Privacy in Electronic Voting Systems

Eoin McDonnell
Northern Kentucky University
Highland Heights, Kentcky, USA
Oldmcdonnell@gmail.com

***Abstract*— This paper explores the intricate balance between non-repudiation, anonymity, confidentiality, and privacy within electronic voting systems. It examines the integrity of voting processes in an online environment, focusing on the roles involved in building a system that ensures non-repudiation while preserving voter anonymity. Using a privacy-by-design (PbD) approach, potential solutions are analyzed to achieve this balance. Specifically, the paper proposes a system where user groups can establish their own administrative portals within the voting system, granting control over confidentiality, privacy, and voter anonymity. Additionally, the access controls for such a system and how they should be setup is critically examined.**

. Keywords— IoT (Internet of Things), non-repudiation, anonymity, authentication, privacy by design (PbD), end-to-end encryption, de-identification, re-identification, immutable, agile, FIDO, FOO, Test Driven Development (TDD)

I. Introduction and Motivation

The purpose of this paper is to explore the value of privacy by design in the development of voting systems that ensure both voter privacy and system integrity, while allowing for oversight. We focus on alternative voting methods such as Single Transferable Vote (STV), ranked choice, or priority voting, instead of first-past-the-post systems. Key to this exploration are various IoT implementation methods, with an emphasis on maintaining privacy without sacrificing transparency and security. Importantly, ease of use will be prioritized, ensuring the process remains agile, effective, and engaging while maintaining trust.

We will also address the need for creating standardized, compliant electronic systems. This involves examining existing models of electronic voting, assessing user confidence, comparing ease of use, and evaluating the security implications of each model. Ease of use is a fundamental consideration in designing voting systems, but it often conflicts with integrity. Many online systems are prone to manipulation, and voting is often seen as a tedious task, discouraging participation. The goal is to create a secure, trustworthy system that simplifies the voting process, maintains integrity, and enhances user experience, encouraging greater voter engagement.

II. Research Questions

My research study in this paper is to answer the following questions:

- What authentication methods should be considered?
- What considerations should be followed for which user roles and administrators?
- How do we guarantee nonrepudiation in conjunction with privacy?
- How complex is too complex? What ease-of-use considerations should be considered?

I hope to answer these questions in this paper and outline some of the various options that could maybe allow for several options when looking at a voting model the has a formidable structure of privacy, integrity and nonrepudiation.

III. System Fundamentals/ Project Overview

A. Rank Choice voting

Ranked choice voting is a system where you can rank your choice or candidate. In a single transferable vote system if a preferred candidate is eliminated the vote is transferred to the second choice. In Multi-Winner RCV it is the same concept. In the scenario where there are board elections, and three seats need to be filled this would be the ideal method. Multiple rounds of voting can maximize utility and minimize disclose risk as not all rounds need to be conveyed to the voters, simply the results. However, that information can be presented if the application is needed.

Sample Multi-Winner RCV Election

Candidate	Round 1	Round 2	Round 3	Round 4	Round 5
Armando Perez Democrat	27.2% 2,500 votes	25.0% 2,300 votes	25.0% 2,300 votes	25.0% 2,300 votes	25.0% 2,300 votes
Cathy Chan Democrat	19.0% 1,750 votes	20.1% 1,850 votes	21.2% 1,950 votes	34.8% 3,200 votes	25.0% 2,300 votes
Hannah Murphy Republican	14.1% 1,300 votes	14.3% 1,320 votes	20.7% 1,900 votes	22.3% 2,050 votes	27.2% 2,500 votes
Charles Lorenzo Republican	14.1% 1,300 votes	14.1% 1,300 votes	17.4% 1,600 votes	17.9% 1,650 votes	18.9% 1,740 votes
Brad M. Jackson Democrat	14.7% 1,350 votes	15.5% 1,430 votes	15.8% 1,450 votes	0.0% 0 votes	0.0% 0 votes
June Smith Republican	10.9% 1,000 votes	10.9% 1,000 votes	0.0% 0 votes	0.0% 0 votes	0.0% 0 votes

Fig. 1. Multisteat ranked choice voting [6]

$$Quota = \left(\frac{Total\ Valid\ Votes}{Number\ of\ Seats + 1}\right) + 1$$

The only extra consideration here is if it is single or multi-seat. A single seat example follows but the consideration would have to be slightly adjusted for multi-seat. This, of course, should be at the discretion of the vote creator and the option should be open. We will revisit the responsibilities of the vote creator in a later section. If there is only one position to be filled the equation will be slightly different than if it is multi seat.

```
@api_view(['POST'])
@permission_classes([IsAuthenticated])
def vote_results(request):
    vote_id = request.data.get('vote_id')
    print('vote ID : ',vote_id)
    if not vote_id:
        print('no vote ID detected')
        return Response({'error': 'vote_id is required'}, status=status.HTTP_400_BAD_REQUEST)

    vote = get_object_or_404(Vote, id=vote_id)
    preferences = Preference.objects.filter(candidate__vote=vote).select_related('voter', 'candidate')

    if not preferences.exists():
        print('preferences  ',preferences.first())
        return Response({
            'winner': candidate,
            'round': 1,
            'vote_counts': round_results,
            'final_votes': 1,
        }, status=status.HTTP_200_OK)
                # return Response({'error': 'No preferences found for this vote.'}, status=status.HTTP_400_BAD_REQUEST)

    vote_count = Counter()
    results = defaultdict(list)

    # Initialize the first round of vote counts
    for preference in preferences:
        if preference.rank == 1:
            vote_count[preference.candidate.description] += 1

    if not vote_count:
        return Response({'result': 'draw'}, status=status.HTTP_200_OK)

    round_results = [dict(vote_count)]
    total_votes = sum(vote_count.values())
    majority_threshold = total_votes / 2
    round_number = 1

    while True:
        print(f"Round {round_number} processing...")

        # Check if any candidate has more than 50% of the votes
        for candidate, count in vote_count.items():
            if count > majority_threshold:
                return Response({
                    'winner': candidate,
                    'round': round_number,
                    'vote_counts': round_results,
                    'final_votes': vote_count
                }, status=status.HTTP_200_OK)

        # Find the candidate with the least votes
        if not vote_count:
            return Response({
                'result': 'No votes have been cast or all candidates have been eliminated.',
                'round': round_number,
                'vote_counts': round_results,
                'final_votes': vote_count
            }, status=status.HTTP_200_OK)

        min_votes = min(vote_count.values())
        eliminated_candidate = None
        for candidate, count in vote_count.items():
            if count == min_votes:
                eliminated_candidate = candidate
                break

        if eliminated_candidate is None:
            return Response({
                'result': 'No candidates to eliminate.',
                'round': round_number,
                'vote_counts': round_results,
                'final_votes': vote_count
            }, status=status.HTTP_200_OK)

        # Eliminate the candidate with the least votes and redistribute their votes
        eliminated_candidate_preferences = Preference.objects.filter(candidate__description=eliminated_candidate, candidate__vote=vote)
        for preference in eliminated_candidate_preferences:
            next_preference = Preference.objects.filter(voter=preference.voter, candidate__vote=vote, rank=preference.rank + 1).first()
            if next_preference:
                vote_count[next_preference.candidate.description] += 1

        # Remove the eliminated candidate from the count
        del vote_count[eliminated_candidate]

        # If all remaining candidates have the same votes, declare the candidate with the most votes from previous rounds as the winner
        if len(vote_count) == 1:
            winner = next(iter(vote_count))
            return Response({
                'winner': winner,
                'round': round_number,
                'vote_counts': round_results,
```

```
                'final_votes': vote_count
            }, status=status.HTTP_200_OK)

        round_number += 1
        round_results.append(dict(vote_count))

    # If no candidate wins by majority, find the candidate with the most votes from previous rounds
    for round_result in round_results[::-1]:
        max_votes = max(round_result.values())
        potential_winners = [candidate for candidate, count in round_result.items() if count == max_votes]
        if len(potential_winners) == 1:
            return Response({
                'winner': potential_winners[0],
                'round': round_results.index(round_result) + 1,
                'vote_counts': round_results,
                'final_votes': vote_count
            }, status=status.HTTP_200_OK)

    return Response({
        'result': 'No clear winner found after all rounds.',
        'round': round_number,
        'vote_counts': round_results,
        'final_votes': vote_count
    }, status=status.HTTP_200_OK)
```

Fig. 2. Sample single-seat rank choice voting in python [16]

B. Project Overview

The project aim is to develop a secure and user friendly Ranked-Choice Voting System using privacy by design and test-driven development. The system should be designed to ensure non-repudiation, anonymity, and confidentiality while allowing for multiple voting rounds and live results updates. Key elements include ease-of-use, security and non-repudiation, anonymity, testing. Prior development [16] have not had the consideration of Privacy by Design and this is a road map for future development.

IV. Authentication Methods

A. Django REST Framework

A simple to implement application of authentication can be managed with SQLite running in Python with a Django rest framework [8]. The inbuilt authentication protocol where the client or frontend stores a Token to process API requests. For this data to be encrypted a asymmetric PKI exchange would need to be implemented. This may work best for smaller scale applications where the information wasn't as vital. You can implement a UUID function [11] in python for the storing of user IDs as the username is not what is tracked for any and all actions and as long as that information is protected on the server side it cannot be linked to a user regardless of when the account was created. Django uses the PBKDF2 algorithm with SHA-256 hashing by default for passwords so they are not stored in plaintext.

B. FIDO/FIDO2

Fast Identity Online (FIDO) uses a private and public key asynchronously but does not require a password. This authentication method is coupled with a device like a mobile phone which also highlights an ease-of-use. When the device registers a username with the server it creates a public and private key. The private key is stored and authenticated on the device with something like biometrics (i.e. fingerprint) and the public key is stored on the server. In our model it would be stored in the SQL server. This would have to be controlled with assigned usernames. This way the administrator would provide simply a username and that username when registered with the server would already be selected for any role available on the system. Another option is FIDO2 or WebAuthn. The demonstration on WebAuthn.io [7] showcases the simplicity and ease-of-use passwordless authentication using the WebAuthn protocol, a core component of FIDO2. Users can quickly and securely authenticate by leveraging built-in device capabilities, such as biometric authentication (e.g., fingerprint or facial recognition) or a hardware security key.

During the authentication process, users only need to register their device once by generating a public-private key pair, and future logins involve a straightforward verification step without the need for remembering or entering passwords. The entire interaction flow is seamless, involving minimal user input while maintaining high security through public key cryptography.

The WebAuthn.io demo serves as a clear illustration of how passwordless authentication can significantly improve the **user** experience without sacrificing security. It demonstrates how users can easily authenticate using their device's biometric sensors or external authenticators (such as FIDO2 security **keys**), emphasizing that this process is not only secure but also highly intuitive for end users.

FIDO has published a white paper [21] detailing how FIDO complements federation protocols and providing guidelines on how to integrate the two in order to add support for FIDO-based MFA and replace or supplement traditional authentication methods in federation environments.

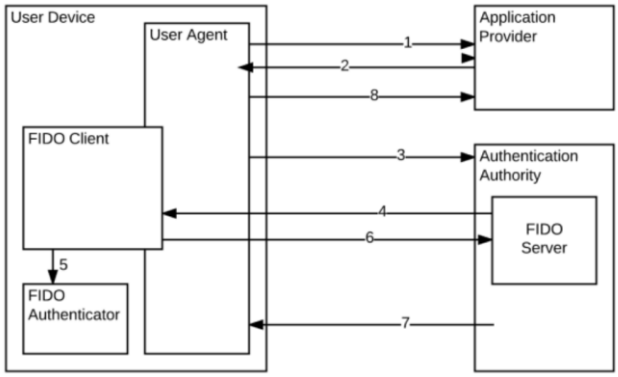


Fig. 3. Multisteat ranked choice voting [12]

1. User accesses Application Provider (via User Agent) and needs to authenticate.

2. Application Provider redirects the User Agent to the FIDO-enabled Authentication Authority with a request that the user be authenticated. This redirect allows the Application Provider to specify that the user authentication be FIDO-based (either explicitly or implicitly) according to its preferences.

3. User Agent accesses the Authentication Authority federation endpoint. The Authentication Authority determines that the FIDO authentication specified in the previous request is both relevant and possible by confirming that the policies of the Authentication Authority match with the preferences of the Application Provider.

4. The Authentication Authority sends a FIDO Server challenge to FIDO Client (via User Agent). This challenge may indicate a particular FIDO Authenticator – as determined by the Application Provider's preference and the Authentication Authority's own policies.

5. FIDO Client locates FIDO Authenticator, and the user is authenticated

6. FIDO Client returns FIDO authentication response (via User Agent)

7. FIDO Server validates the FIDO response and reports same to Authentication Authority which looks up the user and then redirects the User Agent back to the Application Provider with an authentication assertion.

8. User Agent calls Application Provider with the authentication assertion (or an artifact that the Application Provider can exchange against the Authentication Authority for the actual authentication assertion, step not shown). The authentication assertion can include the specifics of the FIDO-based authentication or more generally a policy identifier for that authentication – as previously discussed.

The bolded steps are those by which respectively,

a) the Application Provider is able to indicate to the Authentication Authority its expectations for how the user should be authenticated and

b) the Authentication Authority is able to indicate to the Application Provider details about how the user was actually authenticated.

Feature	FIDO (U2F)	FIDO2
Authentication Method	Second-factor authentication	Passwordless and second-factor authentication
Protocol	U2F Protocol	WebAuthn (API) + CTAP (client-to-authenticator protocol)
Factor Type	Multi-factor (with password)	Single-factor (passwordless) or multi-factor (with PIN/biometrics)
User Experience	Requires password + hardware token	Can be passwordless or use PIN/biometrics + authenticator
Scope	Adds security to existing password-based systems	Password replacement or passwordless login with hardware or biometric authentication
Device Types	Primarily hardware tokens	Hardware tokens, biometrics, mobile devices, etc.
Browser Support	Widely supported for 2FA	Supported by modern browsers for passwordless login and 2FA

C. FOO Protocol

A method employed by Universtiy voting systems is the FOO Protocol. The application of the FOO protocol I will be referencing uses OpenSSL and Crypto++ for end to end encryption.[9]

This method utilizes OpenSSL to establish a TLS connection between the client and server. Crypto++ is responsible for implementing the RSA blind signature process, which guarantees that votes remain anonymous. The use of blind signatures mean the server can sign votes without being able to trace them back to individual users, preserving anonymity. A blind signature in this context is a cryptographic method that allows a server to sign a vote without knowing its contents.[9]

The system assigns a Project ID at the create of the user, and users can vote on specific projects based on this identifier. User IDs can be either dispensed or added to a pool of users in an open system. Using this method does not require everyone to participate and adjusts for failover without a database. The C++ backend is responsive and situational. Given the users' handle the system administrator (SA) can mark them off the list and continue. This is another point where a privacy officer or administrator of some sort will need to come into play. The requirements for such to be unbiased should be open to audit. Someone will always have to administrate a function that needs to be verified. With Test Driven Development and Privacy by Design it should be a formidable solution.

D. Blockchain

Blockchain offers several advantages as an authentication method in voting systems. One of its key features is the use of blockchain addresses to represent identities, allowing for pseudo anonymity [15]. Blockchain is a decentralized digital ledger maintained by participants in a peer-to-peer (P2P) network. A unique feature of blockchain is that each block contains the hash of the previous block, linking them into a secure, unchangeable chain of transactions, as shown in Figure 3.

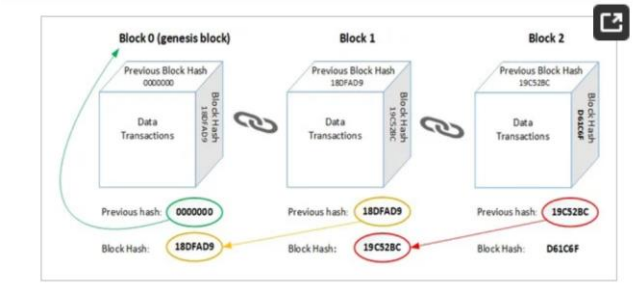


Fig. 4. Blockchain example [18]

This illustrates how blockchain works using linked hashes. Each block contains its own data, a unique hash, and the hash of the previous block. If the data in the block changes the hash changes as well, breaking the link and signaling that the block has been altered. This is the immutability and security of the blockchain. Advantages also include;

(1) Decentralization. Failure of a single node will not impact the system as a whole

(2) Transparency. Voting records stored in the blockchain are easy to view/

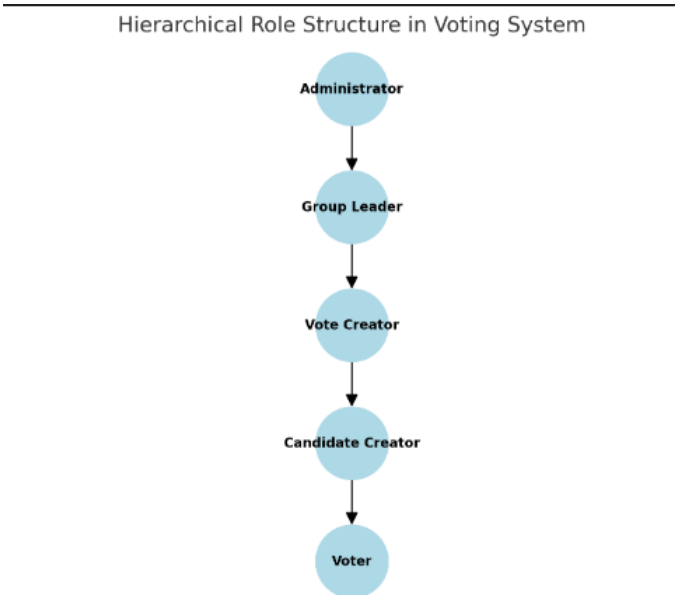
(3) Immutability. The results cannot be modified.

The figure demonstrates that any change in a block's data results in a change in its hash, which can be thought of as a fingerprint of the block. Altering data makes it obvious that something has change as there will be invalid data. This ensures records are tamper proof.

V. ROLE CONSIDERATIONS

Should all board members be trustees? A privacy, security, and organizational officer [13] tends to be the minimum amount for a group to function. The organization and technical aspects need to be divorced from the monetary situation and present only as experts. Potential access levels are system administrator, auditor, group leader, vote creator, candidate creator, and voter.

It's important to note that roles can be flexible and overlapping. For example, a group leader might also be a voter, candidate creator, and vote creator. Similarly, some individuals may only be both a candidate creator and voter, while others might hold the role of vote creator, candidate creator, and voter. This structure creates a hierarchy of permissions that must be clearly defined to avoid conflicts of interest. I would assume that most member would be voters and be able to create candidates or options but that also could be limited to one per person. The vote creator should also be able to define specific operation in the process. Open enrollment would be when candidates or options can be added to the vote, then once that is closed you would have a polls open phase and finally once the votes have all been cast or a period of time has been allotted the process enters the final vote results phase. After this vote results phase data retention rules determined by the group would determine how long the submitted information would be considered.



The system administrator holds the authority to create groups and assign privilege levels, or they can delegate this task to the privacy officer. An auditor is required to ensure the system operates transparently and in compliance with regulations. The key question is: should these super users (e.g., system administrator, privacy officer) also can create votes? Furthermore, should they be able to view breakdowns of voting stages?

When adding users to the system, everyone must be thoroughly considered, as the verification and database creation process initiates non-repudiation measures, ensuring their actions are auditable. The auditor's role should remain independent—they can confirm whether a user has cast a ballot, but not view the user's preferences.

Additionally, there's an internal debate over whether data should be kept for the various rounds of voting. This decision should be based on the team's needs and what best serves the end-user experience. To ensure election fairness, these roles and access levels must be carefully evaluated, particularly if your organization lacks privacy-focused, tech-savvy officers.

Ultimately, privacy should not merely be a function—it must be embedded into the culture and mindset of the organization.

Along with role considerations we will need to determine non-repudiation. Someone will need to be able to verify a specific user has cast a ballot. This would need to be a trusted administrative position. Most database models require a username that would be connected to a specific individual. When a vote is cast to ensure people cannot vote more than once it would have to be noted that the specific username has already registered a vote. This shouldn't be difficult to set up. By limiting the username to only one person it shouldn't be difficult to limit them to one vote per person.

VI. Ease of use considerations

Testing complexity and ease of use will primarily depend on the time it takes users to complete essential tasks like account creation, login, and viewing the results of their actions. In a well-designed voting system, users should be able to receive live updates as votes are cast, which provides instant feedback and helps reinforce the sense that their participation matters.

From a technical perspective, live result updates can be achieved using frameworks like ReactJS by utilizing reducers to manage real-time data changes efficiently. This allows the system to maintain responsiveness and scalability as more votes come in, providing a smooth user experience.

Minimizing the number of steps for registration might also have in this regard, single sign-on (SSO) is a possible consideration for minimization.

Mobile responsiveness is also important for ease-of-use. The application needs to be able to work on desktop and mobile so a user can connect however they need.

Trust is often influenced by ease of use [17] but security is a foundational element in building trust. Ease of use will lead to adoption of the application and promote participation.

Complexity and ease of use are critical factors when designing systems, especially those involving sensitive processes like voting or data collection. From the findings of various studies, including the present one, ease of use directly impacts user trust and adoption of technology [19, 20].

In this study, perceived ease of use was one of the key variables tested, and its importance in influencing trustworthiness and adoption cannot be overstated. In voting systems or other high-stakes applications, if a system is too complex, users are less likely to engage with it. A cumbersome login or voting process could frustrate voters, discourage participation, or lead to errors in voting.

The study utilized a Likert scale 7 point scale to gauge user opinions on ease of use, and the results indicated that systems perceived as easier to use led to higher trust and engagement. This aligns with existing literature on human-computer interaction, which suggests that minimizing the steps required to perform critical actions, like logging in or casting a vote, greatly improves user satisfaction and participation.

VII. Testing

Testing is a crucial phase for ensuring the integrity of any system, especially a voting system where accuracy and transparency are paramount. To ensure trust and reliability, the design planning stages must include agile testing steps at various intervals. These steps help identify vulnerabilities early, reduce bugs, and improve the overall stability of the voting system.

1. **Test Planning and Design**:
    - Before development begins, a test plan should outline all the scenarios to be validated, including edge cases like vote tampering or unauthorized access attempts. This plan should include unit testing, integration **testing**, and end-to-end testing.
    - The system should simulate real-world voting processes and edge cases, ensuring the system remains lossless (i.e., no data loss) and transparent.
2. **Agile Testing / Test Driven Development**:
    - Continuous testing during each sprint ensures that functionality is validated as it's developed. This includes validating that votes are cast and recorded correctly while ensuring that the voting system remains secure.
    - Agile testing allows for early detection of potential integrity issues, such as if the system incorrectly associates users with votes or fails to de-identify data as expected.
3. **De-identification and Re-identification**:
    - To preserve privacy, the system must be able to de-identify voter data. This means the system should separate voter identities from their voting choices while still allowing verification that a vote was cast.
    - The process of re-identification should only be applied where necessary to confirm a user has participated in a vote, without revealing their vote's content. This could involve creating an audit trail that verifies participation while keeping vote content confidential.
4. **Integrity vs. Non-Repudiation**:
    - Ensuring *integrity* in the voting system means that each vote cast is recorded as intended, without the risk of modification or deletion. We have covered several forms of authentication to verify integrity.
    - Non-repudiation is essential for auditing purposes, allowing the system to prove that a particular user participated in the vote. The challenge is to do this while preserving voter anonymity.
    - Solutions might involve implementing blockchain or zero-knowledge proofs to allow for audit trails without revealing individual vote preferences.
5. **Fraud Prevention**:
    - Testing should also focus on fraud detection mechanisms. This includes ensuring there are

safeguards against multiple voting, vote tampering, or unauthorized access.

- Simulating various attack scenarios (e.g., DDoS attacks, insider threats) during testing helps identify potential vulnerabilities. It's also essential to conduct penetration testing to assess the system's resistance to external threats.

6. **Ranked-Choice Voting (RCV) Calculation**:
   - Testing should ensure that the ranked-choice voting system is correctly implemented and follows the intended logic. This involves:
     - Creating multiple scenarios with different vote rankings to validate that the system correctly calculates results based on the RCV algorithm.
     - Simulating large datasets and testing how the system handles the rounds of vote redistribution, ensuring the algorithm's accuracy.
   - It may also be beneficial to run comparative tests with manual calculations to confirm that the system's automated process is functioning as expected.
7. **User Testing**:
   - **Usability testing** ensures that end-users (voters) can navigate the system easily and cast their votes without confusion.
   - During these tests, it's important to gather feedback on how voters experience the de-identification process, whether they feel their privacy is maintained, and how easy it is to verify that their vote has been cast without fear of exposure.
8. **Auditing and Documentation**:
   - As the system undergoes testing, all actions and outcomes should be documented. This allows an auditor to review the system's behavior to ensure it follows the defined rules without allowing tampering or fraud.
   - Any discrepancies or bugs uncovered during testing should be logged and addressed immediately. These logs will be essential for the post-launch a**udit** process, verifying that the system behaves as expected during live elections.

By incorporating these testing practices, the system will be better equipped to handle real-world scenarios, protect voter privacy, and maintain the integrity of the voting process.

VIII. Conclusions

Testing is a crucial phase for ensuring the integrity of any system, especially in voting systems where trust and transparency are paramount. The conclusion we can draw from this research is that while there are many sophisticated methods for authentication and privacy by design, a more simplistic and trustworthy approach often proves to be the most effective.

Ease of use plays a critical role in instilling confidence in users. When users can interact with the system effortlessly, they are more likely to trust it and participate fully. Among the various authentication methods explored, FIDO2 stands out as the most promising. FIDO2 allows users to log in **once** and then verify their identity through a platform-specific authentication method, such as a PIN or fingerprint, making the process essentially passwordless. If FIDO2 can be incorporated with SSO and mobile and desktop responsiveness we can maximize ease -of-use.

This method not only simplifies the authentication process but also enhances security, as it reduces the reliance on passwords, which are often vulnerable to phishing or brute-force attacks. Also this could help with scalability because there would be fewer need for password reset. The only complication with this approach is that users must be assigned access privileges when their accounts are created. This requires a mechanism for securely generating and distributing usernames to individuals, ensuring they can access the system appropriately.

Using group policy administrators could simplify the assignment of user roles and thus administer access control. This can streamline security and reduce unauthorized access.

In conclusion, while test-driven development and advanced authentication methods contribute to security, the user experience remains a central factor. Combining FIDO2 with a well-structured user management process could provide a balance of security, simplicity, and trustworthiness—essential features for a robust voting system.

IX. Acknowledgements

Testing is a crucial phase for ensuring the integrity of any system, especially a voting system where I would like to acknowledge the dedication of my professors at NKU for their endeavors to make the world a better place.

References

[1] Zhang. Yuxian, Li Yi, Fang Li, Chen Ping, and Dong, Xinghua. "Privacy-protected Electronic Voting System Based on Blockchin and Trusted Execution Environment", IEEE 5th Intkernational Conference on Computer and Commuinications on 1252-1257 De.c 2019

[2] Y. Zhang, Y. Li, L. Fang, P. Chen, and X. Dong, "Privacy-protected Electronic Voting System Based on Blockchain and Trusted Execution Environment," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, Chengdu, China, Dec. 2019, pp. 1252-1257. doi: 10.1109/ICCC47050.2019.9064387.

[3] I V. Neziri, I. Shabani, R. Dervishi, and B. Rexha, "Assuring Anonymity and Privacy in Electronic Voting with Distributed Technologies Based on Blockchain," *Applied Sciences*, vol. 12, no. 11, pp. 5477, May 2022.

Available: https://doi.org/10.3390/app12115477 from https://www.mdpi.com/2076-3417/12/11/5477

[4] R. Anderson, "Privacy Technology: The Long Road Ahead," in *Communications of the ACM*, vol. 53, no. 6, pp. 86-95, June 2010. doi: 10.1145/1743546.1743568.R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] B. Ahn, "Implementation and Early Adoption of an Ethereum-Based Electronic Voting System for the Prevention of Fraudulent Voting," *Sustainability*, vol. 14, no. 5, pp. N.PAG, March 2022. doi: 10.3390/su14052917. From https://www.mdpi.com/2071-1050/14/5/2917

[6] FairVote, "Multi-winner Ranked Choice Voting Example," [Online]. Available: https://fairvote.org/archives/multi_winner_rcv_example/. [Accessed: 08-Sep-2024].

[7] WebAuthn.io, "WebAuthn Demo," [Online]. Available: https://webauthn.io/. [Accessed: 08-Sep-2024].

[8] Django REST Framework, "Authentication," [Online]. Available: https://www.django-rest-framework.org/api-guide/authentication/. [Accessed: 27-Sep-2024].

[9] Ruxian, Zhang, Zhang Zhaoju, and Di Hong. "Easy-to-Implement Campus Electronic Anonymous Voting System." *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020.T

[10] Y. Wei, “The Improvement of FOO Protocol and Its Application in Electronic Voting System,” Wuhan University of Technology, Wuhan, China, 2009.

[11] Python Software Foundation, "uuid — UUID objects according to RFC 4122," [Online]. Available: https://docs.python.org/3/library/uuid.html. [Accessed: 27-Sep-2024].

[12] Kulyk, Oksana, et al. "Electronic voting with fully distributed trust and maximized flexibility regarding ballot design." *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*. IEEE, 2014.

[13] H. Abdullah, "A Structural and Navigational Method for Integrated Organizational Information Privacy Protection," 2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), Durban, South Africa, 2018, pp. 1-9, doi: 10.1109/ICABCD.2018.8465138. keywords: {Privacy;Data privacy;Organizations;Risk management;Navigation;information privacy;organizational information privacy;Governance;Risk and Compliance (GRC)},

[14] D. Pelkola, "A Framework for Managing Privacy-Enhancing Technology," in IEEE Software, vol. 29, no. 3, pp. 45-49, May-June 2012, doi: 10.1109/MS.2012.47.

[15] Huang, Jun, et al. “The Application of the Blockchain Technology in Voting Systems A Review.” *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, 2021, pp. 1–28, https://doi.org/10.1145/3439725.

[16] E. McDonnell, "Preferred Polls: A Platform for Ranked-Choice Voting," 2024, Available: https://preferred-polls.vercel.app/.

[17] M. T. Nuseir, A. I. Aljumah and G. A. El Refae, "Trust in Adoption of Internet of Things: Role of Perceived Ease of Use and Security," *2022 International Arab Conference on Information Technology (ACIT)*, Abu Dhabi, United Arab Emirates, 2022, pp. 1-7, doi: 10.1109/ACIT57182.2022.9994207. keywords: {Industries;Security;Internet of Things;Information technology;Internet of things;security;integrity;hospitality;UAE},

[18] V. Sliusar, A. Fyodorov, A. Volkov, P. Fyodorov and V. Pascari, "Blockchain Technology Application for Electronic Voting Systems," *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, St. Petersburg, Moscow, Russia, 2021, pp. 2257-2261, doi: 10.1109/ElConRus51938.2021.9396400.

[19] S. A. Nikou and A. A. Economides, "Factors that influence behavioral intention to use mobile-based assessment: A STEM teachers ’ perspective," British Journal of Educational Technology, vol. 50, pp. 587-600, 2019.

[20] N. Wilson, "The impact of perceived usefulness and perceived ease-of-use toward repurchase intention in the Indonesian e commerce industry," Jurnal Manajemen Indonesia, vol. 19, pp. 241-249, 2019.

[21] FIDO Alliance, *Enterprise Adoption Best Practices: Federation and FIDO*, FIDO Alliance. [Online]. Available: https://fidoalliance.org/wp-content/uploads/Enterprise_Adoption_Best_Practices_Federation_FIDO_Alliance.pdf. [Accessed: 02-Oct-2024].