A new secure Internet voting protocol using Java Card 3 technology and Java information flow concept

Mostafa Mohammadpourfard¹, Mohammad Ali Doostari¹*, Mohammad Bagher Ghaznavi Ghoushchi² and Nafiseh Shakiba¹

¹ Department of IT and Computer Engineering, Shahed University, Tehran, Iran
² School of Engineering, Shahed University, Tehran, Iran

ABSTRACT

Recently, there has been a spate of interest in Internet voting systems because of advantages such as participation, efficiency, accuracy, and transparency. However, challenges for having a secure i-voting system are considerable. Unless these systems are designed and implemented carefully, citizens might lose their trust on the whole voting process. This paper introduces a novel online voting protocol, which satisfies the desired security requirements of i-voting as collusion resistance, fairness, coercion/bribery, and secure voting platform. Although Internet voting systems provide convenience for voters by requiring just a PC and an Internet connection, they might be subject to some drawbacks as PCs are very susceptible to malware and sophisticated attacks. To clarify, voter side insecure platform is one of the biggest challenges in Internet voting, which would breach voter’s privacy and also affect the integrity of election. In this paper, we present an alternative to the voters’ insecure PCs. Java Card 3 is the latest version of Java Card, which could be considered as voter’s portable secure Web server. It can obtain an IP address and communicate with other network nodes with hypertext transfer protocol secure (HTTPS). Therefore, regardless of utilizing a trusted device at the client side, end-to-end security is guaranteed. This means that Java Card 3 can resolve challenges, which are posed by insecurity of the vote casting PC. Furthermore, to enhance the security and guarantee the confidentiality and integrity of the data, which are stored in the card during the voting process, we have used Java information flow. An implementation of this protocol is proposed on the basis of Java Card 3 servlet container and Web server technology in which the card and electoral servers communicate on a machine-to-machine basis. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS

electronic voting; Internet voting; cryptography; Java Card 3 technology; JIF; cryptographic protocols

*Correspondence
Mohammad Ali Doostari, Department of IT and Computer Engineering, Shahed University, Tehran, Iran.
E-mail: doostari@shahed.ac.ir

1. INTRODUCTION

With recent worldwide developments in network infrastructures, new e-services have facilitated and paced the interactions with governmental entities. Electronic voting is one of the most interesting novelties among such e-services. Participation, efficiency, accuracy, and transparency are counted as some of the electronic voting advantages [1]. Internet voting—a derivative of e-voting—enables citizens to participate in voting without the need of physical presence at voting stations. Citizens can cast their votes on their PCs through the Internet, and this is considered as the most important achievement of Internet voting.

Internet voting generally consists of five stages as shown in Figure 1 [2]. Each stage has the entities in the voting model and participates in a cryptographic protocol to meet the specific requirements. All the stages follow in sequence, except for the verification stage, which can be used multiple times to enforce the protocols.

Internet voting has two main drawbacks. First, the voter is not in a controlled scope. They can vote from any remote terminal. Hence, coercion/vote buying issue has become one of i-voting system main problems [3]. Second, voter’s PCs are unreliable because of the presence of viruses, botnet networks, Trojan horses, and so on. These are serious obstacles in the utilization of Internet voting systems [4,5]. However, these problems could be solved through our proposed scheme. The main contribution of this paper is the design and implementation of a secure Internet voting system based on FOO [6] through the use of Java Card 3 technology capabilities.

The rest of this paper is organized as follows: In Section 1, the main requirements of Internet voting protocols are described. In Section 2, four leading e-voting protocols are
introduced. In Section 3, the leading challenge in Internet voting, namely client-side insecure platform, is investigated. Section 4, introduces the Java Card 3 technology and Java information flow (JIF). Our proposed scheme is presented in Section 5. Section 6 briefly sketches our implementation. Section 7 provides a security analysis of our proposed scheme. The paper is concluded in Section 8. Finally, we acknowledge the people who have contributed to doing this research and implementing our protocol.

1.1. Internet voting and security requirements

Although e-voting protocols offer many interesting features, their electronic nature poses many security concerns, which should be addressed to guarantee the validity of election results. Many e-voting proposals outline the following requirements for secure e-voting [7-10]:

Anonymity: It is not possible to link a vote to a voter.
Integrity and accuracy: Only valid votes are counted. Furthermore, altering, deleting, and adding votes are not permitted.
Resistance to collusion: Collusion is a critical matter in election. In this research, this issue has been investigated from two novel perspectives:

1. Masquerade: electoral officers can collude to vote instead of the absent eligible voters.
2. Breaching voter’s anonymity: an ideal Internet voting protocol employs two absolutely distinct phases for identification and registration. It shares voter’s information between the voter and the other parties. According to FOO, the Tallier knows a voter’s vote but does not know the voter’s identity. In a different case, the validator knows a voter’s real identity but cannot see his or her vote. The link between a vote and the voter is in the voter’s hand.

Democracy: Only authorized voters can vote. Also, each voter is eligible to vote only once.
Verifiability: Verifiability is expressed in two forms: individual verifiability and public verifiability. In individual verifiability, every voter is able to testify the integrity and accuracy of the tally process. In public verifiability, all voters can verify the accuracy of counting process.
Complaint tracking: One of the most important principles of democracy is complaint tracking. Because all the interactions in Internet voting are online and transparent to the voters, some suspicions may arise, which may affect voters’ trust. A consistent approach to retain voters’ trust is to utilize easy complaint tracking mechanisms to respond to the voters’ post-election complaints.
Bribery resistance and uncoercibility: A voter could not prove to a coercer how he has voted. In other words, voter traceability is not possible as the vote and other voting data are kept secret.

Coercion and bribery are serious threats as for Internet voting, voters can vote from any terminal without an electoral authority’s supervision. In order to avoid coercion and bribery, different solutions are proposed. For example, in electronic voting context, voters are isolated in voting kiosks. In [11-14], the receipt-freeness solution is proposed. However, our proposed solution provides the following criteria:

1. The amount of information offered to each voter is just enough to enable him or her for individual verifiability. In other words, based on this information, the voter can only verify his or her vote accuracy and integrity before and after the election finales.
2. Accordingly, available information to others is just sufficient to enable them for public verifiability, that is, they can only verify the accuracy of the tally, and there is no way to gain access to a given voter’s information (voting information).

Because this system only lets the voter access his or her vote and voting information, it leaves no room for the voter to convince a briber or a coercer. It should be mentioned that it is assumed that the coercer cannot gain physical access to a voter’s identity card or vote on behalf of the voter.

Fairness: There is no way to understand partial tabulation of the counting phase—by either voters or other electoral officers—before the end of the election.
Vote-and-go: Voter participation is not required in the counting phase.
Voter-side’s platform security: Voter platform must be fully secure to guarantee voter and vote anonymity.
Mobility: Voters can cast their votes from any location.
Robustness: Can be discussed from two different viewpoints:

1. System resistance against defect and failure.
2. Malicious voter’s failure in frustrating or disturbing election.
2. EXISTING VOTING PROTOCOLS

Existing e-voting protocols can be classified into three types according to the used cryptographic techniques:

- Mix-nets [15–17],
- Homomorphic encryption [18,19], and
- Blind signatures [20,21,7].

Mix-nets and homomorphic encryption are inefficient for large-scale elections because of the huge computational and communicational costs [1,3]. Regarding blind signature protocols, they do not require any complex computational operators or high communicational cost [3]. Therefore, they are proper for large-scale elections. Our proposed protocol, which is based on FOO, makes use of blind signature to provide anonymity. In the next subsections, FOO protocol and some other protocols, which are derived from FOO, will be reviewed.

2.1. FOO protocol

Fujioka, Okamoto, and Ohta [6] developed a practical voting scheme, known as the FOO protocol. The protocol has served as a basis of many voting protocols and has been the subject of various enhancements [22]. It models the voting process as follows:

1. The voter fills the ballot, encrypts it with a secret key, and blinds the result.
2. The voter then signs the blinded ballot and constructs a message including his or her identity and encrypts it with the validator’s public key.
3. The validator decrypts the message and verifies that the signature belongs to a registered voter who has not voted yet.
4. If the verification is successful, the validator signs the ballot and returns the blinded ballot to the voter.
5. The voter unblinds the ballot, encrypts it with the tallier public key, and sends the signed encrypted ballot to the tallier (through some anonymous channels).
6. The tallier verifies the validity of the signature on the encrypted ballot; if the signature proves valid, the tallier would place it on a list, which is published at the end of the election.
7. The voters then verify the published list for the availability of their ballots and provide the tallier with the decryption keys necessary to open their ballots.
8. The tallier uses these keys to decrypt the ballots and add the votes to the election tally.
9. After the election, the tallier publishes the decryption keys along with the encrypted ballots so that the voters may verify the election results independently.

2.2. Sensus protocol

While the Sensus protocol [23] is based on the FOO protocol, the two protocols are different in that the tallier can process votes before an election ends in Sensus protocol. In this protocol, a tallier collects the encrypted ballot and returns a receipt to the voter before the election finales. Then, the voter may submit the decryption key immediately after receiving the receipt. However, as mentioned in [4], this protocol suffers from vulnerabilities such as unfairness and uncoercibility and can only support the individual form of verifiability. Furthermore, Sensus suffers from impersonation. This means, it allows one of entities in the protocol to vote instead of absent but eligible voters.

2.3. Robust electronic voting system protocol

An e-voting system called Evox (this is based on FOO, too) has been introduced in [24]. This protocol allows an administrator server to introduce votes. In [25] the Evox-MA protocol, an evolution of Evox has been proposed. The goal of Evox-MA is to prevent the administrator from adding votes. Robust electronic voting system (REVS) [26] is based on the Evox-MA system, but the system has been modified to be fault tolerant; thus, it assures availability. However, the REVS protocol suffers from uncoercibility and collusion, and also, it only supports individual verifiability.

2.4. Secure e-voting applet system protocol

In SEAS [4], a secure e-voting applet system is proposed. It has addressed the impersonation vulnerability of Sensus protocol, while preserving the limited number of servers of Sensus. But it still suffers from other vulnerabilities such as unfairness, uncoercibility, collusion, and lack of strong verifiability.

It should be mentioned that none of the aforementioned protocols have addressed the voter-side insecure platform. In this paper, a secure and practical Internet voting protocol is presented, which satisfies all the security requirements of an i-voting system.

3. THE INTERNET VOTING LEADING CHALLENGE: CLIENT-SIDE INSECURE PLATFORM

In 2007, Vint Cerf, one of the founders of the Internet, made a shocking claim during a conference held in Davos by the World Economic Forum. Cerf claimed that a quarter of all personal computers around the world, roughly 100 to 150 million computers, connected to the Internet are totally or partially controlled by computer criminals. As with i-voting, it would be possible that these attackers modify the votes or even prevent voters from voting. In
this case, any form of security devised for the overall online voting system, finally, will be defeated by the user’s PC [27]. Accordingly, some solutions have been proposed for better security in online voting [28–31]:

1. In the first approach that is unique to electronic voting (not practical for Internet voting); voters are required to cast the ballot from a private voting booth.

2. In the second approach [32], a combination of public key cryptography, smart cards, and financial transactional IC card reader (FINREAD) terminal readers are employed with the purpose of implementing a trusted computing environment within the voter’s insecure system. According to financial transactional IC card reader framework, security is ensured through three components: a secure display, a secure pin pad, and a card reader authentication function [33]. Although this solution protects all interactions between the card and the terminal, the card and PC interactions remain uncontrolled. The existence of malicious codes on a voter’s PC or the remote control of a voter’s system by hackers can cause vote modification regardless of the effort utilized to secure the card and the terminal interactions.

To guarantee client-side security, we suggest replacing the voter’s PC with a Java Card 3.

3.1. Replacing voters’ platform with Java Card 3 technology

Smart cards are generally used in e-voting applications for three purposes: secure storage, strong authentication, and digital signature [34,35]. Smart cards, however, enjoy some other capabilities especially suited for e-voting applications. In this paper, a solution is offered, which does not force the voter to use his or her insecure PC as a voting module. In addition to serving all prior capabilities provided by a PC, including ease of use, network connectivity and fast processing of cryptographic operations, this alternative solution offers new functionalities and much higher security than the existing solutions. All these requirements are fulfilled in the Connected Edition of Java Card 3 technology (the latest version of Java Card).

4. JAVA CARD TECHNOLOGY

Java Card technology enables programs (applets) written in the Java programming language to run on smart cards and other resource-constrained devices. Because of its small memory footprint, the Java Card platform supports only a carefully chosen, customized subset of the Java language features. This subset includes features that are well suited for writing programs for smart cards and other small devices while preserving the object-oriented capabilities of the Java programming language [36]. The main goals of designing Java card technology are portability and security, that is, data encapsulation, applet firewall, cryptography, and applet [37].

4.1. Java Card 3 technology

Java Card 3 technology is the latest version of Java Card technology. The version 3.0 of the Java Card specification comes in the two following editions: the Classic Edition and the Connected Edition [38]:

- The classic edition is an evolution of the Java Card Platform Version 2.2.2 and supports traditional card applets on more resource-constrained devices.
- The connected edition provides a new virtual machine and an enhanced execution environment with network-oriented features. Applications can be developed as classic card applets requested by application protocol data unit (APDU) commands or as servlets using hypertext transfer protocol secure (HTTPS) to support Web-based communication schemes (Html, Rest, soap, etc.) with the card. The runtime supports volatile objects (garbage collection), multithreading, inter-application communications facilities, persistence, transactions, card management facilities, on card byte code verification, dynamic and flexible firewall at object level, and enhanced security support for high speed interfaces.

The connected platform includes a HTTP server and a servlet container. This enables the development of smart card application with rich Web-based interfaces, as well as integration with the existing Web services. It allows the card to truly act as a node in an Internet protocol (IP) network. High-level architecture of Java Card 3 platform (Connected Edition) is specified in Figure 2a [38,39]. Java Card 3 technology is able to act as a secure personal computer or secure customized Web server on the client side. The new connectivity layers and protocol stack features in Java Card 3 are shown in Figure 2b [40]. The connected edition also offers a proper mechanism for secure communication at the application layer including program code isolation, access control, authentication, granting permission, and role-based security for the client application on the card, network communication security, key management, and secure hosting for Web application [41,42]. Furthermore, Java Card 3 makes the following scenarios possible:

- Developers no longer need to create individual client applications to access the data and resources on the smart card. The only required client interface is an ordinary Web browser [43].
- Smart card applications are now fully functioning transmission control protocol (TCP) based servers. These server applications are Java servlets, and they have a full HTTP(S) stack allowing them to process GET requests, POST requests, headers, cookies, sessions, and so on [43].
The Java Card platform’s Web application container manages the lifecycle of Web applications and network services, over which requests and responses are sent along with security of access to the applications and their resources.

Because the card connects to other network nodes directly by HTTPS (secure sockets layer [SSL]) protocol via high speed interfaces like USB, there is no need to utilize trusted client devices. Therefore, by enabling the Java Smart Card 3 SSL stack, end-to-end security is guaranteed.

In addition to these advantages of Java Card 3, providing end-to-end security is among its prominent advantages that would enhance the Internet voting security significantly. Figure 3 shows the steps and actions involved in an interaction between a typical Java Card 3 and desktop application.

In addition to security and flexibility of Java Card 3, we also intended to make use of Java language capabilities to provide more security for the proposed protocol. Finally, we encountered JIF in our research. According to the available literature, existing results, and personal communication with Danfeng Zhang and Owen Arden, as current developers of JIF, our needs could be fully satisfied using JIF. JIF is used to prevent distrusted entities from accessing card data (stored data in the card during voting process). Utilizing JIF guarantees data integrity and confidentiality. JIF can also be utilized for solving attacks originated from shareable interface (SIO).

4.2. Concept of Java information flow

Java information flow is a security-type programming language that extends Java through supporting control over information flow and access. The primary goal is to prevent improper use of confidential and/or untrusted information.

Java information flow allows two mutually distrusting entities (i.e., entities that mistrust each other) to share classified information [44,45]. The compiler tracks the correspondence between information and policies that restrict the information use, enforcing end-to-end security properties. JIF extends Java by adding labels that express restrictions on information use. For example, the following variable declaration states that not only the variable $x$ is an int but also the information in $x$ is governed by a security policy:

$$\text{Int} \{\text{Alice} \rightarrow \text{Bob}\} x;$$

In this case, which aims to ensure confidentiality, policy says that the information in $x$ is controlled by the
principal Alice and that Alice permits this information to be read by the principal Bob. In JIF, confidentiality is ensured by defining reader policy. By a reader policy, the owner of the policy authorizes some principals to read a given piece of information. A reader policy is written o → r, where the principal o is the owner of the policy and the principal r is the specified reader. As a formal semantics for reader policies, a reader (p, c) function is defined as the set of principals that are potentially authorized to read information according to reader policy c.

Integrity and confidentiality are a well-known dual; integrity policies are defined dually to confidentiality policies. Integrity is achieved by defining writer policy. A writer policy o ← w authorizes the owner to specify the principals that may have influenced (written) the value of the given information. The policy o ← w means that according to the owner o, a principal q could have influenced the value of the information only if q can act for (delegate authority to another principal) the owner o or the specified writer w. The function writers (p, c) is defined as a set of principals of which principal p believes that it can influence information according to writer policy c [44,45]. We use JIF reader and writer function concepts to provide confidentiality and integrity of the stored vote in the card.

4.2.1. Java information flow and shareable interface object.

The object sharing mechanism of Java Card is an issue of information sharing among applets/servlets. Objects in a context are prevented from direct access to objects of other contexts by firewall. In order to realize object sharing among the applets owned by different contexts in the card, an abstract interface shareable is defined in the Java Card specification. Any method declared in the sub-interface of shareable is called shareable interface (SI) method [46]. The instance of the class that implements such interface is called SIO. When an applet/servlet calls the server applet, it creates SIO; other applets/servlets call client applet and can request a reference of this object.

Although the SIO claims to satisfy object sharing by preserving the applet context in isolation, the threat of an attack still remains. To illustrate the implications of this approach, consider an example with three applets/servlets named A, B, and C. It is assumed that the applet/servlet B is a legal client of SI services, provided by the server applet that can get an SIO reference. If the applet/servlet B is malicious, it may transfer the SIO reference to the applet/servlet C by SI. Applet/servlet A shares an object X, which contains the FOO method, with applet B. Applet/servlet B shares an object Y, containing the Bar() method, with applet/servlet C. Now, suppose that the Y.Bar, which is shared with C, invokes the method of object X owned by A and calls the X.Foo. In this situation, C can invoke X.Foo indirectly.

Java information flow is able to solve this security problem by controlling the information flow. The JIF compiler analyzes the information flows within programs and determines if security policies of the confidentiality or information integrity are enforced by the program. For example, consider the following variable declaration and assignments [45]:

1. int (Alice → Bob) X;
2. int {Alice → Bob, Chuck} Y;
3. X = Y; // OK: policy on X is stronger
4. Y = X; // BAD: policy on Y is not as strong as X

The variable Y is declared to be an int, and Alice permits the information in Y to be seen by both Bob and Chuck. The first assignment is secure because it does not let the information in Y be visible to any additional principals. The second assignment is insecure because now Chuck can see the information that was in X, and this is forbidden by the policy on X (unless Alice or Bob happen to trust Chuck, as will be discussed later). The JIF compiler makes these decisions at compile time, as a part of type checking. We can extend this example to SIO problem to avoid this problem by the reader and write policy and information flow control as shown in Figure 4.

5. OUR PROPOSED INTERNET VOTING SCHEME

The greatest challenge for securing the entire voting protocol (process) is the insecurity of client-side PCs that may be infected with many types of programs and malicious codes (or botnet network) [30,31,47,48]. In many of the existing protocols, there is a false assumption that client systems are secure [10,26].

![Figure 4. The role of Java information flow in shareable interface objects.](image-url)
According to Section 4.1, a safer replacement for the voter insecure platform is Java Card 3 (which is personalized and issued before the election). In this protocol the assumption is that every voter has a Java Smart Card 3, and he or she could easily participate in e-voting and Internet voting by using it. Within each voter card, there is a voting servlet that can be used to interact with other units involved in the election in the form of Web services and send messages on the HTTP(S). It should be mentioned that all communications between the parties involved in this protocol are secured via SSL protocol; therefore, regardless of utilizing trusted client devices, end-to-end security between card and other network nodes are accomplished.

### 5.1. The model of the proposed scheme

The presented scheme consists of four phases and four electoral servers along with the voters. The responsibilities of these servers are explained in the succeeding text. The structure of this scheme is illustrated in Figure 5.

- **Voter**: people who are qualified to participate in election activities.
- **Registrant authority**: responsible for registering, authenticating eligible voters, and signing their selected blinded alias identifier (ID).
- **Administrator**: responsible for distributing empty ballots.
- **Validator**: responsible for signing blinded filled ballots provided by the voters.
- **Tallier**: responsible for collecting the votes, delivering receipts to the voters, managing the bulletin ballot, and tallying the election result.
- **Complaint investigator organization**: responsible for tracking post-election complaints.

It should be mentioned that, in the proposed scheme, a public bulletin board, which is integrated with tallier server, is utilized to show voting information before and after the election in order to insure individual and public verifiability.

### 5.2. Notations

Following notations are used in the proposed schemes:

- **RA**: registrant authority
- **A**: administrator
- **V**: validator
- **T**: tallier
- **Aid**: voter alias ID.
- **Ex{m}**: encryption function for message m using key x.
- **Sx[m]**: signature functions for message m using key x.
- **Dx(m)**: decryption functions for message m using key x.
- **(PK-Voter, PRK-Voter)**: voter’s public and private key.
- **(PK-RA, PRK-RA)**: public and private keys of the registration authority.
- **(PK-A, PRK-A)**: public and private keys of the administrator.
- **(PK-V, PRK-V)**: public and private keys of the validator.
- **(PK-T, PRK-T)**: public and private keys of the tallier.
- **(PK-Election, PRK-Election)**: election public key or vote encryption key, which is used by voters to encrypt their votes for confidentiality purposes.
• **PRK-Election**: election private key or vote decryption key, which will be revealed after the election to decrypt the votes.
• **R (m, r)**: blinding function for message m and blind factor r.
• **R⁻¹(m, r)**: unblinding function for message m and blind factor r.

### 5.3. Structure of the proposed schemes

Our proposed scheme, which is based on the model described in Section 5.1, is presented in this section.

#### 5.3.1. Registration phase.

**Step 1**: Registration phase is performed within a few days before the election. In this phase, the voters go to the registrant authority website to be verified as an eligible voter. To this end, the voter sends his or her identity information (which discloses his or her complete name, age, and nationality) to registration authority. It is necessary for the voter, to choose an alias ID (Aid). Then the Aid is blinded by servlet using the blinding function \(R()\), and it is sent to the registration authority to sign up as follows:

\[ X = R(Aid, r) \]

**Step 2**: Before sending identity information and blinded Aid to RA, the voter signs the message with his or her private key and then uses registrant authority public key to encrypt the message and sends it to registrant authority as follows:

\[ Ri = S_{PRK-Voter} [identity information, X] \]

\[ (1) \ V \rightarrow RA : E_{PRK-RA} (I, X) \]

**Step 3**: Having received the aforementioned message, RA decrypts the message and then verifies voter signature. If the voter eligibility to participate in the election is approved, RA will sign X for voter i as \( X_i \) by its private key PRK – RA.

\[ A_i = S_{PRK-RA} [X_i] \]

\[ (2) \ RA \rightarrow V : B_i = E_{PRK-Voter} [A_i] \]

**Step 4**: Having received \( B_i \), the voter decrypts it. In order to get the signature of RA on Aid, the voter removes the blinding factor by the unblinding function \( R^{-1}() \) as follows:

\[ R^{-1}(A_i, r) = S_{PRK-RA} [Aid] = RS – Aid \]

Now, the voter can use this signed Aid (RS – Aid) as an anonymous voting license or token to introduce himself or herself to other electoral authorities as an eligible and registered voter without revealing his or her real identity. Actually, this token completely hides all mapping between the voter and his or her citizenship information.

#### 5.3.2. Administration phase.

**Step 1**: In this phase, first, the voting servlet encrypts both the Aid of voter (Aid) and the signed Aid by registrant authority \(RS – Aid\) with administrator’s public key \(PK – A\), then the voter sends it to the administrator as follows. In order to guarantee channel confidentiality, a random number is generated by the card, and it is sent to the administrator. The random number is used to prevent a channel eavesdropper from gaining access to the voter’s information:

\[ (3) \ V \rightarrow Admin : E_{PK-A} [RS – Aid, Rand] \]

**Step 2**: Having received the message, the administrator decrypts it and verifies the signature of Aid. Then, the administrator prepares an empty ballot \(EB\). Next, the administrator uses its private key to sign \(EB\) and \(RS – Aid\), and sends it back to the voter as follows:

\[ (4) \ Admin \rightarrow V : M_i = S_{PRK-A} [RS – Aid || Rand], \ S_{PRK-A} [EB] \]

Concatenation of \(RS – Aid\) with a voter-defined random number in message 4 will prevent the attacker from gaining the true value of \(S_{PRK-A} [RS – Aid]\) in case of channel eavesdropping. Therefore, he cannot masquerade (impersonate) the voter in next phases.

**Step 3**: After the voter receives the aforementioned message, the servlet verifies the administrator signature and then prepares the ballot to be filled by the voter. The servlet also removes the random part and retrieves \(Prove_i\)—the double signed \(Aid\) by registrant authority and administrator—as follows:

\[ Prove_i = S_{PRK-A} [SPRK-RC [Aid] || RS – Aid] \]

It is worth noting that \(RS – Aid\) is encrypted with the voter’s public key, so only the voter can retrieve \(RS – Aid\) in the registration phase. Furthermore, because a random number is employed in step 1, the masquerade is prevented in administration phase.

#### 5.3.3. Voting phase.

**Step 1**: The voter fills the ballot (FB), then selects a blinding factor \(r_i\), and uses the blinding function \(R()\) to blind his or her vote known as \(BB\).
Then she or he encrypts the message \((BB, Aidi, Prove_i)\) by the validator’s public key and sends it to the validator as follows:

\[
BB = R(FB, r_i)
\]

\(V\rightarrow Validator: \quad E_i = EPK_{-V} \{BB, Aidi, Prove_i\}\)

Because eavesdropping attack is prevented in registration and administration phases, only the voter who possesses both \(Aidi\) and \(Prove\) can produce the message.

**Step 2:** Having received the \(E_i\) the validator decrypts and verifies it and compares \(RS - Aidi\) signature with \(Prove_i\), then uses its private key \(PK - V\) to sign \(BB\) and sends it to the voter as follows:

\[
Validator\rightarrow V: \quad V_i = SPRK_{-V} [BB]
\]

**Step 3:** Having received the message, the voter removes the binding factor by using unbinding function \(R^{-1}(t)\) and attains the validator signature on the filled ballot (SFB) as follows:

\[
R^{-1}(V_i, r_i) = SPRK_{-V} [FB] = SFB
\]

### 5.3.4. Collecting and counting phase.

**Step 1:** In this phase, the voting servlet encrypts the signed filled ballot with the election public key \(PK - Election\) and sends it with \(Prove\) to tallier as follows:

\[
(7) \quad V\rightarrow Tallier: EPK_{-T} \tau \{EPK_{-Election} \{SFB_i\}, Prove_i\}
\]

**Step 2:** Now, in order to prevent bribery and coercion, a seed is generated by the tallier to diversify the \(Prove_i\) (after verifying \(Prove\), signature). Tallier then displays the signed message \(receipt = SPRK_{-T} \tau \{EPK_{-Election} \{SFB_i\}\}\) with diversified \(Prove_i\) on the public bulletin board. It also stores the receipt, the \(Prove_i\) and the seed within the voter card. In this case, because only diversified \(Prove_i\) is displayed on the public bulletin board, traceability of the voter is not possible.

\[
(8) \quad Tallier\rightarrow V: SPRK_{-T} \tau \{EPK_{-Election} \{SFB_i\}, seed, Prove_i\} = receipt
\]

The bulletin board is a subset view of the tallier’s database. Therefore, the bulletin board owner is the tallier, and the computations are done at the tallier side. The bulletin board consists of two buttons: verify me and verify election. The verify me button enables each voter to accomplish individual verifiability, and the verify election enables public verifiability to download the bulletin board before and after the election and compare and decrypt the votes to verify the tally accuracy. In order to individually verify a vote integrity as soon as the voter refers to the public bulletin board with his or her card and presses the verify me button, a HTTP(S) request is sent for the e-voting servlet to fetch the seed and the \(Prove\) from the card. These information elements are stored as sensitive and non-extractable application data and are protected with JIF-based bulletin board reader Policy; That is, no entity except the bulletin board can fetch them from the card. When the diversified \(Prove_i\) is calculated (the diversification algorithm is only known to bulletin board), the voter specific bulletin board entry is highlighted. Because it is assumed that it is only the voter who owns the card and because the seed and \(Prove\) are stored inside the card with JIF-based access condition [read: bulletin board reader Policy, extract and modify (writer policy); never], neither the coercer and briber nor any other unit can access them so they cannot verify individually.

### 5.3.4.1. Applying Java information flow to the proposed protocol.

In the proposed scheme, confidentiality (reading) and integrity (writing) of the stored vote in the card are provided by implementing the JIF concept. As mentioned in Section 4.2, JIF defines readers function and writers function, for information confidentiality and integrity. As it was defined in the preceding sections, by readers, we mean people who are considered eligible by the owner to read the data on C policy. By writers, we mean people who are considered eligible by the information owner to affect information on C policy. Therefore, in addition to the security that Java Card 3 provides for us, this protocol uses the JIF concept and its readers and writers methods to achieve confidentiality and integrity of the stored vote.

In our scheme, a JIF-based writer function states that an authority is authorized to write the vote in the card with a predefined set of {IP address, hardware address, and digital certificate}; this is equivalent to C policy in the JIF. Here, the tallier is authorized to write the vote (receipt), the \(Prove\), and the seed in the card:

- Token = \{IP, Mac address, Digital Certificate\} (C policy)
- Writers = \{Tallier\} + Token

This means that the tallier can write data in the card by presenting his or her token. So, step 8 is modified as follows:

\[
(8) \quad Tallier\rightarrow V: SPRK_{-T} \tau [Token], SPRK_{-T} \tau [Token], SPRK_{-T} \tau \{EPK_{-Election} \{SFB_i\}, Prove_i\} = receipt
\]

A sequence diagram for the mentioned writer policy is shown in Figure 6. It should be noted that the digital certificate of the tallier is checked online based on online certificate status protocol (OCSP). In case a voter complains about the result, an investigator organization by readers function would be activated to read the content of the card and address the complaint:

- Readers = \{complaint investigator\} + Token

This means that the complaint investigator organization (namely its enterprise server) is the sole election authority,
which can satisfy all features defined in the token to trace the voting servlet’s content from the card.

**5.3.4.2. Tallying phase.** Once the voting phase is completed and the election private key (PRK – Election) is revealed, the votes are counted in the presence of candidates and electoral authorities. Then the decrypted content of each vote is added as a new column to the bulletin board. Therefore, every voter can check the accuracy of the election by comparing the information on public bulletin board before and after the election finales. After decryption, the vote may stand:

- Valid with a valid signature of the validator with a valid content.
- Rejected with a valid signature of the validator but an invalid content.
- Invalid without the valid signature of validator.

Polling sequence diagram is shown in Figure 7.

### 6. IMPLEMENTATION REMARKS

NeBeans IDE 7.0.1, ORACLE has been used for implementation. Java Card Web project, which contains Java Card 3 servlet, has been used for the voter side, and Java Web application has been used for other electoral servers. Java Card 3 uses Jetty Web server. Jetty provides a HTTP server, HTTP client, and javax.servlet container.
For communication between card and other Web application, classes, and interfaces such as Connector (javax.microedition.io.Connector), HttpConnection (javax.microedition.io.HttpConnection), and other classes and interfaces have been used in the voter-side card. It should be remarked that these application programming interfaces (APIs) are available in Java Card 3 but not in earlier versions.

For voter-side’s Java Smart Card 3, two servlets have been designed. One is only responsible to provide data for tallier server, that is, to provide seed and Prove, and the other one is in charge of all processes (encrypting, sending, receiving, and so on) in the voter side.

Figure 8 shows the first page, which voter sees when he or she enters the address of voting servlet. The left flowchart in the webpage shows the existing status of voting. By pressing the start protocol button, voting servlet is invoked. Figure 9 shows the step through which the card is interacting with RA. The identity information, which is shown, has been fetched from the card automatically after entering the verification pin or fingerprint template, and it is not changeable by the voter. In this step, the voter chooses an Aid. The chosen Aid is then blinded and encrypted by the card. By pressing the Send Data to Registrant Authority, the blinded-encrypted Aid is sent to the RA server. At RA side, a servlet has been designed, which is responsible for receiving requests from voter’s cards and checking the eligibility of the voter and signing the blinded Aid. Note that the interaction between voter’s
cards and electoral servers (RA, A, V, T) is machine-to-machine based (voters do not need to interfere in this process).

In our implementation, 512-bit RSA has been used for data encrypting/decrypting, and application programming interfaces such as javacard.security.RSAPrivateKey, javacard.security.RSAPublicKey, and javacardx.crypto.Cipher are used for cryptography functions at the voter side. Figure 10 shows the received signed Aid from the RA. Figure 11 shows the second phase, which is the interaction between card and the administration server. In this step, the encrypted signed Aid is sent to A, and after verifying the signature, the administration sends the empty ballot to the card. At the administration server side, there is a servlet responsible for receiving requests from voters and to distributing empty ballots. The structure of the ballot is defined by an XML document, which makes it very flexible; for instance, we may easily implement elections with multiple and nested choices. For simplicity, a very simple ballot has been used. The received ballot is stored in the card to be filled and blinded later in subsequent steps. In order to store a vote, FileConnection interface (javacardx.io.FileConnection) is used.

Figure 12 shows the phase in which the card is interacting with the validator server. The Prove value, which is received...
from the administration server, is shown in this figure. In this step, the voter enters the candidate’s code and name. As mentioned before, a very simple ballot is used in this implementation. By pressing the Send Data To Validator, the filled-blinded ballot along with Provei value and Aid, are sent to the validator server. At the validator server side, there is a servlet that receives the filled-blinded ballot, verifies the signature of Provei, and returns the blindly signed filled ballot to the voter.

Figures 13 and 14 show the last steps of the protocol in which the data are sent to tallier server by pressing Send Data To Tallier button. Figure 14 shows a message to the voter that confirms a successful receipt of the vote. By pressing the Verify My Vote button, the voter is redirected to tallier server’s bulletin board. At tallier side, there are two servlets: one is responsible for receiving signed filled ballot, generating a random string as a seed to diversify Provei, inserting data in database (we have used MySQL database version 5.5.21, Oracle Corporation), and returning receipt, seed, and Provei to be stored inside the voter’s card. Another servlet is responsible for the public as well as individual verifiability.

Figure 15 shows the tallier server’s bulletin board during the election hours. By pressing public verifiability button, as shown in Figure 16, the whole encrypted votes along with diversified Provei would be shown. In this
Figure 14. Screen shots of implementation—ensuring screen, which shows the vote has been collected by the tallier.

Figure 15. Screen shots of implementation—the bulletin board first page.

Figure 16. Screen shots of implementation—bulletin board schema in the collecting phase—the bulletin board schema before pressing Verify Me button.
implementation, for simplicity and comprehensibility purposes, seed is considered as a random 4 character string, which is concatenated with Prove, (from left).

By pressing Verify Me button, a request is sent to the voter’s card and its second servlet to fetch the Prove, and seed from the card. Then, the diversified Prove, is calculated and compared with database, and the related vote is highlighted with green as shown in Figure 17.

After the election, votes are decrypted and the election result (decrypted votes) will be shown as Figure 18. By pressing Verify Me button, a request is sent to the voter’s card, and by fetching seed and Prove, from the card, the voter’s vote is highlighted with green as shown in Figure 19.

It is generally agreed that usability and functionality is a very important aspect of Web-based voting systems [49]. Our implementation shows that our proposed protocol, which is based on Java Card 3, is fully functional and usable.

7. SECURITY EVALUATION

7.1. Anonymity

In any voting mode, it is critical that the anonymity of the voter will not be compromised. We will describe how this feature is met in our proposed protocol.

(1) Anonymity in registration phase: To hide the link between voter’s identity and ballot, the blinded Aid is signed by registration authority. The RA can-
not derive a link between a voter’s identity and the Aid. The only link between Aid and its unblinded value is the blinding factor, which is available to the voter. The entire process is as follows:

\[ \begin{align*}
p, q & : \text{large and distinct primes} \\
N & : pq \\
r & : a \text{ random number which satisfies } \gcd(r, n) \\
\end{align*} \]

(1) \text{Voter} \rightarrow \text{RA} : BA \equiv \text{Aid}^{\text{RA}} \mod N

(2) \text{RA} \rightarrow \text{Voter} : SBA \equiv (BA)^{\text{PRK} \mod \text{RA}} \mod N

Considering that RA is not aware of r, it cannot obtain Aid from BA. Finally, the voter unblinds the received message by removing the blinding factor and obtains the RA signature on \( \text{Aid}(RS \mod N) \).

(3) \text{Voter} \rightarrow \text{Voter} : 
\begin{align*}
&= \text{SA} \equiv \text{SBA} \cdot r^{-1} \mod N \\
&= (\text{Aid}^{\text{RA}} \cdot r^{-1} \mod N \\
&= (\text{Aid}^{\text{PRK} \cdot \text{RA}} \cdot r^{-1} \mod N \\
&= \text{Aid}^{\text{PRK} \cdot \text{RA}} \mod N \equiv \text{SA}
\end{align*}

Since then, the voter uses his or her Aid to vote, instead of the real ID.

(2) Anonymity in other phases: Other electoral servers validate each received message using the following equation:

\[ \text{Aid} \equiv \text{SA}^{\text{PRK} \cdot \text{RA}} \mod N \]

Servers can verify that the voter was certified by RA but no voting authorities can recover the link between the voter’s real identity and his or her Aid. The only way for anonymity violation is IP tracking. To prevent the possibility of IP tracking, we recommend the voters to send votes to the electoral units via an anonymous channel or public proxy servers.

### 7.2. Integrity and accuracy

It is critical that the ballot casted by an eligible voter will not be altered and counted correctly.

(1) Accuracy in voting phase: In this phase, because of blinding and encrypting the ballot \( \text{BB} \) with the validator public key \( E_{\text{PK} \cdot v} \{ \text{BB} \} \); no adversary can alter the ballot. Even if the validator will be corrupted, it cannot modify the ballot because it has been blinded. Also, it should be mentioned that because the voter unblinds and checks the received message from the validator \( \text{SPR}_{v} \{ \text{BB} \} \), every alteration or substitution of the vote would be noticed easily.

(2) Accuracy in casting phase: In this phase, the secret sharing mechanism is applied to each vote. Further information on how to keep the election private key confidential is presented in Section 7.8. Also, by storing the receipt \( \text{SPR}_{v} \{ \text{BB} \} \), \( \text{SPR}_{v} \{ \text{BB} \} \) in the card and providing complaint tracking mechanism, any suspected case as well as the voter’s claims will be addressed by comparing the receipt and the announced vote.

It is noteworthy that in some protocols such as in [4, 6, 50], the voter encrypts the ballot with its public key in order to achieve ballot integrity. Then, at the end of the election or immediately after voting and checking the correctness of his or her vote on the bulletin board, the voter provides his or her private keys to the tapper. Therefore, the fairness as well as the vote-and-go issue can be violated in such approaches. However, the proposed protocol satisfies this feature along with fairness and vote-and-go issues.
7.3. Resistance to collusion

Electoral officers’ collusion is prevented in two levels:

1. **Masquerade (impersonation):** In our proposed scheme, masquerade is prevented in each phase.
   - **Masquerade in registration phase:** In this phase, each voter is identified with personal identification data, which are fetched from his or her own Java Smart Card 3 after voter authentication (by PIN number or preferably biometric templates). Therefore, masquerade would not be possible.
   - **Masquerade in administration phase:** Because all the interactions are encrypted with the voter’s public key $PK_V$, it is hardly possible to eavesdrop the channel to gain $RS_A$. Indeed, without $RS_A$, the attacker cannot produce $EPK_{V} \{RS_A, Rand\}$ (message 3). Therefore, the administrator cannot produce a valid $EB$.
   - **Masquerade in voting and collecting phase:** Because in the administration phase, the $Prove_i$ is protected from eavesdropping by padding the $Prove_i$ with a voter-defined confidential random number shared only with the administrator, only the voter can remove the padding to retrieve the $Prove_i$. Consequently, without a valid $Prove_i$, the attacker cannot produce $E_{PK_{V}} \{BB, Aid, Prove_i\}$ (message 5), hence cannot vote instead of the voter. Figure 20 shows the summary of the aforementioned explanations.

Masquerade may also occur when an authority or attacker misuses its rights to introduce votes in place of eligible but absent voters. Unlike the FOO protocol in which the registration authority may access the voter’s information, this protocol determines the eligibility of the voters through the information read on each card and confirmation of a valid authority signature (checking sod†). An authority or attacker cannot vote in place of the absent voter without having the voter’s card (it does not have a database).

2. **Breaching voter’s anonymity:** As stated earlier, under the collusion between the electoral authorities, the voter cannot be tracked back from the ballot/vote as the voter’s voting information is shared between the electoral authorities and the voter itself. The registrar authority has blinded $Aid$, and other authorities have signed unblinded $Aid, SPRK_{RA}[Aid]$. In our scheme, the voter is the only one who knows the link (blinding factor of $Aid$) between the information delivered to the registration authority and those delivered to other voting authorities.

7.4. Verifiability

**Individual verifiability:** Each voter can verify his or her vote using his or her card. The process is as follows:

1. The voter intends to verify his or her own vote. Therefore, he refers to the bulletin board.
2. Now, the bulletin board content is like Figure 13.
3. By pressing verify me button, a request is sent to the voter by teller for fetching the seed and $SPRK_{A}[SPRK_{RC}[Aid]](Prove_i)$.
4. The teller computes the diversified $Prove_i$, based on his or her algorithm and highlights the vote. Therefore, the voter is able to check whether the system counted his or her vote correctly.

**Public verifiability:** In the proposed protocol, the content of the bulletin board is public, so anyone can observe the content. During the election, the bulletin board shows the encrypted votes $SPRK_{RC}[EPK_{Election}\{SFB_i\}]$ with the corresponding diversified $Prove_i$ as exemplified in Figure 12. After the election, decrypted values will be shown in Figure 13. This means that anyone can check the accuracy of the election by comparing the content of the public bulletin board before and after the election.

---

† Sign of document.
7.5. Complaint tracking

Complaint tracking is another mechanism embedded in our protocol, which responds to all uncertainties and suspicions. Probably, after the counting phase, the content of some decrypted votes are regarded as invalid. These votes will be highlighted on the public bulletin board. As for the decrypted invalid votes, the investigator will publish the voters’ IDs on the public bulletin board and asks these voters to track their votes. This feature easily detects a series of such complaints. Voters may refer to the election’s complaint investigator organization with their card in hand. In this case, the signed and encrypted content of the vote, signed by tallier and stored in the card during the collection phase \( receipt = S_{PRK → T} [E_{PK → Election} \{SFB_i\}] \), is read and compared with the contents of the public bulletin board, and the complaint can be investigated.

7.6. Democracy

In our scheme, only eligible voters can obtain Java Card 3 and participate in voting. In addition, upon receiving message 1 in the registration phase:

\[
I_i = S_{PRK → Voter} [identity\ information, Blinded\ \{Aid\}] \\
V → RA : \quad I_i, Blinded\ \{Aid\}
\]

RA only signs the voter’s blinded Aid after checking the voter’s available identity information and certificate. It means that only eligible voters can obtain signed Aid and then cast their votes. Therefore, the democracy requirement is guaranteed.

7.7. Bribery resistant and uncoercibility

Because it is possible to vote from any remote location in Internet voting, there is no definite solution to absolutely avoid bribery and coercion. Polling-booth based voting systems protect the voter from physical coercion because the pooling-booth is controlled by the government or voting authorities. But there is no cryptographic solution to avoid physical coercion [51]. It is assumed that the coercer has no physical access to the voter’s card. This feature is guaranteed because of the following reasons:

1. Encrypting votes with the election public key \( E_{PK → Election} \{SFB\} \). Therefore, the voters cannot prove their votes till the election finales. As a result, buying/selling of the votes will be postponed to the end of election. After the election, if the briber considers the candidate as the loser, he will have no incentive to pay to the voters. This also leads to voter’s uncertainty to be paid after the election.

2. Requiring a card for verifying the vote. Only the voter who owns a card can identify his or her vote entry on the public bulletin board. Because the briber or coercer does not own the voter’s card, he or she cannot see a particular vote entry on the bulletin board. In our proposed scheme, the diversified \( Prove_i \) is shown on the bulletin board in which the diversification seed and the basic \( Prove_i \) are stored in the card. These sensitive and non-extractable voting data are protected by a JIF-based reader policy and can never be fetched by the briber, coercer, or even the voter. The voter, on the other hand, has no information on or access to the diversified \( Prove_i \) and seed, so he or she cannot convince anyone.

Although our approach dramatically decreases the probability of bribery and coercion, it does not make it impossible. By bribery and coercion in a protocol, we mean that this property is satisfied by the protocol itself, regardless of whether the protocol is implemented as an electronic or Internet voting protocol. So in protocols such as [52], where using a private kiosk is suggested, bribery is not actually solved.

Moreover, recasting mechanism can be easily embedded in the proposed protocol to somehow address the condition when the coercer physically can access to the voter’s card. In this case, the voter can cast another vote after being physically coerced.

7.8. Fairness

In this protocol, all votes are encrypted by the election public key \( E_{PK → Election} \{SFB\} \), and the private key is shared between the authorities and candidates on a confidential basis. This makes it impossible to decrypt and count votes before the end of an election.

7.9. Vote-and-go

Here is a definition of vote-and-go: There would be no need for voter participation in the counting phase. In FOO, each voter encrypts his or her vote with his or her own public key. Once a voter casts the vote, waits until the vote displays on the public board, then the voter can send his or her private key for the tallier either immediately or later after the election deadline. While providing fairness is the protocol responsibility, FOO requires voter participation to guarantee fairness, which means the protocol is fair if all voters send their keys at the end of the election. In FOO, supporting fairness stands against breaching the vote-and-go property. However, our scheme does not require any involvement of the voter in the counting phase.

7.10. Voter-side’s platform security

Because Internet voting allows voters to cast their votes from any remote terminal, the protocol and electoral authorities have no control on a terminal security. Remote terminals (PCs) used by voters may be infected with many types of programs and malicious codes (or botnet network). Voter-side malware poses serious risks to the casted ballot by voters such as credential theft, modification of the
<table>
<thead>
<tr>
<th>Issue</th>
<th>Our scheme</th>
<th>FOO scheme</th>
<th>Sensus scheme</th>
<th>SEAS</th>
<th>REVS scheme</th>
<th>Dini’s scheme</th>
<th>JCJ</th>
<th>Fan’s scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Anonymity</strong></td>
<td>Yes</td>
<td>Yes(^a)</td>
<td>Yes(^b)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>Because of storing important elements in the card, modification/deletion of vote is not possible</td>
<td>Voters are relied upon to verify that their votes were counted</td>
<td>Voters are relied upon to verify that their votes were counted</td>
<td>Yes Voters are relied upon to verify that their votes were counted</td>
<td>Yes</td>
<td>The observer can detect mistakes(^c)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Resistance to collusion</strong></td>
<td>No one can vote instead of absent voter even if electoral units collude with each other</td>
<td>Validator may introduce vote instead of the absent eligible voters</td>
<td>One of the entities involved in the election process can cast votes on behalf of absent eligible and registered users</td>
<td>No (it is vulnerable)</td>
<td>The protocol prevents the administrator from adding votes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Verifiability</strong></td>
<td>Yes</td>
<td>Only individual verifiability is possible</td>
<td>Only individual verifiability is possible</td>
<td>Only individual verifiability is possible</td>
<td>Only individual verifiability is possible</td>
<td>An external observer verifies the tally accuracy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Complaint-pursuing</strong></td>
<td>Any change or suspicious act can be traced</td>
<td>By the information shown on bulletin board, the voter can pursue, but if electoral parties collude, he cannot prove, because nothing is saved on the voter side</td>
<td>Like FOO scheme</td>
<td>Like FOO scheme</td>
<td>If the electoral authorities including commissioner collude to eliminate vote or change it, the voter cannot pursue it</td>
<td>If the observer colludes with the tallier, voter cannot pursue his or her probable complaint</td>
<td>Not considered</td>
<td>Not considered</td>
</tr>
<tr>
<td><strong>Uncoercibility</strong></td>
<td>Highly prevents both bribery and coercion</td>
<td>Voter can maintain its receipt number and pair it with its encrypted ballot in the published list</td>
<td>Like Sensus</td>
<td>Prevent vote buying but not coercion</td>
<td>Highly prevents</td>
<td>It is stated that although it is not theoretically satisfied, the proposed method can largely reduce the possibility of coercibility</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\(^a\) M. Mohammadpourfard et al., Security Comm. Networks 2014 © 2014 John Wiley & Sons, Ltd. DOI: 10.1002/sec

Continued...
<table>
<thead>
<tr>
<th>Issue</th>
<th>Our scheme</th>
<th>FOO scheme</th>
<th>Sensus scheme</th>
<th>SEAS</th>
<th>REVS scheme</th>
<th>Dini’s scheme</th>
<th>JCJ</th>
<th>Fan’s scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fairness</td>
<td>Yes</td>
<td>Yes, but because of dependence on voter participation to guarantee fairness, it violates vote-and-go feature</td>
<td>No</td>
<td>No</td>
<td>Yes, but if commissioner and counter collude, they can know about voting outcome before the end of election</td>
<td>Yes</td>
<td>No</td>
<td>The concept that is assumed for fairness is different from our definition, knowledge of any partial information of the tally result will not affect the final result</td>
</tr>
<tr>
<td>Vote-and-go</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Voter-side’s platform security</td>
<td>Yes</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Not considered</td>
<td>Yes</td>
</tr>
<tr>
<td>Robustness</td>
<td>No voter can disturb the election</td>
<td>No voter can disturb the election</td>
<td>No voter can disturb the election</td>
<td>No voter can disturb the election</td>
<td>Invalid and repeated votes may frustrate the election</td>
<td>Yes</td>
<td>Not considered</td>
<td></td>
</tr>
<tr>
<td>Mobility</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

*REVS, robust electronic voting system; SEAS, secure e-voting applet system.

*a Anonymity must be investigated in each phase. In FOO, Sensus, and REVS, because of using anonymous channel, there is no possibility of tracking IP address, to violate anonymity of the voter. But, in the case of colluding the registrant unit with other units, breeching the anonymity of the voter is possible.

*b But what if the observer and other electoral units collude?
ballot before encryption, sending the ballot to definite destination, and prevention of voting [53]. To address this problem, we have used Java Card 3. Java Card 3 can act as a secure Web server, which directly connects to the Internet to receive and process HTTP(S) requests and send back HTTP(S) responses without requiring a PC. Therefore, regardless of utilizing any trusted client device, end-to-end security is provided between the voter’s card and the electoral servers.

7.11. Robustness

There are two perspectives to robustness. The first perspective is mostly on the implementation model of the protocol, and the second perspective is on the issuance of the protocol.

(1) In order to achieve robustness, each server can use clustering and load balancing methods that can be served by a number of servers operating in an active state as a cluster. Furthermore, the communication channel shall use different independent paths.

(2) Even if a voter aims to disturb the election, there is little, if any, way to do that. One way is to repeat sending ballots to the tallier. However, the tallier verifies the validity of \textit{Prove}, signature and does not allow the same \textit{Prove}, to vote twice. Moreover, once the receipt is issued, the voting servlet can disable the sending ballot option in the servlet.

7.12. Mobility

On the basis of Section 6, our proposed scheme is efficient for Internet voting, and the voters are not restricted to a certain physical location.

8. CONCLUSIONS

In this paper, we proposed a secure voting protocol based on the FOO protocol. Our proposed scheme not only fixes FOO protocol vulnerabilities but also strengthens it and introduces various security properties such as complaint tracking and collusion from two novel perspectives. The protocol addresses dominant challenges of Internet voting protocols, such as collusion, unfairness, bribery, coercion, and most importantly voter-side insecure platforms that would highly affect the voting system. By utilizing Java Card 3 technology, we would be able to secure the client-side platform. The Java Card 3 technology is able to act as a secure Web server and provides end-to-end security for all network interactions with other electoral authorities through enabling the SSL stack on the smart card without the need for utilizing the trusted device at client-side. In addition, to enhance security, JIF concept is utilized for integrity and confidentiality of the stored receipt. Table I compares our scheme with other protocols namely FOO, Sensus, SEAS [4], REVS, Dini [54], JCJ [55], and Fan [8].

We have developed an implementation based on Java Card 3, J2EE, and XML technologies. It should be mentioned that this is not only the first implementation of an Internet voting protocol with Java Card 3 in Connected Edition, but to the best of our knowledge, this implementation is the first instance of using Java Card 3 technology in e-services. The Java Smart Card 3 simplifies end-user interactions with the card because it brings a look and feel to card applications. On the basis of our implementation, all interactions between card and other network nodes are machine-to-machine based.

Considering its capabilities, it is expected that Java Card 3 would be a suitable alternative for client-side computers. Therefore, we consider that with further study and research on Java Card 3 technology, it will provide conditions for more use of this card in Internet voting. Our scheme is more consistent in meeting the voting demands of the future.

ACKNOWLEDGEMENTS

We thank Thierry Violleau, from oracle; Eric Vetillard, from oracle; Nicolas Bousquet, senior engineer from Oberthur technologies; Samia Bouzefrane, associate professor at CNAM (Conservatoire National des Arts et Metiers, an institution dedicated to life-long higher education in Paris) and researcher at the CEDRIC Labo in embedded systems and smart card area in particular; and Vincent GUERIN—from Oberthur Technologies and Pierre RICOURTmobile embedded systems engineer. We really appreciate their help and this research would not be finished without their support. We would also like to thank the anonymous reviewers for their thorough reviews and highly appreciated comments and suggestions.

REFERENCES

A new secure Internet voting protocol


