Thesis structure:

Abstract

Introduction ✅

-Brief overview of the general issue about What is voter coercion and why are online voting systems susceptible to it ✅

Background ✅

 Define **voter coercion** (types, contexts, real-world examples if available). ✅

 Categorize **anti-coercion strategies** (e.g., re-voting, fake credentials, delayed verification). ✅

 Briefly mention technical or usability trade-offs inherent in these strategies. ✅

Background Literature ✅

-Overview of other coercion-resistant voting system prototypes and their tests ✅

-Overview of Loki ✅

Technology stack ✅

Development methodology ✅

Design goals and implementation ✅

Usability study

-Methodology

-Results

Discussion

-Key areas of common failings

-Areas for future research

Conclusion

**Introduction**

A natural continuation of government digitalisation is enabling citizens to vote in elections online. Countries like Switzerland and Estonia, at the forefront of modernising bureaucracy, experimented with digital voting. Estonia declared the development a large success and in 2023, more than half of the votes were cast over the internet for the first time since the introduction in 2005.

Despite the convenience, internet voting is not without its issues. One such issue is the potential of voter coercion. Unlike physical voting booths in healthy democratic systems, online voting solutions have no way to guarantee the user’s right to the secret ballot. A coercer can thus be physically present and watch while the user casts their vote, forcing them to cast a vote against their true intentions. This risk, if not adequately addressed, can undermine the fundamental principles of free and fair elections, and can become an obstacle to the widespread adoption of digital voting systems.

Several research papers were published detailing novel internet voting systems that employed strategies to prevent voter coercion. One such strategy was called Flexible Vote Updating, which allowed voters to change their vote at any point before the election closes, potentially letting a coerced individual correct their vote after being forced to vote under coercion. The Estonian internet voting system currently allows this.

While helpful in increasing the system’s coercion resistance, Flexible Vote Updating has a key weakness – the coercion can occur at the last minute of the election period, making it impossible for the user to change their vote at a later point. The research paper Thwarting Last-Minute Voter Coercion by [TODO: CITATION] from the IT University of Copenhagen details a novel internet voting system named Loki, which focuses on preventing this scenario alongside the more usual coercion strategies.

The goal of this project is to design a user interface for Loki. As most security features usually come with a usability trade-off, the key area of focus for the UI design is to maximise the usability, while preserving all of Loki’s security features.

**Background**

**Voter Coercion**

Coercion is a situation where an individual is pressured to cast a vote that does not reflect their true intentions. This can be due to external pressures, threats, offering of rewards, or manipulative tactics. In free elections, coercion is mitigated by each citizen’s right to a secret ballot, which stipulates that no one may be present to see one filling out their ballot. This right is protected by the official polling stations, where government workers and security personnel enforce it.

Internet voting fundamentally alters this environment. It removes the spatial and procedural safeguards of physical polling places, allowing ballots to be cast in uncontrolled settings. As a result, there is no effective mechanism to guarantee the voter’s privacy or prevent external influence during the act of voting.

Coercion in digital elections can take several forms. A coercer may instruct the voter to select a specific candidate, abstain from voting, or even surrender control of their voting credentials. Vote buying also becomes more viable, as coercers may demand proof of compliance—such as screenshots, voting receipts, or direct access to the system. In more severe cases, coercers may use technical means, such as phishing or social engineering, to impersonate the voter and submit a ballot without their knowledge or consent.

**Coercion-resistance**

Coercion resistance is the capacity of a voting system to prevent adversaries from forcing voters to cast a particular vote or from verifying whether a voter complied with such demands. It is a critical security requirement for internet voting systems, intended to preserve the secrecy and integrity of the vote even in the presence of coercion attempts.

Academic literature defines coercion resistance through formal security models. Many of these rely on game-theoretic frameworks, analysing whether a coercer can infer a voter's true choice with better than random accuracy. Others use the concept of observational equivalence, where a system is considered coercion-resistant if a coercer cannot distinguish between a coerced vote and one cast freely.

**Coercion-resistant systems proposed in background literature**

The following section presents several proposed coercion-resistant voting systems and their key design strategies. Each system underwent usability testing or theoretical evaluation, revealing specific weaknesses. These recurring usability issues are explained in the subsequent section.

Besides usability and coercion, another crucial issue with internet voting systems is that they must also remain secure from insider tampering – they must not rely on a single authority to trust with all security properties. These issues are out of the scope of this project.

**Fake credentials**

This method involves providing users with the ability to create one or more fake login credentials, indistinguishable from the real ones. When a voter is under coercion, they can login using one of the fake credentials or present them to the coercer, casting a vote that will then be silently nullified during the tallying process. A key security advantage is that the user can effectively deceive coercers, provided that the fake credentials are cryptographically indistinguishable from the real ones. [1] [3]

**Deniable revoting**

This method allows the user to cast multiple ballots, with only the last submitted ballot being counted. A voter who is being coerced can initially cast their vote as instructed by the coercer. The voter can then login again to cast their actual vote, once they can do so privately. This new vote will overwrite the previous coerced one. Like fake credentials, this allows the user to seemingly comply with the coercer, while removing the hurdle of managing multiple credentials. [6]

The country of Estonia, currently the only one with full-scale nationwide internet voting, offers this option.

**Decoy tokens**

This method provides the user with a set of “voting tokens”, out of which only a smaller subset is valid, while the rest are decoy. When voting, the user selects options on their ballot by assigning them the valid tokens, while assigning the decoy tokens to the rest. Since the coercer does not know which tokens are valid and which are decoys, this theoretically allows the user to cast their intended vote even if the coercer is physically watching them during the whole process. [7]

**System for this project’s UI design – flexible vote updating**

The system Loki, as described by [8], combines characteristics of all three methods mentioned above, enabling users to both revote and cast deniable ballots under coercion, similar to the fake credentials and decoy tokens systems.

Loki maintains a Cast Ballot Record (CBR) for each voter on an append-only bulletin board. This bulletin board is public to ensure transparency and verifiability. The CBR contains all ballots associated with the voter, including those cast by the voter and “noise” ballots generated by the Voting Server. The Voting Server periodically adds new noise ballots to the user’s CBR, which are computationally indistinguishable from real ballots. These serve to obfuscate the user’s voting pattern and to make it impossible to determine the user’s real ballots from the public bulletin board.

The voter maintains a secret list containing the indexes of the actual ballots they cast within their own CBR. When a voter casts a ballot, they encrypt both their vote and the list of indexes, sending both to the Voting Server. If the list of indexes is correct, the Voting Server accepts the ballot and obfuscates it with noise ballots. If the list is incorrect, the system assumes coercion and discards the vote silently by generating noise ballots indistinguishable from real ones. Neither the user nor a coercer can detect that the vote was invalidated, and it cannot be inferred from the CBR.

When voting under supervision, the coercer cannot determine if the submitted list of indexes is correct or not and thus cannot know if the coerced vote will be counted or not. This gives the user a second way, alongside revoting, to resist coercion. This mechanism depends on the voter having at least one opportunity to cast a valid ballot.

**Usability issues identified in background literature**

**Comprehension of New Concepts**

Voters often struggle to understand novel anti-coercion mechanisms without clear guidance. In a usability test of a coercion-resistant voting app (“Vote App”), many participants failed to grasp core features when instructions were minimal. For example, users did not realize that the system assigns a unique voting PIN that remains constant, nor that they were allowed to **re-vote multiple times** – concepts very different from typical apps. Several participants also misunderstood the purpose of the *fake credential (ruse PIN)* meant to thwart coercers, with some questioning why they needed a special “decoy” PIN at all when they could “simply enter a random one”​ [1]. These gaps in understanding highlight a consistent challenge in prior systems: critical coercion-countering features are often misunderstood or overlooked by users due to unclear interface communication.

**Memory Burden and Secrecy**

Many coercion-resistance strategies rely on the voter memorizing a secret or performing an extra step. This raises a **usability vs. security tension**: the voter must remember secrets (like an alternate password, PIN, or code) and recall them under stress, but humans have limited memory reliability. Preliminary findings from a study (N=26) on voter coercion found *memorability* to be a pivotal factor – most counterstrategies require the voter to remember a secret, and users perceived this as a usability hurdle​ [2]. If a voting system requires voters to retain a PIN or credential without writing it down (to avoid coercers finding it), memory lapses could lead to mistakes. Indeed, a user study of a fake-credential scheme showed ~10% of participants *mistakenly used the wrong credential to vote*, effectively invalidating their ballot​. This error occurred even though 96% of participants *understood* the concept of fake credentials in theory​ [3], underscoring how remembering and correctly using secrets under real conditions is challenging.

**Information Overload vs. Lack of Guidance**

Striking the right amount of instructional information is difficult. Voters need to be informed about unusual steps (like “enter a backup code” or “choose a control number”) but excessive text can overwhelm or annoy them. The Vote App study found that some participants felt the on-screen explanations were *too long or detailed*, while others felt those explanations were *essential* for such a security-critical application​. Too little information led to user confusion in that study – for example, encountering a screen with **“control numbers”** to select left many users asking *“What am I supposed to do now?”*​. At the same time, lengthy warnings or technical jargon can reduce usability. Finding this balance is an acknowledged challenge [1].

**User Perception of Need and Trust**

Another challenge is that not all voters perceive coercion as a real threat, which affects their willingness to tolerate extra security steps. Interviews reveal that if voters think voter coercion “could never happen here,” they may see anti-coercion features as unnecessary complications [2]. On the other hand, participants with personal or direct experience of coercion were more willing to use safeguards and even rated a coercion-resistant system as *equally trustworthy as traditional voting*​ [3]. Prior research noted that introducing entirely new voting concepts requires careful user education and even public awareness campaigns, but without undermining confidence (for example, scaring users about threats to the point they lose trust in the system)​ [3][4].

**Stress and Performance**

Coercion scenarios are high-stress situations for users. A voter under pressure (real or perceived) is more prone to errors. Usability testing in controlled settings can only partially simulate this dynamic. Researchers caution that it’s hard to truly assess how users will behave “in a situation which involves fear” [1]. Prior work has also shown that limited system feedback, often intended to preserve ballot secrecy, increases the cognitive burden on voters who must complete sensitive actions without confirmation​ [4].

**Design Goals and Implementation**

**Maintain plausible deniability at all points**

Plausible deniability refers to “circumstances where a denial of responsibility or knowledge of wrongdoing cannot be proved as true or untrue due to a lack of evidence proving the allegation.” [Plausible Deniability Law and Legal Definition (US Legal, Inc., 2023). <https://definitions.uslegal.com/p/plausable-deniability/>] Since the coercer and the user will see the same user interface, both must be considered as the same person from the view of UI design. Any information revealed to the user is also revealed to a potential coercer. This means that the interface must not disclose any information about the user’s voting history to the user. This gives the user the power of plausible deniability for all actions they take under supervision. In other words, they can intentionally cast an invalid ballot under supervision while plausibly claiming its validity.

This presents a significant usability trade-off, as users instinctively expect the website to know these things about them and when it looks like it doesn’t, they perceive it as a malfunction (see section Usability Study, Results). The UI must therefore take great care to explain to the user that every instance where they might expect the site to already know things it is asking them about, that this is intentional and for a reason.

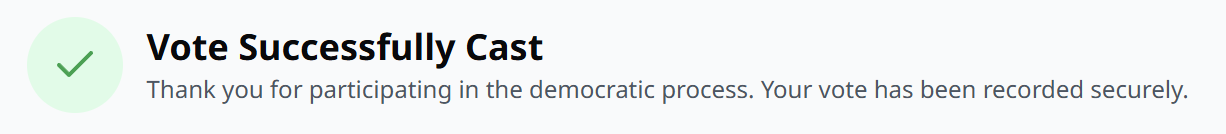


Figure 1: The UI displays a "vote successfully cast" message regardless of the ballot's actual validity – giving the user plausible deniability when intentionally casting an invalid ballot, while also sacrificing usability by not informing the user of mistakes

**Prevent Critical User Errors**

The design must be robust against stress-induced errors. In a coercive moment, a voter may not have time to read help pages or recover from mistakes. Any confusing or fragile step could result in either compliance with the coercer or an accidental vote invalidation. In the fake-credential study, 10% of users accidentally used their decoy credential to vote​ [3]. In Loki’s case, the user can avoid dealing with the security features when casting their first vote, but not when revoting. The UI should therefore provide enough information during the revoting process to minimise the risk of accidentally invalidating a vote meant to be valid. It should also inform the user about the possibility to vote physically in a situation where the user can’t recall their secrets.

During internal testing, a common pattern was that users felt everything was obvious and didn’t read the initially less visible security notices. This resulted in them sometimes unintentionally casting an invalid ballot. Despite being advised against in the Vote App evaluation, the eye-grabbing red warning texts helped alleviate this issue.

A screenshot of a ballot

AI-generated content may be incorrect.A screenshot of a ballot selection

AI-generated content may be incorrect.

Figure 2: Upon submitting their selection of past ballots, the users are prompted with a warning text to ensure they're aware of the possibility of casting an invalid ballot. Initial version on the left, revised version on the right

**Use Progressive Disclosure for Help**

A practical way to balance information is to layer it. Loki’s UI should provide on-demand help (tooltips, “Learn more” pop-ups) rather than wall-of-text descriptions on every screen. Vote App’s [1] interface added a “magnifying lens” icon on almost every screen which users could tap to get more detailed info about that step​. The UI could similarly include context-sensitive help – for instance, a help icon next to the *re-vote* button explaining when and why to use it. This way, novice or uncertain users can easily access guidance, while experienced users aren’t slowed down. An integrated user manual or FAQ accessible from the interface is also recommended​  
[1]. By designing help as an optional layer, one can ensure voters have the information they need without overwhelming everyone by default.

A yellow background with black text

AI-generated content may be incorrect.

Figure 3: The help button is always present at the top right of the screen

A white background with black dots

AI-generated content may be incorrect.

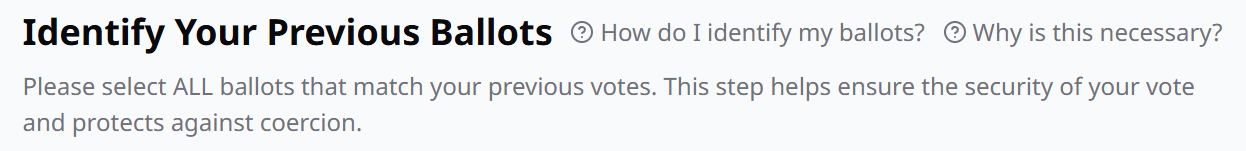
Figure 4: The help button opens the "Help Centre" card, which holds the entire manual. This is available to the user at every point of the process. The question buttons mentioned in an earlier paragraph open this manual at the relevant section

**Explain Security Features in Plain Language Without Sacrificing UX**

The UI must provide clear, succinct explanations for any non-standard voting step. Users should never be left guessing “why am I being asked to do this?” In Loki’s case, the user must identify their last previously cast valid ballots to change their vote. The Vote App evaluation showed that lack of upfront explanation led to confusion about core functions. At the same time, it should avoid alarming or technical language – instructions should be neutral and reassuring. Designers found it helpful to remove exclamation marks or words like “ATTENTION!” which can unnecessarily panic users. The UI must strive for a calm, guided experience where security steps feel like a normal part of the flow. Building trust is key – if the UI appears too complicated or “paranoid,” some users might abandon it or distrust the system.

The research suggests that users do appreciate security when it’s made usable. In the Vote App study, participants reported high satisfaction and specifically appreciated the system’s security features and overall ease of use [1]. This indicates that a well-designed UI can turn security measures into a net positive part of the user experience (users feel safer and in control, rather than burdened).

However, a caution is warranted: another study found that when users were given a stronger security briefing (making them more vigilant for threats), their subjective usability ratings went down slightly. Essentially, reminding users of security risks can cause anxiety, affecting their comfort. The trade-off was that those users were far better at detecting a simulated attack (catching a fake voting terminal) [3]. The lesson for Loki is to find a balance in messaging – educate users about security in a gentle, non-alarming way, perhaps by including a short optional tutorial or use tooltips that mention security tips, rather than a scary warning banner. By carefully tuning the tone and amount of security messaging, Loki’s UI can keep users alert without overwhelming them. As one recommendation puts it, incorporate user education and awareness early, but ensure the additional information “will not overwhelm the voter or make them distrust the system” [4]. Achieving this balance is tricky but vital for a positive user experience.



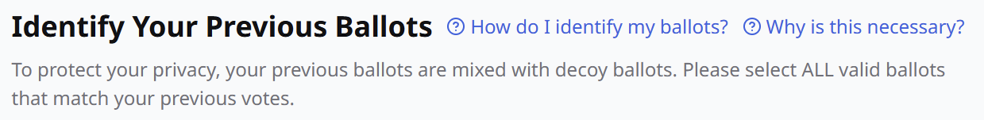


Figure 5: Every "unusual" security feature has a clickable button which opens a relevant explanation section in the help card. Instead of using technical language, these buttons are framed as questions the user might ask while using the app. Initial version on top, revised version on bottom.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6: Security features where the user isn't at risk of making a critical error are worded in a calm, non-intimidating way

**Provide Memory Aids**

Given the importance of memorability, the UI should incorporate features that help users manage the required secrets or codes. One approach is using recognition over recall – for example, representing confirmation codes or PINs with easily recognizable images or emoji sequences in addition to text. Vote App displayed a string of “Private PIN Emojis” alongside the user’s PIN as a visual cue, which some users found helpful (familiar from apps like Telegram)​ [1]. Other research similarly found that user-generated visual markings can significantly improve retention of secrets without external notes​ [3]. Another aid is letting users double-check or confirm entries: for instance, a participant suggested allowing voters to enter their PIN twice to be sure they didn’t mistype it, since the second entry would catch if they mis-remembered it and saw the wrong emoji feedback [1]. However, any such aid must be covert if an adversary is present. Designers should avoid solutions that rely on visible notes or additional devices during the voting session, because a shoulder-surfing coercer could see those​. Instead, *the* *secret itself should be easier to remember* (e.g. a passphrase of four common words) and users should be guidance on memorization techniques upfront​ [4].

In Loki’s case, users must remember the date and time they cast their ballot to be able to recast a new one later. The purpose of visual memory aids is thus to provide the user with something they might find easier to remember than a timestamp.

A white background with black and white clouds

AI-generated content may be incorrect.

Figure 7: The UI provides two memory aids – a two-word phrase and an “identicon”, both generated from the ballot’s hash

**A screenshot of a phone

AI-generated content may be incorrect.**

Figure 8: The filters in the ballot identification page allow the user to also search for the two-word phrase if they remember it better than the timestamp. The “Why am I seeing so many ballots?” help button was later revised to be blue

**Normalize Coercion-Resistance Features for All Users**

A crucial lesson is to design the UI so that using anti-coercion features doesn’t itself look unusual. Everyone using Loki should follow roughly the same visible process, whether or not they are under duress, so that a coercer observing cannot distinguish a voter who is invoking a special countermeasure​. For example, if the system supports re-voting, the interface should encourage all voters to review and even recast their vote if they change their mind – not frame it as a niche “coercion recovery” feature. Estonia’s i-voting system did this by allowing any voter to vote multiple times (only the last vote counts), making re-voting a normal behaviour​ [4]. Loki can adopt a similar approach: advertise revoting as a convenience (“You can adjust your vote until the deadline”) so that a coerced voter’s extra vote doesn’t raise flags. The key is that nothing in the interface blatantly says “coercion mode” or otherwise tips off an observer. Blending these security features into the routine flow is one of the strongest lessons from the literature on coercion-resistant voting schemes [4].

A screenshot of a computer

AI-generated content may be incorrect.

Figure 9: The UI informs the user about being able to change their vote

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 10: Revoting is presented as an equally valid option as voting for the first time, framing it as a natural thing one may do. At the same time, the system ensures the user that they are not seeing a defect and informs a potential coercer that the user is not using some special anti-coercion mode

**Iterative User-Centred Design**

A meta-lesson from all these papers is the importance of testing with real users and refining. Usable security doesn’t come for free – it requires iterative design. Kulyk and Neumann’s review explicitly recommends involving users from the start in developing coercion-resistant systems and getting feedback on prototypes​ [4]. Usability testing should be focused on Loki’s unique features (e.g., have users attempt to vote with and without a coercion scenario) to catch issues early. As seen in these studies, even well-intentioned designs had unforeseen pitfalls: e.g., users misinterpreting an icon, or failing to notice a crucial detail. By observing users, it can be discovered where the UI might be too confusing or too obvious (potentially revealing a covert action). Continuous refinement will significantly improve both usability and security. Moreover, providing continuous support throughout the election (help desks, hotlines, or in-app support chat) can mitigate the limited feedback inherent in coercion-resistant systems ​[4]. If a voter is unsure or encounters a problem, there should be resources to assist them (without the coercer’s knowledge if possible). In summary, building Loki’s frontend should be an iterative, user-informed process, not a one-off design – this ensures the final product is intuitive for voters from diverse backgrounds and capable of guiding them through a safe voting experience.

**User-Friendly Verification Mechanisms – area for future work**

Although not implemented in the current version of Loki’s UI, prior research on user-friendly verification mechanisms offers valuable guidance for future iterations.

End-to-end verifiability is often at odds with coercion-resistance, but many systems include some verification step (like checking a bulletin board for your vote’s hash or a confirmation code) that voters *can* perform. Designing this for usability is critical, otherwise few will do it or do it correctly. Studies on comparing cryptographic fingerprints offer guidance on presenting codes in human-friendly ways. They found that graphical representations (like patterns or avatar images) can make comparison quick and intuitive, but some formats led users to miss subtle differences, allowing attacks in tests​. A seemingly user-friendly method where participants had to “compare and select” the correct fingerprint from options performed *poorly* – many failed to notice when the correct option wasn’t present, so this approach is *not recommended* for critical verifications​. Instead, textual representations – even the “nerdy” hexadecimal strings – fared surprisingly well in usability and error detection​ [5]. The lesson for Loki is to choose a verification UI that maximizes accuracy for the average user. For instance, showing a short list of dictionary words or a two-row code (as some messaging apps do) might be easier to read and compare than one long hex string or a complex image. If Loki uses emoji or symbols as part of verification (as Vote App did), it should ensure users can easily tell if something is off. The Vote App experience showed that while some users liked emoji confirmations, others found them confusing or even thought the emojis were software glitches​ [1]. Thus, any novel verification aid (colours, animals, unicorn avatars, etc.) should be tested for clarity. One practical design might be combining modalities: e.g., show a small image *and* a short code – if they match what’s on the official site, great; if not, the user should know to report a problem. Ultimately, the verification step in Loki must be as simple as “look and check” with minimal mental calculation. If a voter has to manually compute something (even as simple as XORing numbers or summing values), the chance of error goes up dramatically​ [4]. Wherever possible, offload the complexity to the system and let the user just confirm a visible matching piece of data.

**Technology Stack**

The user interface was developed using **Next.js** and **TypeScript**. Next.js is a web development framework built on top of React. It provides additional functionality, such as support for multiple rendering strategies per page (static generation, server-side rendering, and client-side rendering).

A key library used was **shadcn/ui**, which provides locally stored, prebuilt UI components such as buttons, breadcrumb navigations, and accordions. Unlike traditional UI libraries that fetch components from external sources, shadcn/ui stores them within the codebase, enabling full customization and eliminating runtime dependencies.

**Design and Implementation of Memory Aids**

This implementation derives a SHA-256 hash deterministically from a ballot object’s timestamp and ID, producing consistent results for the same input. It then uses this hash to generate both an identicon and a two-word passphrase.

To create the identicon, the algorithm selects a foreground colour from a predefined, colourblind-friendly palette based on the hash’s leading bytes. It builds a symmetrical grid of boolean values by evaluating the parity of hexadecimal digits in the hash. To enhance visual recognizability, it mirrors each row vertically. Finally, it selects a high-contrast background colour—either black or white—based on the foreground’s relative luminance to ensure readability and accessibility.

To create the two-word passphrase, the algorithm parses two four-character segments from the beginning of the hash and converts them into numeric indices. It then maps each index to a word from a predefined word list [<https://github.com/bitcoin/bips/blob/master/bip-0039/english.txt>], using modulo to stay within bounds. Finally, it combines the selected words into a space-separated phrase. By basing the selection on the hash, the function ensures that the phrase is deterministic and repeatable for the same input.

The UI uses two passphrases since the 2048 word long Bitcoin word list used gives a 100% chance of duplicates for 1000 ballots.

% Collision Probability (Birthday Problem Approximation)

P \approx 1 - e^{-\frac{k^2}{2n}}

With k=1000 ballots and n=2048 words.

Not only that, we can estimate using the balls and pins formula that for k=1000 ballots and n=2048 words, there will be E≈791 unique words. This subtracted from the 1000 ballots means approximately 209 duplicates.

% Expected Number of Unique Values (Balls and Bins Model)

E = n \left(1 - \left(1 - \frac{1}{n} \right)^k \right)

By using two words, this increases the possible word combinations to 2048^2 = 4 194 304, which is more than enough to prevent duplicates for a single user.

**Development Methodology**

The project was developed by a single individual using an iterative approach. Each iteration spanned two weeks and concluded with a meeting involving the thesis supervisor and two faculty members from ITU. These meetings served to collect feedback on the implemented features and to discuss potential changes.

In addition, each iteration included informal testing sessions with other students to obtain early feedback on features under development.

A total of four iterations were completed. The final iteration focused on refining and finalising the design in preparation for the usability study.

[1] Cristiano, L., Longo, R., & Spadafora, C. (2025). *Click and Cast: Assessing the Usability of Vote App*. Fondazione Bruno Kessler & University of Trento.

[2] Christina Nissen, Tobias Hilt, Melanie Volkamer, Jurlind Budurushi, and Oksana Kulyk. (2025). *Voting Under Pressure: Perceptions of Coercion and Counter-Strategies in Internet Voting*. Manuscript shared by authors.

[3] Merino, L.-H., Azhir, A., Zhang, H., Colombo, S., Tellenbach, B., Estrada-Galiñanes, V., & Ford, B. (2024). *E-Vote Your Conscience: Perceptions of Coercion and Vote Buying, and the Usability of Fake Credentials in Online Voting*. 45th IEEE Symposium on Security and Privacy.

[4] Kulyk, O., & Neumann, S. (2025). *Human Factors in Coercion-Resistant Internet Voting: A Review of Existing Solutions and Open Challenges*. IT University of Copenhagen & Independent Researcher.

[5] Tan, J., Bauer, L., Bonneau, J., Cranor, L. F., Thomas, J., & Ur, B. (2017). *Can Unicorns Help Users Compare Crypto Key Fingerprints?* Proceedings of the CHI Conference on Human Factors in Computing Systems, 3787–3799. <https://doi.org/10.1145/3025453.3025733>

[6] Wouter Lueks, *EPFL;* Iñigo Querejeta-Azurmendi, *Universidad Carlos III Madrid/ITEFI, CSIC;* Carmela Troncoso, *EPFL (2020)*

[7] Riccardo Longo, Chiara Spadafora, Amun: Securing E-Voting Against Over-the-Shoulder Coercion

[8] Thwarting Last Minute Coercion

WIP notes:

**Potential adversary tactics:**

The adversary forces the user to vote at the beginning of the voting period.  
Countermeasure: The user can revote later.

The adversary forces the user to vote at the very end of the voting period.  
Countermeasure: The user votes earlier and then provides false information during the revoting process, i.e. says they hadn’t voted before or intentionally misidentifies their previous ballots to invalidate the new vote.

The adversary forces the user to login at the beginning and makes them leave while the adversary submits a vote. Then, at the end of the process, the adversary once again forces the user to submit a vote.  
Countermeasure: There should be a way for one to vote in person and permanently lock their vote from being overridden.

**Potential memory aids for ballot identification:**

Timestamps – can be difficult to remember but also have the advantage of the user being able to recall them even in the event of the adversary forcing the user to login at the beginning of the voting period and making them leave the room while the adversary casts a vote for the user. Verification codes or symbols prevent the user from nullifying this vote. Vulnerable to the coercer knowing when the user last cast their valid vote.

Identicons – mentioned as confusing and not a bad idea in the context of page integrity verification

Emojis – mentioned as seemingly popular with some users while seeming unprofessional or even like glitches to others

Hex codes – mentioned as successful in the context of page integrity verification, testing should be done to determine their usefulness as ballot memory aids

Colours – personally seems like a good choice to use in combination with any of the above

Used timestamp as mandatory since this is decided in the loki implementation. identicons with colour + two keyphrases as mnemonic helps. Phrases from bitcoin <https://github.com/bitcoin/bips/blob/master/bip-0039/english.txt>

One word:

A screenshot of a black and white math problem

AI-generated content may be incorrect.

Two words:

A screenshot of a computer program

AI-generated content may be incorrect.

**Usability testing logs:**

4th March w/ Ida – identified no issues, didn’t look through tutorials, said everything seemed intuitive -> the issue was that we did not explore the scenario of the coercer knowing she’s voted before prior to me explaining the system, I should provide no information and force the user to figure everything out on their own using the UI if I want to identify issues with the UI

4th March w/ Søren – quickly clicked though everything, again because everything seemed “obvious”, didn’t realise he submitted an invalid ballot during the coerced scenario -> it needs to be clearly and visibly communicated that this might happen

5th March w/ Adrian – lots of feedback about small things in the UI, wording, etc., mostly fixed now