# A Practical Secret Voting Scheme for Large Scale Elections

Bucataru Cristian, Ciulin Alexandru

January 20, 2024

**Abstract**

This report details the implementation of the system described in "A Practical Secret Voting Scheme for Large Scale Elections," adapted to facilitate anonymous teacher evaluations by students. The core of this implementation revolves around ensuring voter anonymity, vote integrity, and ease of use, making it suitable for academic environments. Utilizing a blind signature technique for encryption, the system allows students to securely and confidentially evaluate their teachers. Adjustments were made to the original scheme to cater to the specific needs of educational settings, focusing on scalability for smaller-scale elections and data integration through the use of a file format providing the votes. This adaptation demonstrates the flexibility and potential of the original voting scheme for various contexts, maintaining its foundational principles of security and confidentiality.

**Keywords:** e-voting, encryption, anonymous, multi-sign process.

## 1 Introduction

### 1.1 Context and Importance

With the rapid evolution of digital technologies, electronic voting systems (EVS) are increasingly becoming a focal point in modernizing voting processes. Their ability to enhance security, ensure transparency, and improve the accuracy and reliability of elections makes them an invaluable asset in the democratic process.

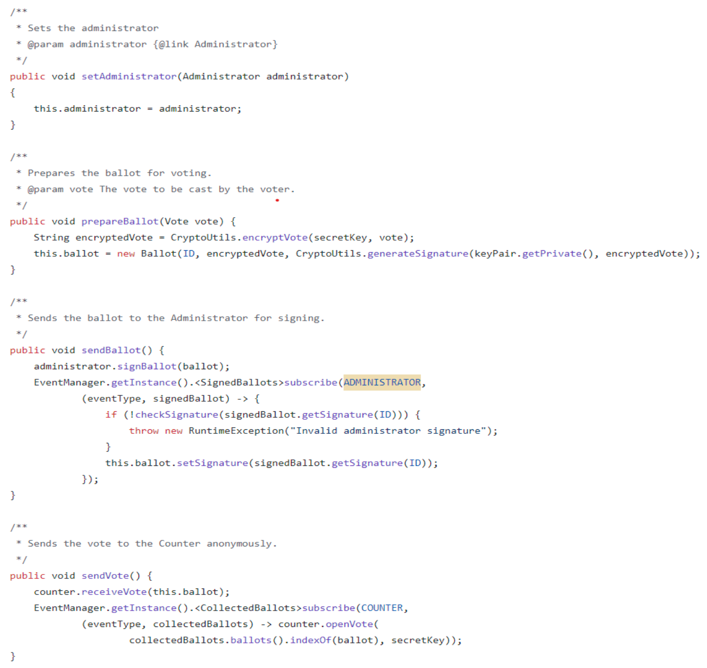
### 1.2 System Overview

Our system introduces an innovative approach to anonymous voting, particularly in educational settings for teacher evaluations. By utilizing a blind signature technique for encryption and ensuring anonymity through a multi-layered process, our system stands out in balancing confidentiality with integrity and efficiency.

## 2 Solution

### 2.1 Voter

The voter's interface is designed for simplicity and security. Upon accessing the system, voters complete an evaluation form (ballot) that is encrypted using a sophisticated blind signature technique. This ensures that their anonymity is preserved throughout the voting process.

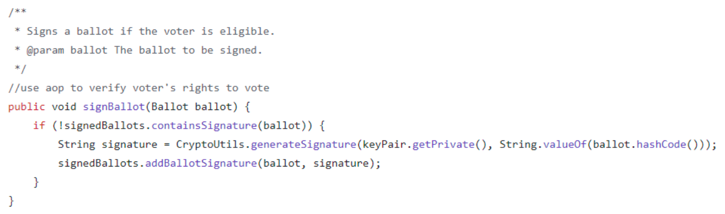


```java
/**
 * Sets the administrator
 * @param administrator {@link Administrator}
 */
public void setAdministrator(Administrator administrator)
{
    this.administrator = administrator;
}

/**
 * Prepares the ballot for voting.
 * @param vote The vote to be cast by the voter.
 */
public void prepareBallot(Vote vote) {
    String encryptedVote = CryptoUtils.encryptVote(secretKey, vote);
    this.ballot = new Ballot(ID, encryptedVote, CryptoUtils.generateSignature(keyPair.getPrivate(), encryptedVote));
}

/**
 * Sends the ballot to the Administrator for signing.
 */
public void sendBallot() {
    administrator.signBallot(ballot);
    EventManager.getInstance().<SignedBallots>subscribe(ADMINISTRATOR,
            (eventType, signedBallot) -> {
                if (!checkSignature(signedBallot.getSignature(ID))) {
                    throw new RuntimeException("Invalid administrator signature");
                }
                this.ballot.setSignature(signedBallot.getSignature(ID));
            });
}

/**
 * Sends the vote to the Counter anonymously.
 */
public void sendVote() {
    counter.receiveVote(this.ballot);
    EventManager.getInstance().<CollectedBallots>subscribe(COUNTER,
            (eventType, collectedBallots) -> counter.openVote(
                    collectedBallots.ballots().indexOf(ballot), secretKey));
}
```

Figure 1: Voter functions

## 2.2 Administrator

The administrator plays a pivotal role in verifying voter eligibility. Once a voter submits their encrypted ballot, the administrator validates it without gaining access to the content, thereby maintaining the confidentiality of the vote. The administrator's digital signature on the ballot is crucial for the counting process.

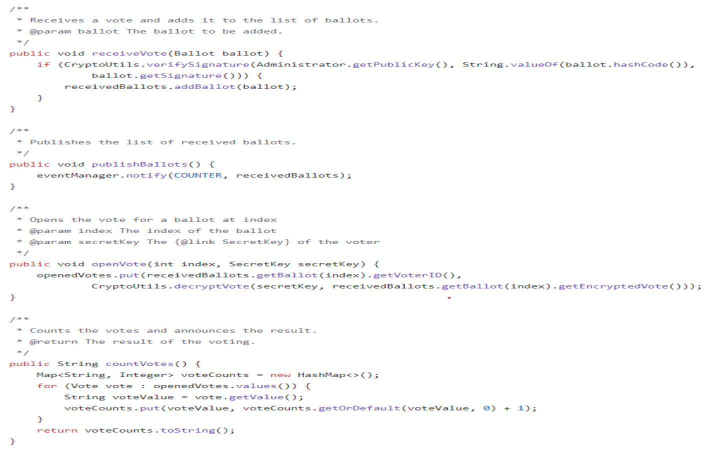


```
/**
 * Signs a ballot if the voter is eligible.
 * @param ballot The ballot to be signed.
 */
//use aop to verify voter's rights to vote
public void signBallot(Ballot ballot) {
    if (!signedBallots.containsSignature(ballot)) {
        String signature = CryptoUtils.generateSignature(keyPair.getPrivate(), String.valueOf(ballot.hashCode()));
        signedBallots.addBallotSignature(ballot, signature);
    }
}
```

Figure 2: Administrator functions

## 2.3 Counter

The counter's function is to collect, decrypt, and tally the votes. After collecting the signed encrypted ballots, the counter waits for voters to anonymously submit their decryption keys. This process guarantees that the counter can tally the votes without compromising voter anonymity.

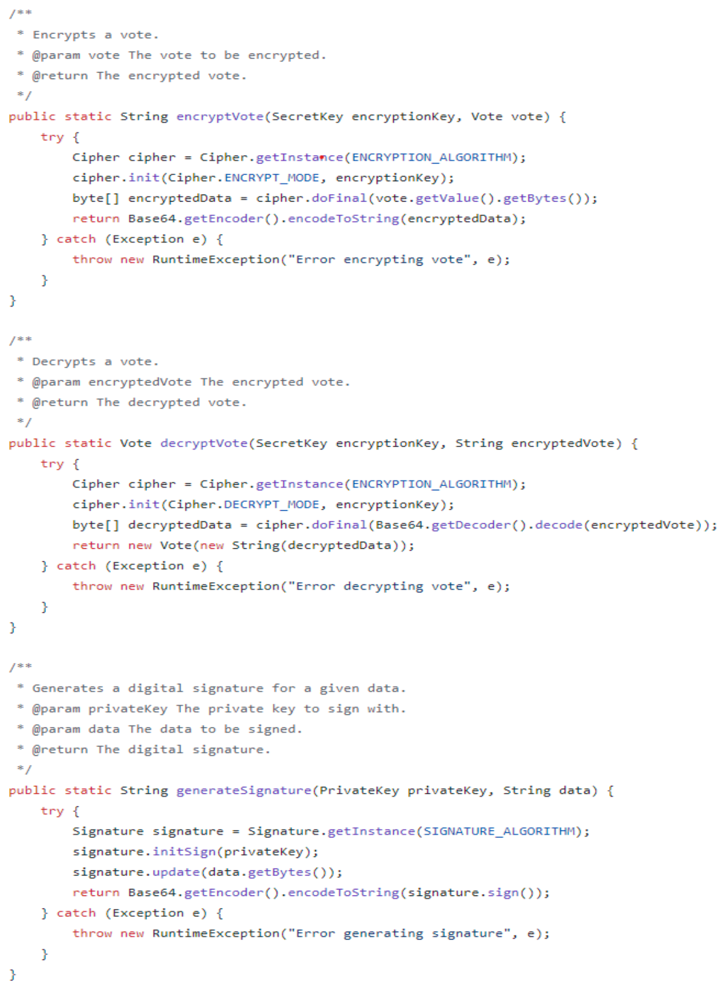


```
/**
 * Receives a vote and adds it to the list of ballots.
 * @param ballot The ballot to be added.
 */
public void receiveVote(Ballot ballot) {
    if (CryptoUtils.verifySignature(Administrator.getPublicKey(), String.valueOf(ballot.hashCode()),
            ballot.getSignature())) {
        receivedBallots.addBallot(ballot);
    }
}

/**
 * Publishes the list of received ballots.
 */
public void publishBallots() {
    eventManager.notify(COUNTER, receivedBallots);
}

/**
 * Opens the vote for a ballot at index
 * @param index The index of the ballot
 * @param secretKey The {@link SecretKey} of the voter
 */
public void openVote(int index, SecretKey secretKey) {
    openedVotes.put(receivedBallots.getBallot(index).getVoterID(),
            CryptoUtils.decryptVote(secretKey, receivedBallots.getBallot(index).getEncryptedVote()));
}

/**
 * Counts the votes and announces the result.
 * @return The result of the voting.
 */
public String countVotes() {
    Map<String, Integer> voteCounts = new HashMap<>();
    for (Vote vote : openedVotes.values()) {
        String voteValue = vote.getValue();
        voteCounts.put(voteValue, voteCounts.getOrDefault(voteValue, 0) + 1);
    }
    return voteCounts.toString();
}
```

Figure 3: Counter functions

## 2.4 CryptoUtils

This class provides some core cryptographic utilities used in the voting system, like generating AES and RSA keys, encrypting/decrypting votes with AES, and generating/verifying RSA signatures. The main methods handle the high-level crypto operations while the private helpers do the lower-level validation and initialization work.



```java
/**
 * Encrypts a vote.
 * @param vote The vote to be encrypted.
 * @return The encrypted vote.
 */
public static String encryptVote(SecretKey encryptionKey, Vote vote) {
    try {
        Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, encryptionKey);
        byte[] encryptedData = cipher.doFinal(vote.getValue().getBytes());
        return Base64.getEncoder().encodeToString(encryptedData);
    } catch (Exception e) {
        throw new RuntimeException("Error encrypting vote", e);
    }
}

/**
 * Decrypts a vote.
 * @param encryptedVote The encrypted vote.
 * @return The decrypted vote.
 */
public static Vote decryptVote(SecretKey encryptionKey, String encryptedVote) {
    try {
        Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, encryptionKey);
        byte[] decryptedData = cipher.doFinal(Base64.getDecoder().decode(encryptedVote));
        return new Vote(new String(decryptedData));
    } catch (Exception e) {
        throw new RuntimeException("Error decrypting vote", e);
    }
}

/**
 * Generates a digital signature for a given data.
 * @param privateKey The private key to sign with.
 * @param data The data to be signed.
 * @return The digital signature.
 */
public static String generateSignature(PrivateKey privateKey, String data) {
    try {
        Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM);
        signature.initSign(privateKey);
        signature.update(data.getBytes());
        return Base64.getEncoder().encodeToString(signature.sign());
    } catch (Exception e) {
        throw new RuntimeException("Error generating signature", e);
    }
}
```

Figure 4: CryptoUtils functions

### 2.5 AI Generated Test Cases

To validate the system's robustness, we employed AI to generate diverse test cases that simulate real-world voting scenarios. These test cases were instrumental in identifying potential vulnerabilities and ensuring the system's resilience against various attack vectors.

## 3 Results and Evaluation

### 3.1 Successful Scenarios

Our testing included several scenarios where the system functioned as intended. In these tests, votes were cast, encrypted, signed, and counted accurately. The system efficiently handled multiple simultaneous votes, demonstrating its capability to manage real-world voting scales.

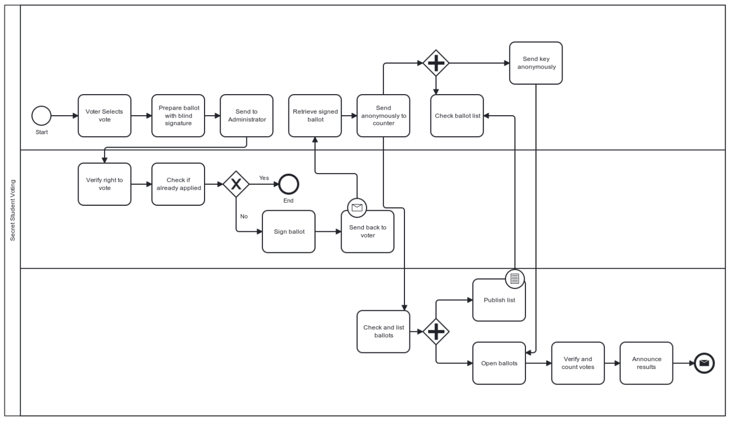


Figure 5: BPMN diagram

### 3.2 Security Scenario

An important part of our evaluation was testing the system's response to potential security threats. In one scenario, an attempt was made to manipulate the votes. The system successfully detected and prevented this, showcasing its robust security measures. This test confirmed that our e-voting scheme can effectively safeguard against common vulnerabilities in electronic voting.

## 4 Comparison with Other Solutions

In comparing our e-voting system with others, such as those detailed in [Tej+21] and [NBK15], several differences become apparent. Our system stands out in its unique application for educational settings, specifically for anonymous teacher evaluations. While other systems primarily target general elections, our solution is tailored for smaller, more focused voting scenarios.

A key advantage of our system is its simplicity and ease of use, making it accessible for everyday academic use. Unlike some complex systems that require extensive setup and maintenance, our system can be deployed quickly and managed with minimal technical expertise.

Moreover, our approach to maintaining voter anonymity, through the use of blind signature encryption and anonymous key submission, provides a level of security and privacy not always present in other solutions. This is especially important in the context of teacher evaluations, where ensuring the confidentiality of student feedback is crucial.

## 5 Future Work

While our e-voting system demonstrates promising results, there are several avenues for future development. One area of potential is the integration of innovative input methods, such as biometric authentication or voice recognition. These technologies could make the voting process more seamless and add an extra layer of security.

Another direction for future work is the exploration of blockchain technology for voter authentication and record-keeping. Blockchain could offer a decentralized and tamper-proof method for storing votes, further enhancing the system's integrity.

Scalability is also a key consideration for future iterations. As the system is currently optimized for smaller-scale elections, like teacher evaluations, expanding its capability to handle larger-scale elections would increase its applicability.

Finally, continuous improvement of the system's codebase is essential. Future efforts should include regular code reviews, security audits, and updates to stay aligned with best practices in software development and emerging security threats.

## 6 Conclusions

In this report, we presented an e-voting system designed specifically for anonymous teacher evaluations in educational settings. Our system showcases several strengths, including its emphasis on voter anonymity, ease of use, and robust security measures against potential threats. The implementation of blind signature encryption and anonymous key submission plays a critical role in ensuring the confidentiality and integrity of votes.

Our system's application in a focused context like teacher evaluations demonstrates its versatility and potential to adapt to different voting scenarios. While the current version is optimized for smaller-scale elections, its foundational principles and technologies have broader applications.

One notable aspect is the simplicity of the system, making it accessible and manageable even with limited technical resources. This characteristic is particularly valuable in educational environments where resources may be constrained.

Looking forward, the integration of advanced technologies such as biometrics, and blockchain could further enhance the system's security and scalability. Continuous refinement and adherence to software development best practices will be crucial in maintaining its effectiveness and reliability.

Overall, our e-voting system represents a significant step forward in the domain of electronic voting, combining practicality with advanced security measures. It not only addresses current needs in educational settings but also lays the groundwork for more sophisticated and wide-ranging applications in the future.

## References

[NBK15] Divya G. Nair, V. P. Binu, and G. Santhosh Kumar. "An Improved E-voting scheme using Secret Sharing based Secure Multiparty Computation". In: *CoRR* abs/1502.07469 (2015). arXiv: 1502.07469. URL: http://arxiv.org/abs/1502.07469.

[Tej+21] Marino Tejedor-Romero et al. "Distributed Remote E-Voting System Based on Shamir&rsquo;s Secret Sharing Scheme". In: *Electronics* 10.24 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10243075. URL: https://www.mdpi.com/2079-9292/10/24/3075.