more ballots can be generated later should the need arise.

n number of candidate IDs ID_n (6-digit, random numbers) are generated for each candidate c of every constituency C. 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.

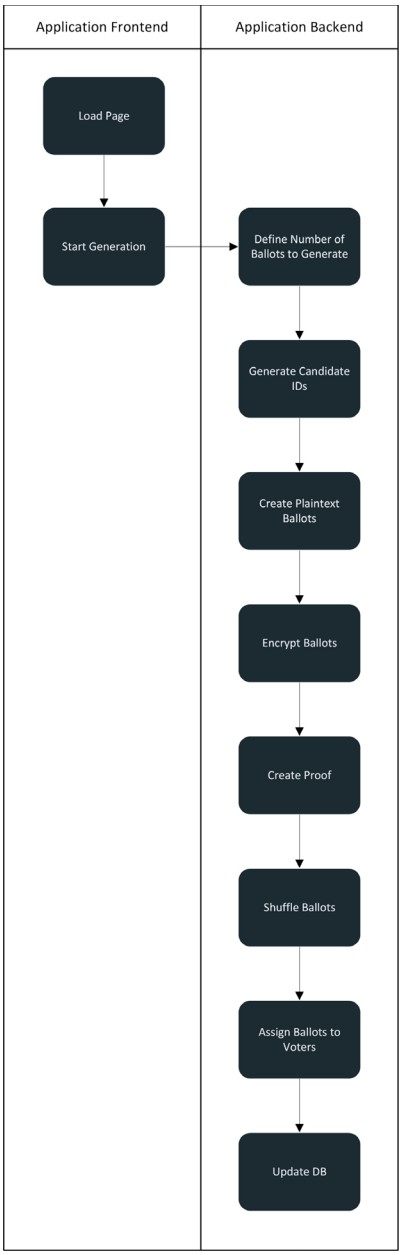

Figure 34 - Ballot generation process diagram

The result should be n ballots, each containing different candidate IDs representing 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. The same is the case for candidate 2.

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

A cryptographic proof is created to attest the correctness of the encryption and then the encrypted candidates within each ballot set are shuffled. 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 repermuted randomly. This is key to ensuring the secret ballot.

For voter convenience, a random 3-digit number is generated for every encrypted candidate ID for every ballot (each PIN is unique to a 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 publicly 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 entry in every constituency.
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 PIN 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.

The application generates ballots such that no layer of the digital application, nor an attacker without access to the election keys, can identify which candidate each 3-digit PIN corresponds with.

This security guarantee is essential to uphold the system's end-to-end verifiability.

There does however need to be one layer of the system that maps candidates to the PINs on each ballot (otherwise voters would not know which PIN corresponds to which candidate).

ethlect. is designed such that the PINs are mapped to candidates solely on the physical ballots used by voters to cast their vote.

The printing process involves the decryption of all ballots to reveal the relationship between the PINs and candidates. It is essential that this process is done at a safe location using airtight computers as exposing this mapping would compromise voter anonymity.

The following process provides an overview of the printing process:

1. The ballots generated by the application are 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 then use to encrypt their threshold keys and input them back into the system.
3. This computer then decrypts and combines the inputted keys and uses the resulting private key to decrypt all candidate IDs of all ballots, this exposes the relationship between the PINs and candidates.
4. The computer should then print these ballots. Every printed ballot consists of a 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 PIN that is to be introduced into the system when casting a vote. It is recommended that this ID is covered with scratch off ink.

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

voters. Figure 35 illustrates how a ballot would look like.

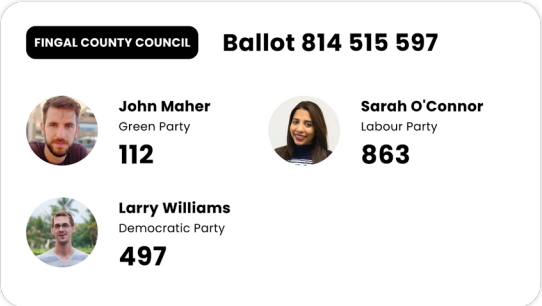

Figure 35 - Example ballot design

The application needs to prove that the ballot printing process was performed successfully. A simple cryptographic proof (such as proofs provided during the tabulation process) cannot be provided in this case because these proofs would expose the mapping between candidates and IDs to auditors.

Instead, ethlect. employs a novel auditing approach which involves the introduction and verification of auditable ballots by independent auditors.

During the printing process, auditors are invited to generate auditable ballots and add them to the ballot set before it is introduced into the decrypting system.

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 and the election's Elgamal public keys.

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 relates to.

It is recommended that auditors provide a commitment[27] to their ballot before introducing it into the decrypting computer such that they can prove to the election authority indisputably whether or not their audit succeeded.

The ballot set containing both the actual and auditable ballots is then inputted into the decrypting function. After all ballots are decrypted and 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 are printed correctly. This is the case as the ballot decrypting and printing computer cannot discern an auditable ballot from a typical ballot and hence cannot mutate ballots without the mutation ultimately being present in auditable ballots which would then be identified by auditors.

[27] https://en.wikipedia.org/wiki/Commitment_scheme

Simulating the Ballot Decryption

I developed an application that simulates the ballot decryption process autonomously without the need for auditors. It is important to note that this application is designed to simulate the actual process described above and is not a replacement for it.

The ballotprinter application (available alongside ethlect. in the repository) automatically generates auditable ballots, includes them in the ballot set, decrypts all ballots, and autonomously verifies the decrypted auditable ballots.

Whereas the official ballot printing process previously described relies on the trust of multiple auditors to accurately report the correctness of the process, the ballotprinter app assumes the security of the hardware and software elements in its system (which is why it is only meant to simulate and not replace the official process).

The application functions by asking its user to input a connection string for the ethlect. application's database. The election admin in this case should create a read-only account for the database and provide its connection string to the application (the app uses MongoDB).

The application will also ask for the ID of the election wished to print ballots for. This is shown in figure 36.

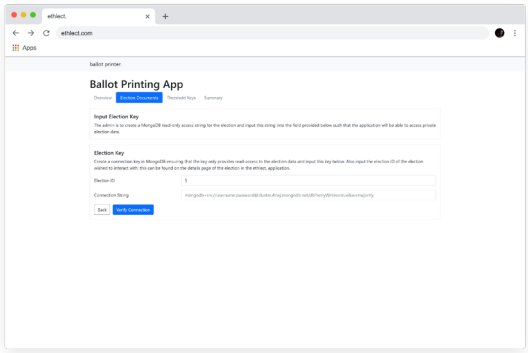

Figure 36 - Ballot printing app first page

Upon submitting the form, the application will send a request to `/api/verifyConnectionString` which will attempt to connect to the database using the provided string and extract the election's public key E_{pk}.

Should the process succeed, the application will provide E_{pk} and ask the admin to distribute the key to the keyholders to encrypt their threshold keys with it.

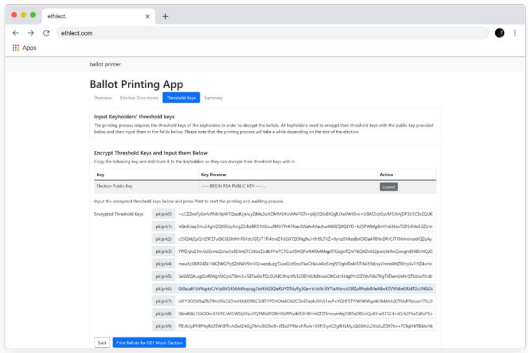

Figure 37 - Admin inputs encrypted threshold keys

Upon receipt of the encrypted threshold keys, the admin can input these keys into the form and submit it as shown in figure 37. The application will send a request to `/api/print` which will run through the

following process to verifiably print the ballots:

1. The API will ensure it can connect to ethlect.'s database successfully with the provided string again (in case of frontend data mutation).
2. The route will then attempt to decrypt and reconstruct the Elgamal private key and check its hash against the one stored in the database.
3. If this process succeeds, the API will send a request to the ballot generator module at `/api/helper/generateAuditableBallots`.
4. This route will generate n number of auditable ballots for every constituency in the same way as the ballot generation process described earlier (where n is defined by the admin, the more ballots the greater the security guarantee of the audit). The function will not generate any proof as it is not needed in this case.
5. Should the ballot generation process succeed, the API will respond with a hash of the plaintext set of auditable ballots and the encrypted auditable ballot set.
6. The print route will shuffle the auditable ballots with the other ballots and send them to the decryption module at `/api/helper/decryptBallots`.
7. This route will decrypt all ballots and return the decrypted ballot set back to the print route.
8. The print route will separate the auditable ballots from the rest and will send the decrypted auditable ballot set together with the hash provided by the ballot generation route to `/api/helper/verifyDecryption`.
9. This route will sort the decrypted auditable ballots, hash them, and compare the two hashes together. If the hashes match, the auditable ballots are proven to have been decrypted correctly.
10. The print route will then format the ballots, preparing them for printing, and return the decrypted ballots, together with the encrypted ballot set, encrypted auditable ballot set, and the hash provided by the ballot generation route to the frontend.
11. The admin will then be able to download the datasets provided as seen in figure 38, verify the decryption themselves, and then print the ballots.

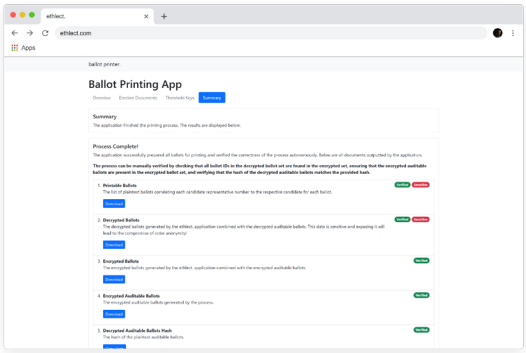

Figure 38 - Ballot printing app downloadable datasets

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 as shown in figure 39.

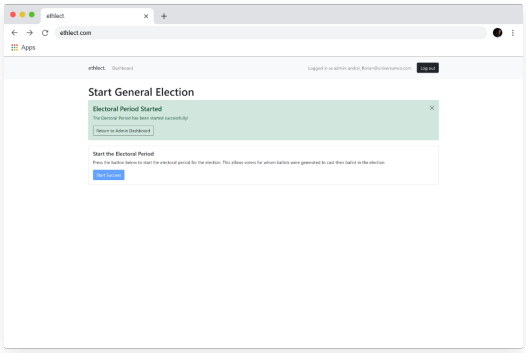

Figure 39 - Start electoral period

Casting a Ballot

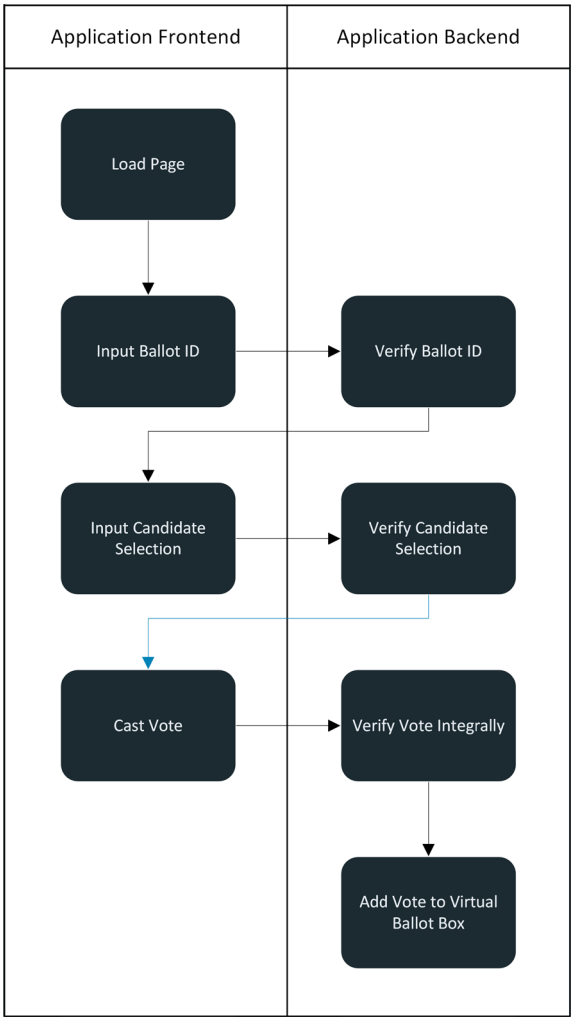

Figure 40 - Ballot casting process

Figure 40 illustrates the ballot casting process. Upon receipt of a ballot, the voter can scratch the ink off to reveal the 3-digit PINs, 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 as shown in figure 41.

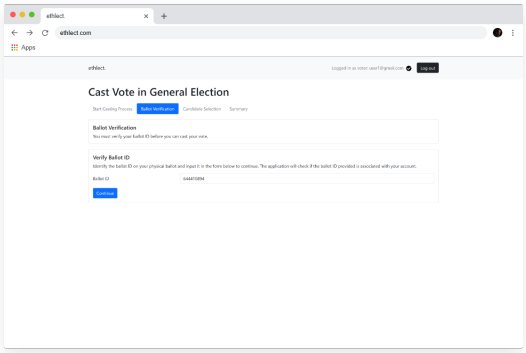

Figure 41 - Ballot ID verification

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 PINs of the candidates they wish to vote for in preference order as shown in figure 42.

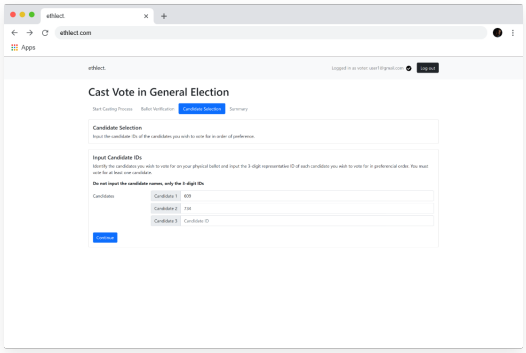

Figure 42 - Input of candidate selection

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. Otherwise, the voter is alerted that their inputted candidate IDs are not valid.

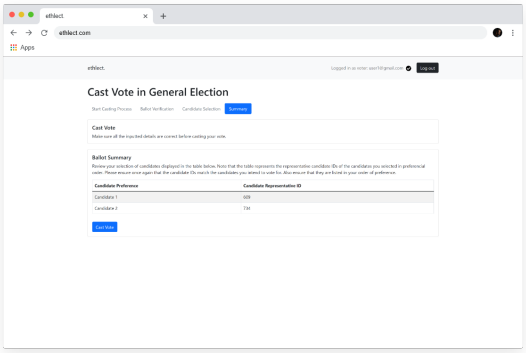

Figure 43 - Review and cast vote

If the operation was successful, a voter can review their selection and cast their vote by pressing the *Cast Vote* button as shown in figure 43.

This sends a request to `/api/vote/cast` which will run the verifications previously completed on the cast ballot again to prevent frontend data mutation.

If all verification steps check out, the app will ensure that the ballot was not cast already and then add it to the ballot box. This is 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 candidates voted for in order of preference. If the process is

successful, the voter will receive a success message as seen in figure 44.

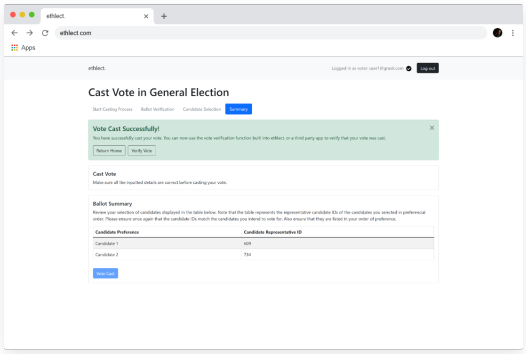

Figure 44 - Ballot casted successfully

Ballot Verification

A crucial element of an end-to-end verifiable system is allowing voters to verify that their ballots were recorded successfully by the application. This can be done directly, using the application, or independently by downloading the ballot box and finding the cast ballot in it.

ethlect. enables the verification that a ballot was cast correctly while persisting voter anonymity. A voter needs to have access to their physical ballot to be able to verify their vote. Hence an attacker would have to source the unique ballot the voter used to cast their vote to successfully compromise their anonymity.

It is crucial that voters securely dispose of their ballots after verifying their vote to prevent the compromise of anonymity.

Direct Verification

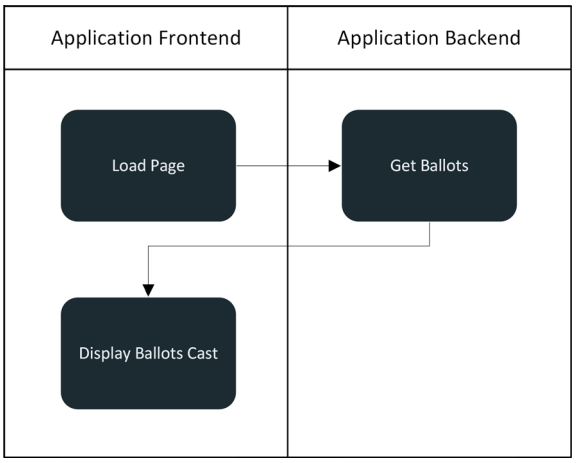

Figure 45 - Direct ballot verification process

The voter can access `/${electionID}/verify` to verify the ballot they cast using the application. This process is illustrated in figure 45. 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 as shown in figure 46.

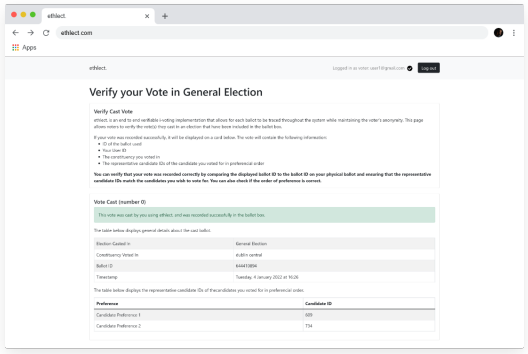

Figure 46 - Ballot verification page

The page will display the date and time 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[28] allows for all public election information 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 done by userID). The voter can then verify that the 3-digit IDs in the ballot object represent their intended vote.

Casting Another Ballot

As previously outlined, the application allows for voters to cast multiple ballots where the last ballot cast is the only one being tabulated. This system is designed to prevent voter coercion and mitigate the case of a vote being recorded incorrectly.

The application by default will generate two ballots per voter which will then be shipped to the voter.

The voter can cast the second ballot received to replace the first one at any point during the electoral period in the same way they cast their initial ballot.

The application will then spoil the first ballot cast and only include the second in the count.

Should the voter wish to cast another ballot, the electoral authority can facilitate the generation and posting of another ballot which can then be used by the voter.

Election Tabulation

The election tabulation process is performed 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`.

[28] see auditability in this section

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.

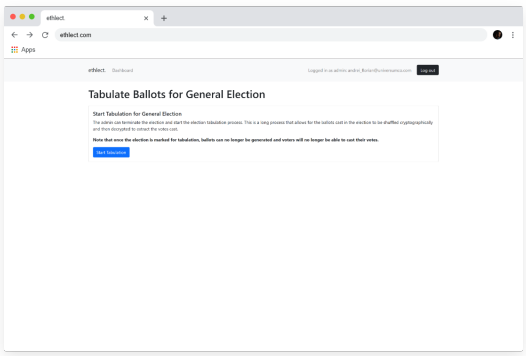

Figure 47 - Start tabulation process

By pressing the *Start Tabulation* button as shown in figure 47, the app will send a request to `/api/election/startTabulation`. This route 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 as illustrated in figure 48.

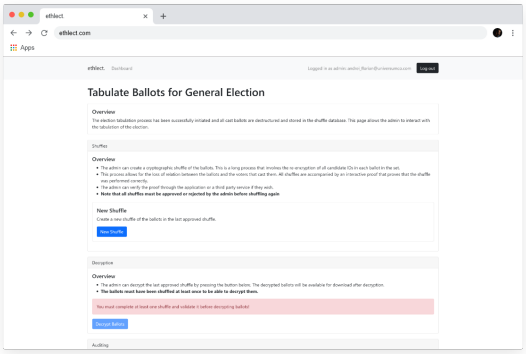

Figure 48 - Tabulation page

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.

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

Shuffle Ballots

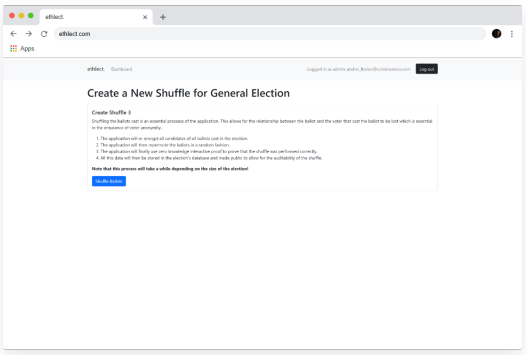

Figure 49 - Ballots shuffle page

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 as shown in figure 49.

Unless this is the first shuffle, the previous shuffle must be approved by the admin before they can create a new shuffle.

Figure 50 illustrates the abstract of this process. 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).

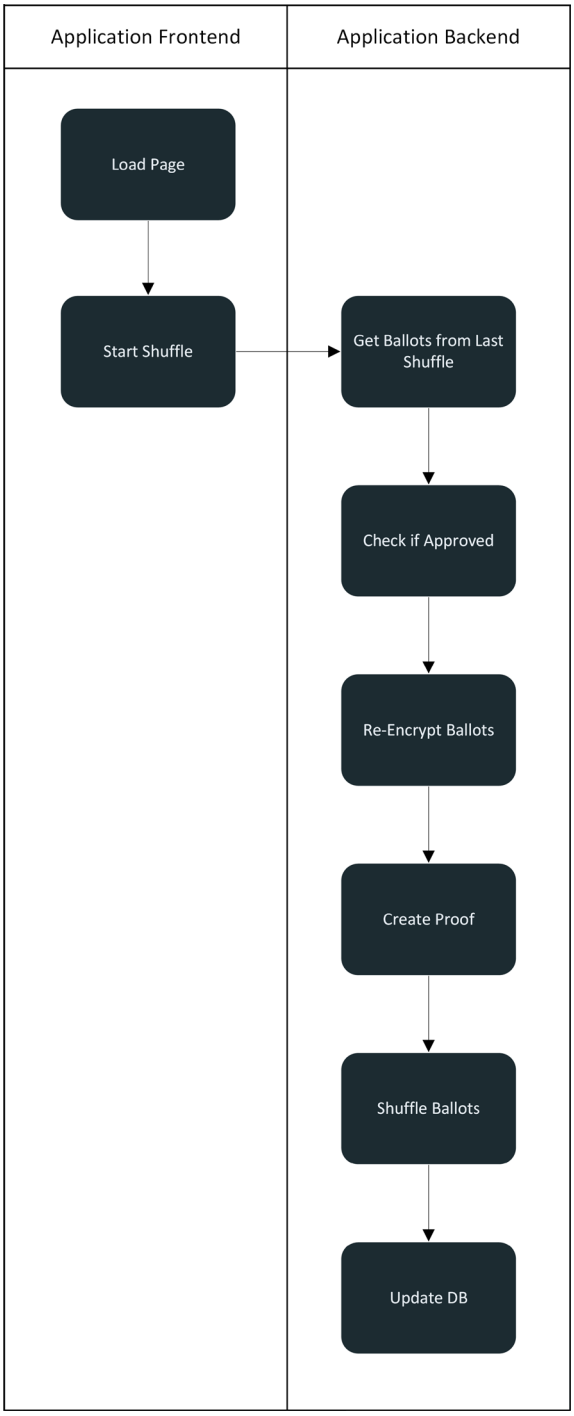

Figure 50 - Ballots shuffle process diagram

The resulting re-encrypted ballot set will be shuffled and then the shuffled set 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

shuffle ID of the previous entry 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 approve or reject the shuffle as shown in figure 51.

Ideally the shuffle would be audited independently before it is approved or rejected as explained in the auditability section.

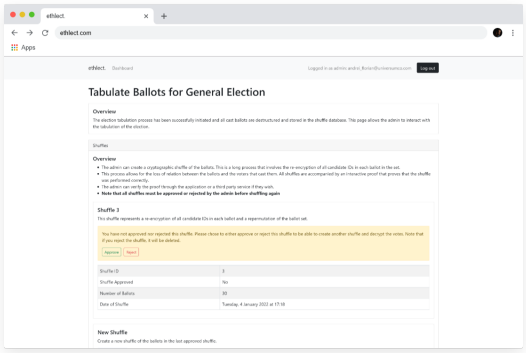

Figure 51 - Verify shuffle

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.

This feature is added such that should a shuffle be corrupted, this corruption would be identified through an audit of the shuffle after which the shuffle can be deleted, and a new shuffle commenced.

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 which will direct them to `/admin/${electionID}/tabulate/decrypt`.

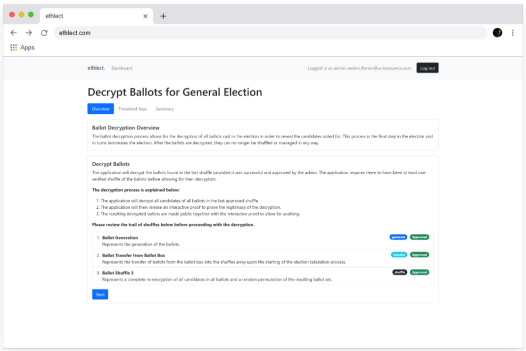

Figure 52 - Ballot decryption page

Figure 54 illustrates the decryption process. After loading the page, the admin will be presented with an overview of the processes that the ballots underwent until present as shown in figure 52. 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).

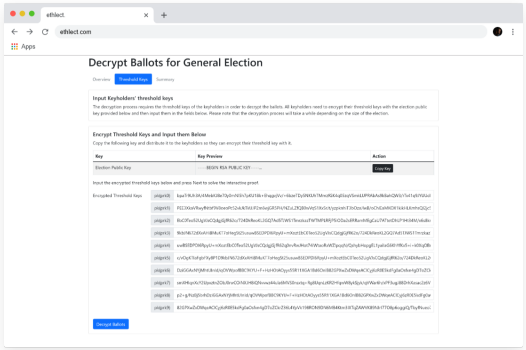

Figure 53 - Input encrypted threshold keys

The admin can input the keys as shown in figure 53 and finally press the *decrypt* button on the page which 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).

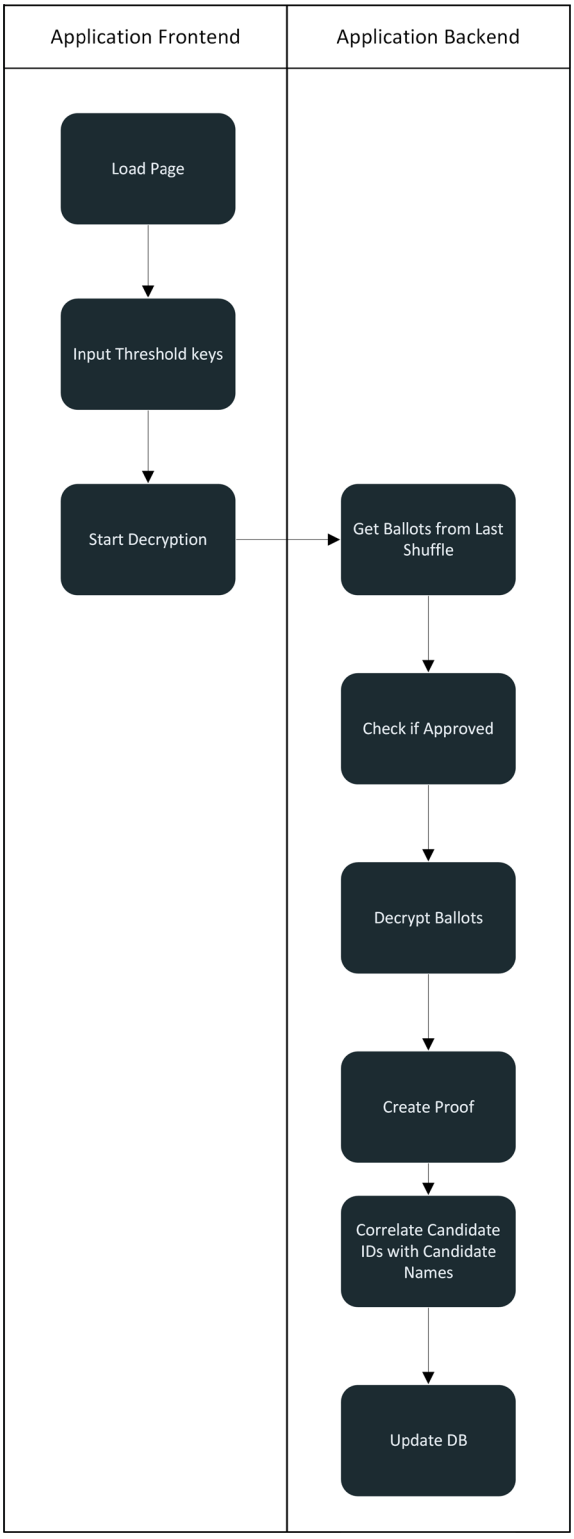

Figure 54 - Ballot decryption process diagram

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

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 outputted and a timestamp.
2. The decrypted ballots containing the candidate names will be added to the database in the `electionResults` array.
3. The property `{electionComplete: true}` will be set.

Vote Counting

As previously mentioned, the ethlect. application allows for the counting of votes either through the mixing of digitally casted voted with pen-and-paper votes (if the system is implemented in conjunction with traditional voting) or digitally via the votecounter application.

Both these options have been described in the *Objectives of ethlect.* section of the paper. This section will focus on the digital counting of votes and will explain the votecounter application.

Digital Vote Counting

Figure 55 illustrates the architecture of the application. The app is packaged alongside ethlect. in the repository and can be run by any party interested in counting the votes themselves.

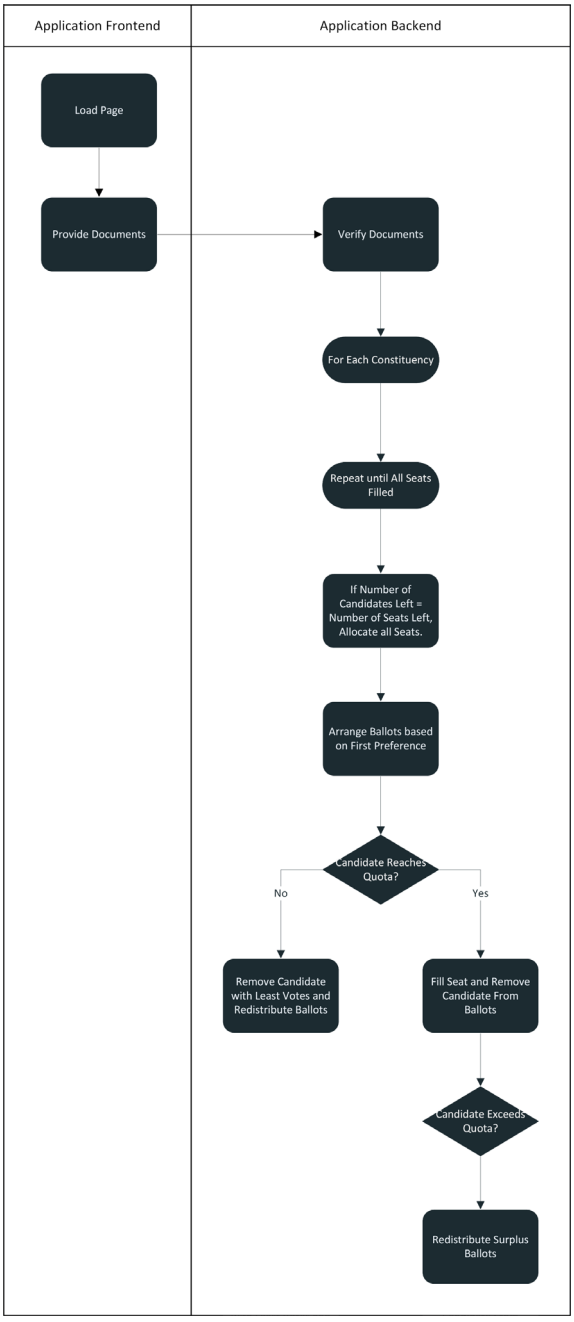

Figure 55 - votecounter application architecture diagram

By heading over to the app's home page, the user is instructed to upload two files: the election constituency and election results files as shown in figure 56.

Both files can be accessed and downloaded publicly by visiting the election's audit page on the ethlect. application (described in the auditability section).

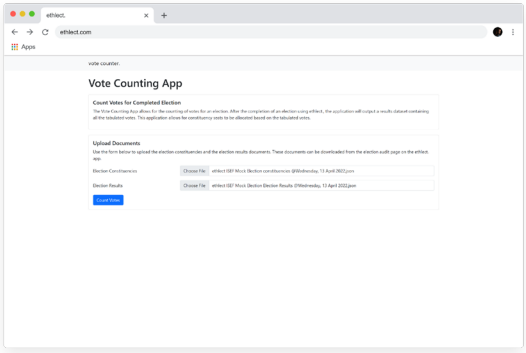

Figure 56 - votecounter app document submission

The user can press the *Count Votes* button which will send the documents uploaded to the backend at `/api/count`.

This route will first verify that all uploaded documents are formatted correctly and will then start the counting process described below:

1. The application will separate ballots based on their constituency and will then do the following for every constituency until all seats in the constituency are filled:
2. The application will first check if the number of candidates remaining is equal to the number of seats remaining in which case it will allocate all seats accordingly.
3. The application will arrange all ballots remaining in piles according to their first preference.
4. The number of ballots in each pile will be counted to check if any piles meet the quota.
5. If a candidate's pile meets the quota, the candidate will be given a seat and will be removed from all remaining ballots in the system.
6. If the respective candidate exceeds the quota, the surplus ballots will be redistributed as outlined in the Objectives of ethlect. section of the paper.
7. Should no candidate reach the quota, the candidate with the least number of votes will be eliminated and all their ballots will be redistributed to the remaining candidates based on the next preference.

This process will take a few seconds after which all seats in all constituencies will be filled. The application will then compile a list of elected candidates for each constituency and send it to the frontend where it is displayed as shown in figure 57.

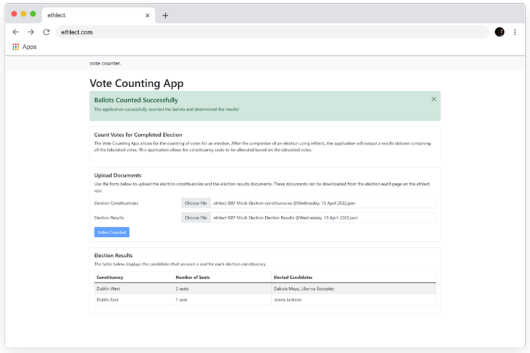

Figure 57 - votecounter app successfully counted votes

Auditability

Auditability is crucial to ethlect. as it allows for all stages of the election 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 election data.

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[29]. 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 in the MongoDB database.

Database Structure

Table 5 shows all the fields in an election document in the database, provides a short description of each entry, and notes if the entry is public or private.

Field	Data
electionID (public)	The unique ID of the election.
electionVerified (public)	Marked as true once the threshold 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 the casting of ballots.
electionTabulating (public)	True if the electoral period is over and the election is tabulating.
elGamal (public)	The $g, y, p, SHA384(x)$ Elgamal values
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 an array of 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 that can be voted for. Each candidate object has a 3-digit representative value and the respective 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,

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

	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

As previously mentioned, all public datasets related to an election are accessible from the election's audit page as shown in figure 58.

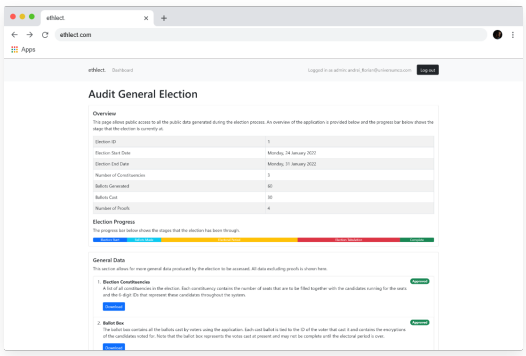

Figure 58 - Election Audit Page

This page is public and does not require any authentication to access. The following data can be downloaded from this page:

- Election Constituencies
- Ballot Box (as of time of download)
- Election Results
- Ballot Processes and Proofs

Datasets become available to download in the audit page as they are generated by the application.

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 proofs associated with ballot processes are stored in shuffle objects in the shuffles array in the ethlect. database. Each shuffle object is formatted as shown in table 6.

Field	Data
shuffleID	A unique ID for the process
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 process 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 to validate the specific operation performed on the ballots (generation, shuffle, decryption)

was completed correctly. 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 from the audit page and are displayed as shown in figure 59. This allows for independent parties to prove an election's integrity.

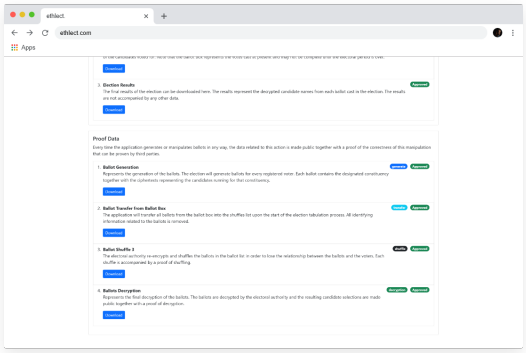

Figure 59 - Election audit page proofs

Proving Proofs

An application that allows for the automatic verification of proofs outputted by the ethlect. system is included with the package in the repository. The auditapp allows for the proving of correctness of the generation, shuffle, and decryption processes with the click of a button, requiring no technical know-how.

The auditor simply uploads the proof file as provided by the application to the form on the audit app and submits it as shown in figure 60.

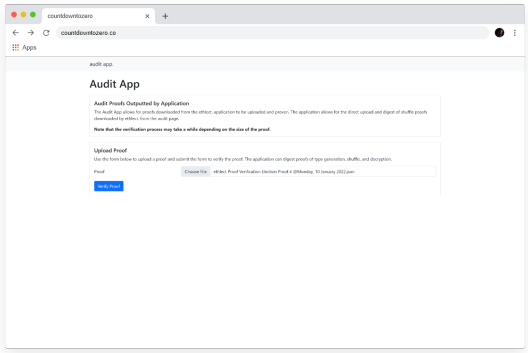

Figure 60 - Upload File to Audit Page

Upon submitting the form, the application will send a request to `/api/verify` with the proof document uploaded.

This route will firstly identify the type of proof submitted (ballot generation, shuffle, decryption) and will then ensure that the file is properly formatted.

The app will then process the data in the proof as explained in the appendices – each proof is processed in a different way.

If the proofs resolve successfully, the application will send a success message back to the frontend to assert the correctness of the process as shown in figure 61. Should the proof fail, the backend will respond with an error alerting that the ballot process is likely corrupted.

The source code for this application is open source allowing auditors to understand the processes behind the app and construct their own solution should they wish to do so.

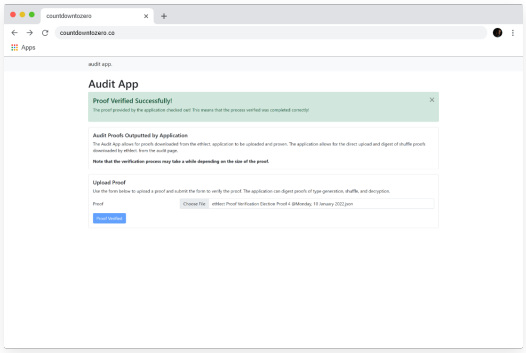

Figure 61 - Verification Successful

Security Analysis

This section of the paper will discuss the security of the application in detail. I will present all ballot processes and highlight how ballots can be traced throughout the system from generation to counting. I will then assess the application's security guarantees and briefly highlight the testing conducted. Finally, I will present common threats the application may face, and models designed to identify and resolve them.

Ballot Verification Chain

Figure 62 illustrates all the stages that ballots go through in the application. As previously noted, all ballot processes are accompanied by a proof of correctness that can be audited by third parties to ensure the integrity of the process.

By verifying each process, 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 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 dispute regarding the integrity of ballot processes.

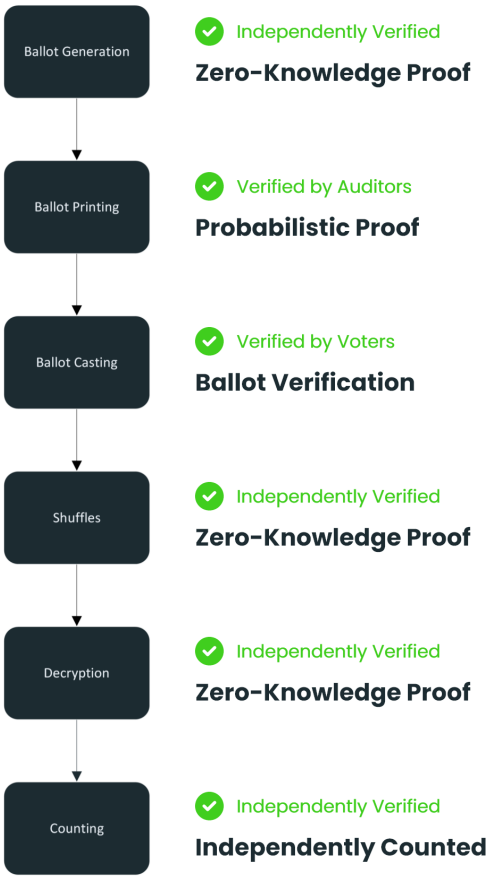

Figure 62 - Verification chain diagram

Ballot Generation

The ballot generation process is done by an election admin through the ethlect. application. The generation process takes place on the application's server and releases a zero-knowledge proof of

correctness which is then publicised on the election's audit page.

This proof can be downloaded by any party from the website and audited using the provided auditapp or an application the auditor composed.

Should the proof not work, the auditor should contact the election authority and inform them of this. If a threshold number of auditors submit complaints, the election authority should investigate the issue further and work with the auditors to identify if the proof outputted does in fact alert towards an issue with the process.

If this is the case, the election authority can easily delete all generated ballots and generate a new set to replace them.

Ballot Printing

The ballot printing process must take place at a safe and monitored location and the decryption of ballots must be completed on an airtight computer.

These restrictions prevent the public from verifying proofs of correctness for this process which instead relies solely on independent auditors.

A group of auditors should be present during the printing process and generate auditable ballots as described in the ethlect. Workflows section. These auditors are entrusted with correctly reporting their audits – cryptographic methods such as the use of Pedersen Commitments are used to ensure correct reporting by auditors.

The auditors are to provide their auditable ballots to the ballot decrypting computer where their ballots are randomly mixed with the normal ballots.

After the decryption and printing of ballots, the auditors are to identify the ballots they cast by ballot ID and verify that the mapping between candidates and PINs represent the values they generated.

If any of the auditors can prove that their ballot was printed incorrectly, the printing process can be restarted without any loss.

Ballot Casting

Ballots are returned digitally by voters using their physical ballot as reference. After casting their ballot, the voter can verify that it was recorded as intended in the system.

If the voter is not satisfied with the recording of their ballot, they can simply cast another ballot which replaces the previous.

The system generates two ballots per voter by default allowing them to easily recast a vote should they need to. In the case where a voter wishes to vote a third time, they can contact the election authority and request a new ballot be generated. This new ballot will be verifiably generated (as the other ballots)

and added to the digital ballot record. The ballot will then be sent to the voter via mail.

The electoral authority should monitor the number of voters who cast another vote as a large enough number could signal a potential flaw with the application as opposed to voter hardware.

Shuffles

All shuffles are completed at the command of the election admin on the application's backend. Every shuffle is accompanied by a zero-knowledge non-interactive proof of correctness which is made public on the election's audit page.

Any third party can download this proof and verify it using the provided application or their own.

It is important to note that all ballot proofs include the results of the previous ballot process, hence if these results are included correctly, it is safe to conclude that the process is using the correct data.

The election authority should provide an audit window where auditors can verify the correctness of the process before the admin approves the shuffle. Should a threshold number of auditors report an error with the proof, the shuffle can simply be discarded, and a new shuffle can be completed without any loss.

Decryption

Ballots are decrypted by the election admin using the threshold keys provided by the keyholders. The decryption process occurs in the application backend and releases a non-interactive zero-knowledge proof which is published on the election's audit page.

Any third party can download this proof and verify it to ensure its correctness. As with previous proofs, should a threshold number of auditors signal issues with the proof, the election admin can simply run the decryption again with no loss.

Counting

The counting process can be performed by any interested party using the decrypted ballots provided by the application. Hence any party can ensure that the seats are correctly allocated by counting the ballots themselves.

Security Guarantees

The application is composed of many elements proposed and reviewed in other internet voting systems revised to work with the PR-STV system. The application also implements novel elements that have been designed specifically for it.

STV type elections are the most difficult to design internet voting solutions for because of their mechanism requiring the selection of multiple candidates in preferential order (Carback, Internet

Voting, 2022). This factor combined with the specificity of PR-STV resulted in no application being developed thus far for this system.

The ethlect. application employs code voting which involves the marking of ballots using codes representing candidates. This is implemented in multiple end-to-end verifiable systems and is the only method to circumvent vulnerabilities in voter hardware in remote voting at present (Zagórski, et al., 2013).

Code voting is paired with a hybrid system involving the transmission of physical ballots to voters via mail and the return of these ballots over the internet. This system is also implemented in Remotegrity and other i-voting apps (Zagórski, et al., 2013).

At least one element in the system must hold a mapping between a candidate and the code a voter must input to select this candidate. This layer cannot be a digital layer of the application as attacking a digital layer is proven to be simpler than a physical layer (Batt, 2019).

The storage of this mapping in the physical layer of the application allows the app to benefit from the security properties of mail voting. The use of a hybrid approach such as this appears to be the only known way to assure voter anonymity in case of an attack (Carback, Internet Voting, 2022).

The ballot generation and printing processes are unique to ethlect. The application generates and encrypts ballots before they are cast as to allow for code voting. The ballots are encrypted such that no parties (including the application) have a mapping between candidates and their respective PIN.

This mapping is created while printing ballots in an airtight setting, hence the mapping is kept away from any digital layer of the application.

The voter registration workflow used in ethlect. is one of the strongest in any i-voting system due to its integration with the national voter registry and the Stripe Identity Service allowing for the secure onboarding of registered voters in minutes.

The ballot casting and verification process is similar in nature to Remotegrity (Zagórski, et al., 2013).

Voters are allowed to cast another vote that spoils the previous should they wish to do so. This mitigates voter coercion and the incorrect recording of ballots by the application and is an adequate way of solving these challenges (Carback, Internet Voting, 2022).

The application backend running the shuffle and decryption processes are an adaptation of the system proposed in Simple Verifiable Elections (Benaloh, Simple Verifiable Elections, 2006).

Security Testing

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 users can access them.

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.

The implementation of the end-to-end verifiability elements was tested to ensure their accuracy, security, and adequate provision of proof.

Threat Models

ethlect. has been designed with the assumption that all layers of the application can be corrupted and has no inherent reliance on any hardware or software element.

The system is designed such that in the case of all private data being leaked and/or the compromise of all layers of the application, an attempt to tamper with ballots cast through the system will still be identified by auditors. Hence elections held on the system cannot be compromised without the compromise being identified. This is indeed a requirement of e2e-v systems.

This section will describe the worst-case scenario that could occur as a result of various secrets being leaked and the compromise of various layers in the application.

Known Challenges

There are two challenges that have not yet been solved in remote voting: these are vote selling and private ballot casting.

A solution is yet to be found for ensuring that a voter casts their ballot alone remotely (National Academies of Sciences, Engineering, and Medicine, 2018). This is the primary challenge with remote voting.

Voters could potentially provide entities with their ballots and credentials for them to cast a vote on their behalf. This is the case for all remote voting implementations including postal voting (Kitcat, 2015).

ethlect. does however make it more difficult to successfully buy and sell votes, as selling a vote requires sending the physical ballot, login credentials, and 2FA token to an entity, which involves a greater complexity when compared with vote selling for postal voting.

Data Compromise

The Ballot Verification Chain section explains how any compromise in ballot processes will immediately be identified by third party auditors.

The application also stores other datapoints in election documents in the database, some of which are private as shown in table 5. Table 7 describes the worst-case scenario that could occur as a result of these datapoints being exposed or mutated.

Field	Data
rsaKeypair (private)	Should the keypair be leaked, any transfer of threshold keys between keyholders and the application can be intercepted, and hence the keys can be decrypted. Should the threshold number of keys be intercepted and decrypted, the election's Elgamal private key can be composed, and all ballots can be decrypted. This would result in the compromise of all voters' anonymities but would not affect the election's integrity. Should this compromise be detected, a new keypair can be generated and used for all future communications.
ballots (private)	The generated ballots are kept private by the application to prevent malicious actors from randomly mutating voters' candidate selections with less difficulty. Should this data be exposed, malicious actors would have access to the PINs available on each ballot. It is important to note that they would not have access to the candidates these PINs represent, yet a malicious actor could alter a voter's ballot randomly (without knowing who they are voting for) to create confusion should they have access to available PINs.
ElGamal Private Key (not stored)	The ElGamal private key is never stored in any memory element and the only way this key could be intercepted is either through the interception and composition of the decrypted threshold keys or access to the RAM element on the application's servers. Should this key be leaked, voter anonymity would be compromised but the election's integrity would be persisted. In the case of a detected compromise, depending on the stage of the election, it can be decided to either terminate the election and restart it or to proceed regardless.
Authentication Database (private)	The leaking of this database would result in the exposure of voter credentials. User passwords are stored as a hash in the database making it more difficult to access this datapoint. Ultimately, an attacker would need to have access to the voter's unique physical ballot to cast a vote on their behalf, hence being able to log into their account would not be sufficient.

Table 7 - Election Document Structure

Ballot Printing Compromise

The ballot printing process is arguably the most vulnerable element of the system and must be done with great care.

This layer exposes the relationship between IDs and candidates in a centralised location meaning that any

unauthorised viewer could potentially observe these relationships hence compromising voter anonymity.

An option to ameliorate the impact of such breaches is to print ballots at multiple facilities hence making it harder to target all facilities.

Should a breach be identified, ballots can be regenerated to replace the existing ones and the printing process can begin again once the breach has been resolved.

Ballot Casting and Verification Compromise

As previously mentioned, an attacker would require access to both the voter's unique ballot and control over their computer or access to their ethlect. voter account to successfully alter their vote in a meaningful way.

Conducting a successful attack is infeasible as it would involve having access to a great number of voter specific ballots and credentials. A vote cannot be cast with intent having access to only one of these items.

The security guarantees provided by the physical layer are analogous to those provided by postal voting and gaining access to a voter's account requires their email, password, and time sensitive 2FA token.

Without access to the physical ballot, an attacker can mutate a ballot after it was cast by the voter, but this mutation cannot be with intent (as the attacker would not know which PIN represents which candidate) hence serving no purpose. The voter can always vote again in this case and replace the altered vote.

Attackers could mutate the voter's view when verifying their ballot to make it seem that their vote was recorded incorrectly. If this is the case, the voter can directly download the ballot box dataset from the application and manually verify their vote.

Compromise in Application Backend

The worst-case scenario is for spyware type malware to be installed on the servers monitoring election processes. This spyware, should it have privileged access to the computer's RAM, could potentially extract the election private Elgamal key and compromise voter anonymity.

Test Election

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

Voters were registered, ballots were generated and printed using the ballotprinter application and all voters cast a ballot in the election where some

casted multiple. All voters verified their votes directly and the ballots were shuffled twice and decrypted.

The votes were counted using the votecounter app and all proofs were checked using the auditapp. The election proved to be a great success proving the functionality of the system. Details about the election are in the project's repository.

Conclusion

Democratic elections are the instrument that enables people to select representatives to advocate for their beliefs. Distrust in elections is reflected in a steady decrease in voter turnout and increase in anti-democratic movements worldwide.

The primary problem electoral systems face is that their means of guaranteeing the secret ballot results in an inability to incontestably prove electoral integrity – a voter receives no guarantee that the vote they cast has been tabulated correctly.

Recent breakthroughs in cryptography can be used to achieve ballot secrecy while allowing for the indisputable verification of an election's integrity.

Such breakthroughs lead to the development of internet voting – the ability to cast votes using personal computing devices. Although such implementations can have many benefits, guaranteeing the security of such systems is considered an insurmountable obstacle due to the inherent security vulnerabilities in voter hardware. Due to security concerns, internet voting is considered inappropriate for use in high-stakes elections.

I propose an innovative internet voting solution that provides full-stack security and adheres to end-to-end verifiability standards. The inevitable security vulnerabilities in voter hardware are circumvented through the implementation of a hybrid system which involves casting ballots through the introduction of encrypted candidate selections from posted paper ballots into personal computing devices.

My solution allows third parties to independently verify the integrity of an election at all stages while persisting voter anonymity. The innovative methodologies employed in this application achieve unparalleled security and form the basis of a novel end-to-end verifiable internet voting application.

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[30].

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

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.

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

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

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[31].

The challenge for the application is that it cannot simply release the key used to encrypt the ballots as this would prove

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

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 SHA384 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[32]. This results in bits $c_1, c_2 \dots, c_n$ 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 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. This proof provides a guarantee of $\frac{1}{2^n}$ that the process is completed correctly. Setting $n \approx 100$ should be sufficient to ensure the correctness of the process. 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

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

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 $c = SHA384(g^k \parallel b^k)$ where $SHA384$ is the SHA384 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 $c = SHA384(g^r * y^c \parallel b^r * P^c)$ where $P = \frac{a}{M'}$, where M is the decrypted candidate ID being verified. If the check passes for all ballots, the decryption process is proven to have been completed correctly.

Bibliography

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

Adida, B. (2017). Helios: Web-based Open-Audit Voting. *USENIX Security Symposium*, 335-348.

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/

Benaloh, J. (2006). *Simple Verifiable Elections.* Microsoft Research.

Benaloh, J. (2022, April 12). Internet Voting. (A. Florian, Interviewer)

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. (2022, April 14). Internet Voting. (A. Florian, Interviewer)

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 Local Government Ireland. (2018). *A Guide to Ireland's PR-STV Voting System.* Dublin: Government of Ireland .

Department of Housing, Planning and the Local Government. (2020). *Costings Breakdown GE 2020.* Dublin: Not publically available, must 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*.

Gomez, B., Hansford, T., & Krause, G. (2007). The Republicans Should Pray for Rain: Weather, Turnout, and Voting in U.S. Presidential Elections. *The Journal of Politics*, all.

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). Internet Voting. (V. News, Interviewer)

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

Herrnson, P. (2022, April 11). Internet Voting. (A. Florian, Interviewer)

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.

Kitcat, J. (2015, March 30). leader of Brighton and Hove City Council. (T. Guardian, Interviewer)

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.

MacAlpine, M. (2022, April 14). Internet Voting. (A. Florian, Interviewer)

Martin, D. (2022, March 7). Internet Voting. (A. Florian, Interviewer)

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

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

Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., & Halderman, A. (2014). *Security Analysis of the Estonian Internet Voting System.* Michigan: University of Michigan.

Stuart-Mills, I. (2021, November 26). Internet Voting. (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, March 17). *Timeline of the 2020 United States presidential election (November 2020–January 2021)*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Timeline_of_the_2020_United_States_presidential_election_(November_2020%E2%80%93January_2021)

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

Zagórski, F., Carback, R., Chaum, D., Clark, J., Essex, A., & Poorvi, V. (2013). *Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System.* Washington: International Association for Cryptologic Research.