IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 10, 2019, accepted January 28, 2019, date of publication February 7, 2019, date of current version February 22, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2897252*

# SeVEP: Secure and Verifiable Electronic Polling System

**AMNA QURESHI, DAVID MEGÍAS, (Member, IEEE), AND HELENA RIFÀ-POUS, (Member, IEEE)**

Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC), CYBERCAT-Center for Cybersecurity Research of Catalonia, 08860 Barcelona, Spain

Corresponding author: Amna Qureshi (aqureshi@uoc.edu)

This work was supported by the Spanish Government, in part by the "Ayudas para la excelencia de los equipos de investigación avanzada en ciberseguridad" under Grant INCIBEC-2015-02491, and in part under Grant TIN2014-57364-C2-2-R "SMARTGLACIS."

**ABSTRACT** Electronic polling systems promise benefits to voters such as accessibility and convenience that enable them to cast their votes at any time, from any Internet-connected computing device anywhere in the world. However, unlike traditional paper-based voting systems, an e-polling system introduces several security risks such as privacy of vote, unlinkability of a voter, voter coercion, secrecy of partial election results, verifiability, and poll integrity. The authenticity of a voter is another security concern, i.e., a voter must be identified through an authentication mechanism that prevents voting of unauthorized voters or multiple voting from authorized voters. Another security concern is the manipulation of votes by an infected (e.g., virus, malware, and so on) voting device. Since the voters use their personal devices to cast votes in an unsupervised environment, a malware-hosted device could make unauthorized modifications to the voter's voting choices. Many e-voting systems have been proposed, however, to date, all these schemes either fail to provide all the required security properties or are not practically feasible on light-weight computing devices. In this paper, we present a secure and verifiable polling system, SeVEP, that employs well-known cryptographic primitives to provide vote and voter's privacy, and poll integrity, confirms the identity of voters through a multifactor authentication scheme, enables multiple voting within the allowed polling period, prevents double voting, and achieves verifiability and uncoercibility in the presence of untrusted voting device. The security, performance, and comparative analysis in terms of security properties and cryptographic costs show that SeVEP is secure, verifiable, and practical e-polling system.

**INDEX TERMS** Authentication, efficiency, electronic polling, malware, security, verifiability.

## I. INTRODUCTION

A poll is a measurement tool that enables citizens to express their opinions on various issues ranging from public policies (e.g., health care, immigration, education, etc.) to public affairs (e.g., election campaign, approval of a political party, etc.) and private businesses (brand management, consumer-focused marketing, etc.) by giving their nod of approval or rejection. With a few questions with multiple answers, a poll conducting authority can obtain hundreds or thousands of opinions of potential stakeholders to point it in the right direction. For example, in the United States, the Agriculture department carries out online polls occasionally to find out the opinions of citizens on local issues [1]. Similarly, before national elections, various research organizations carry out opinion polling to gauge voting intentions [2].

Traditionally, polls were conducted face-to-face, which required a citizen's physical presence. With the rise and popularity of the Internet and mobile phones, polls could be conducted remotely. In recent years, a trend towards electronic and Internet polls can be observed. For example, SurveyMonkey [3] is a public-opinion poll that enables people across the world to give their feedback on anything, e.g., a recent SurveyMonkey online poll was conducted in which over 2 million people participated to either approve or disapprove the job of the United State's president [4].

Internet-based polls involve many components including user's registration and authentication, poll setup, polling (selected options chosen by the user are sent from the user's

The associate editor coordinating the review of this manuscript and approving it for publication was Junggab Son.

connected device across the Internet to the relevant polling authorities), tabulation, result publication, auditing, and validation. Since the Internet-based polls involve three different environments (the poll user's computing device such as a smartphone, a tablet, a desktop PC, etc., the Internet, and the polling system), a security attack on any part of the system can lead to an incorrect poll result. These three different environments and the information shared between them are vulnerable to various attacks [5], which must be prevented by the poll conducting administration or authority to provide fair, secure, accurate, and unbiased polling results. Similar to electronic voting (e-voting) systems, Internet-based polls are threatened by exactly the same security attacks, such as unauthenticated voting (a non-eligible voter may cast his/her votes), double voting (an eligible voter may cast multiple votes using his/her polling credentials), voter coercion (a voter is put under pressure or is threatened by a coercer to vote in a particular manner), vote buying (a voter is offered monetary benefits by a vote buyer to vote in a particular way or abstain from voting), vote modification by a voting device that is either controlled by a malicious program (e.g., malware, virus, etc.) or a hacker, who may cause unauthorized and potentially undetected alterations to voter's selected voting choices, theft/forgery of voter's identity (an attacker with access to credentials of an authenticated voter could cast votes using the identities of a legitimate voter), a coalition of malicious participants to alter or eliminate any voter's vote, or cast fake ballots on the behalf of authenticated voter, and disclosure of partial vote tally before the end of the voting period.

In scientific community, little attention has been paid to develop e-polling systems, i.e., low-risk or small-level public-opinion systems, where reasonable level of security, privacy and functionality should be provided to the end user. Instead, considerable amount of research work can be found related to design and implementation of secure, verifiable and practical remote voting systems for national-level or big elections. Though the security and privacy requirements are identical in both e-voting and e-polling systems, the former system provides strong security guarantees than the latter one: the national-level or regional elections require election conducting authorities to provide strong end-to-end security guarantees to voters with minimum trusted entities and minimal security assumptions, and without overly impacting voters with computational and communicational overheads. Whereas, in small-level such as campus elections, the polling authorities provide relaxed security properties to all eligible voters at fair costs with as many trusted entities and security assumptions as desired to provide a secure and practicable system. In the following, we present the research work of the scientific community related to the design and implementation of a secure Internet-based voting system.

Chaum proposed the use of cryptography to design secure elections [6]. This was followed by many voting systems with the aim to improve security and privacy guarantees such as vote integrity and ballot secrecy. For example, Civitas [7] is a coercion-resistant Internet-based voting system, which provides openly verifiable evidence that all the votes are correctly included and accurately tallied, but requires voters to trust their computers for both vote integrity and privacy. However, it does not allow voters to verify that their votes are cast as they intended. Also, the system is not very efficient due to the quadratic overhead. Several more e-voting proposals with prototypes were developed to provide end-to-end verifiability, which includes the following two principal components:

- Cast-as-intended verifiability: It provides a voter with means to make sure that the vote cast by his/her voting device contains the intended voting option and that no changes have been performed.
- Tallied-as-cast verifiability: It allows voters, auditors and third party observers to check that tallied votes correspond to the cast votes. This process can be divided into two phases [8]:
  - Recorded-as-cast: It allows voters to check that their cast votes have been properly recorded in the ballot box.
  - Tallied-as-recorded: It allows voters, auditors and third party observers to verify that the votes published on the public space (such as Bulletin Board) are correctly included in the tally, without knowing how any voter voted.

A number of end-to-end e-voting verifiable schemes have been implemented and deployed in real-world. For example, Remotegrity [9] is an extension of the Scantegrity [10] system (a paper-based voting system), and employs both paper and electronic communications to allow remote voters to detect whether their votes have been tampered with, and to prove that such tampering exists without having to reveal how they have voted. However, Remotegrity does not protect against vote coercion or vote buying. Also, Remotegrity assumes the voter to be honest in order to ensure vote secrecy.

Haenni and Koening [11] proposed a voting system that employs a trusted device for voting over the Internet on an untrusted platform. In the proposed system, the platform is responsible of generating encryptions in the form of barcodes. The voter scans the chosen candidate's barcode through the dedicated trusted hardware device. Later the votes chosen by the voter are transferred to a trusted computer using a smart card or a USB connector. The device is complicated and requires a camera, matrix barcode reader, and uplink to a computer. Also, a built-in camera of the device can record a voter's behavior while he/she scans his/her chosen candidate's barcode. Additionally, the device learns voter's vote.

Another example of end-to-end verifiable systems is Helios [12], which is an open-audit web-based voting system based on *cast-xor-audit* technique and is designed for low-coercion elections. The correctness of Helios elections is guaranteed through a series of audit procedures that the voters can perform to check all steps from the vote casting process to the computation of the election results, thus, providing both

individual and universal verifiability. However, in Helios, the vote integrity and the ballot secrecy are assured under the assumption that the voting environment (i.e., the device used to cast the vote) is trusted for privacy and verifiability. This assumption is unrealistic because a voting device might be controlled by an attacker or host a malicious program.

Adder [13] is a an open-source online voting system, which is designed for both small and large-scale elections, as well as surveys and data collection applications. In order to vote, the voter logs to the system using his/her voting credentials that identify him/her as an eligible voter, and after successful login, obtains the election key. Then, the voter selects his/her candidate, his/her web-browser generates and encrypts his/her vote using the election key, computes and appends a proof of well-formedness. All encrypted votes received by the system are posted on an online bulletin board, where voters can verify they have been correctly recorded. Similar to other Internet-based voting systems [11], [12], Adder is vulnerable to untrusted platform problem, i.e., it is trivially easy for a malicious program on the voter's device to leak voter's vote or modify the vote without voter's knowledge.

In literature, a few remote voting protocols exist that use return codes to provide cast-as-intended verifiability [14]–[18], and a Bulletin Board to provide tallied-as-cast verifiability [19]. In these systems, before the voting phase, a code sheet is sent to a registered voter over a secondary channel (postal mail). These code sheets contain pre-generated return codes and finalization codes. During voting, when the voter selects his/her voting choices and the voting device submits an encrypted vote to the voting server, the voting authorities calculate or retrieve return codes corresponding to the choices of the voter. These codes are returned to the voter, who compares them with the printed pre-generated return codes on his/her code sheet against the selected choices. If the codes match, the vote casting phase is finalized by the voter via finalization codes, which are sent to the server. The underlying problem of these schemes is that they assume the voting device is not compromised (trusted for privacy), and support single vote casting. Our previous work named VSPReP [20] supports multi-vote casting in the presence of untrusted voting devices and provides both cast-as-intended and tallied-as-cast verifiability to the voters. However, VSPReP does not provide secure mechanism to link multiple votes to a single voter, and an authentication method that prevents non-eligible voters from casting ballots. Also, VSPReP does not provide integrity and authenticity of the polling code sheets that are used by the voter during vote-casting phase. A better solution is expected.

Designing a practical remote voting system is a very challenging task as many aspects (security, authenticity, efficiency) have to be considered. For example, some security properties conflict with each other in subtle ways, resulting in a variety of technical challenges, e.g., vote privacy clashes with vote verifiability: if a voter can successfully prove his/her vote to a third party, he/she could easily sell his/her vote or be coerced into voting for a certain candidate or choice. Similarly, to provide key security properties, many remote voting systems employ complex cryptographic methods, which in turn increase the overheads and, thus, reduce the practicality of these schemes. Also, it is crucial to find a trade-off between the security properties and usability. Most verifiable systems require voters to verify their votes through complex cryptographic protocols, which may negatively impact usability [21], [22].

### A. MAIN CONTRIBUTIONS

In this paper, we propose SeVEP, which is based on our previous work on remote polling system named VSPReP [20]. We improve it [20] by designing a polling system that provides flexible polling, device fingerprinting to allow multi-factor authentication for different devices used by the voter, zero-watermarking of polling code sheets, and generation of polling tags. More specifically, the main contributions of this paper are as follows:

- Unlike most of the remote voting schemes, a 3-layered authentication scheme is designed to provide access to an authenticated voter only. The proposed multifactor authentication scheme employs device fingerprinting to recognize the computing device that the voter is utilizing to login. Based on the results of the fingerprinting, the voter is required to input his/her authentication factors (possession, biometric) to establish his/her identity.
- The polling code sheets are watermarked using zero-watermarking scheme to provide integrity and authenticity.
- Polling tags are generated to allow an authenticated voter to cast multiple votes within a permitted polling time.
- Unlike return codes-based protocols [15], [16], [23], SeVEP carries out cast-as-intended verification mechanism in the presence of untrusted voting devices, and supports multiple voting within an allowed polling period such that the voter can vote several times but only the last one is counted to prevent double voting. This cast-as-intended mechanism is designed in a distributed manner, i.e., the computations performed to compute return codes, acknowledgment and confirmation codes are distributed in nature such that all the involved entities (the code generator, six polling code generators, and the printing facility) must collude in order to carry out a successful attack on the voter.
- SeVEP provides recorded-as-cast verifiability, while preserving the privacy of the voter. Also, SeVEP provides tallied-as-recorded verifiability to the voter, who can check whether the random 3-digit code he/she has chosen himself/herself when casting his/her ballot appears in the poll result along with his/her voting choice on the Bulletin Board.
- The detailed security analysis of two main phases (pre-polling and polling) is provided w.r.t cast-as-intended and tallied-as-recorded verifiability, coercion resistance, fairness, poll integrity, resistance against collusion of

voting authorities and voter authentication. The analysis presented in Section IV-A shows that SeVEP provides reasonable security in the presence of an untrusted voting device.
- The experimental results and evaluation of two main phases of SeVEP in terms of computational and cryptographic costs are presented in Section IV-B to show the proposed scheme's feasibility and practicality on light-weight computing devices.
- A comparative analysis of SeVEP with the most relevant state-of-the-art e-voting systems in terms of security properties and cryptographic analysis is provided in Section IV-C.

### B. OUTLINE OF THE PAPER

The rest of this paper is organized as follows. Section II provides the building blocks of the proposed system. In Section III, we discuss the design of our electronic polling system. This section also details the threat model and three phases of the proposed polling system. Security analysis of the threat model is presented in Section IV-A. Also, this section presents a performance analysis in terms of computational costs (Section IV-B1), and cryptographic costs (Section IV-B2). A comparative analysis between SeVEP and other similar e-voting systems in terms of security properties and cryptographic costs is also presented in Section IV-C. Finally, Section V summarizes the conclusions and presents future research issues.

## II. BUILDING BLOCKS

SeVEP employs a distributed ElGamal cryptosystem, a pseudorandom function based on Decisional Diffie-Hellman (DDH), a keyed-hash message authentication code (HMAC), a non-interactive zero knowledge proof (NIZKP), a verifiable mixnet, a digital signature scheme, multifactor authentication, and zero-watermarking. This section presents a brief overview of these building blocks.

### A. DISTRIBUTED ELGAMAL

In a distributed cryptosystem, a set of agents cooperate to perform decryption on encrypted messages so as to provide confidentiality by preventing any single agent from decrypting messages. In SeVEP, distributed ElGamal cryptosystem proposed in [24] is used to provide voters' privacy. Distributed ElGamal cryptosystem is a set of three protocols: key generation (*KeyGen*), encryption (*Enc*), and decryption(*Dec*). In *KeyGen* algorithm, a subgroup $\mathcal{G}_p$ is taken on as input which has a generator $g$ of order $q$ of elements in $\mathbb{Z}_p^*$ (the message space of the cryptosystem), where $p$ and $q$ are two large numbers with $p = 2kq+1$ for some integer constant $k > 0$. *KeyGen* outputs ElGamal public key $y = g^x$ (global and known to all parties), and a secret key $x$ that is shared among $t$ polling organizers $(PO_1, \ldots, PO_t)$ using a polynomial $f$ of degree $l$ over $\mathbb{Z}_q$ such that each polling organizer holds a share $x_i = f(i)$. In *Enc* algorithm, a message $m \in \mathcal{G}_p$, $y$, and a randomly chosen $r \in \mathbb{Z}_q$ are taken as inputs to compute a ciphertext $c$: $c = (c_1, c_2) = (g^r, y^r.m)$. For the decryption of $c$, *Dec* algorithm requires all polling organizers to compute decryption shares $d_i = {c_1}^{x_i}$ to output a plaintext message $m$. To provide verifiability, non-interactive zero-knowledge proofs are computed during *KeyGen* and *Dec* protocols.

### B. PSEUDORANDOM FUNCTION

A pseudo-random function (PRF) is a deterministic-keyed function $F : \mathcal{K}$ x $\mathcal{X} \rightarrow \mathcal{Y}$ (where $\mathcal{K}$ is the set of keys, $\mathcal{X}$ is the domain, and $\mathcal{Y}$ is the range) guaranteeing that a computationally bounded adversary having access to PRF's outputs at chosen points, cannot distinguish between the PRF and a truly random function mapping between the same domain and range as the PRF. In SeVEP, we use a variant of PRF, a key homomorphic PRF ($F_{\text{DDH}}$ based on DDH), proposed by Naor, Pinkas and Reingold [25]. A PRF is key homomorphic if given $F(k_1, m)$ and $F(k_2, m)$, there is a procedure that outputs $F(k_1 \oplus k_2, m)$, where $\oplus$ denotes group operation on $k_1$ and $k_2$. $F_{\text{DDH}}$ is constructed by considering a cyclic group $\mathcal{G}_p$ of order $q$, and a hash function $\mathcal{H}_1 \colon \mathcal{X} \rightarrow \mathcal{G}_p$ modeled as a random oracle. $F_{\text{DDH}}$ is defined as: $F_{\text{DDH}}(k, m) \leftarrow \mathcal{H}_1(m)^k$ with the following homomorphic property,

$$F_{\text{DDH}}(k_1 + k_2, m) = F_{\text{DDH}}(k_1, m) \cdot F_{\text{DDH}}(k_2, m).$$

$F_{\text{DDH}}$ is a secure PRF in the random oracle model assuming the DDH assumption holds in $\mathcal{G}_p$.

### C. HASH MESSAGE AUTHENTICATION CODE

A Message Authentication Code (MAC) is a cryptographic primitive that relies on a pseudorandom function to provide authentication and verification of received messages. A specific type of MAC, the keyed-hash message authentication code (HMAC), is used to provide data integrity and authenticity of the message. HMAC is obtained by using a cryptographic hash function (e.g., MD5, SHA1, and SHA256, etc.) over the data (to be authenticated) in combination with a secret (symmetric) key. The cryptographic strength of a HMAC depends on the properties of the underlying hash function. The polling code sheets generation phase (Section III-D4) and ballot processing phase (Section III-E2) of SeVEP relies on the HMAC algorithm described in [26].

### D. NON-INTERACTIVE ZERO KNOWLEDGE PROOF

A non-interactive zero knowledge proof (NIZKP) is a variant of zero knowledge proof, that does not require an interaction between the prover and the verifier. The prover computes and sends a statement to the verifier, who either accepts or rejects it. NIZKPs can be obtained in the random oracle using Fiat-Shamir heuristic [27]. To provide verifiability in SeVEP, we have used the following proofs to ensure the honesty of the parties involved in different phases of the polling: (1) proof of correct encryption based on Schnorr protocol [28] (polling phase), (2) proof-of-equality of discrete logarithms based on Chaum-Pederson protocol [29] (polling phase), (3) proof of correct decryption of ElGamal ciphertexts ($\pi_{dec_t}$) (mix and

tallying phase), and (4) proof of correct mixing ($\pi_{mix_t}$) of ElGamal encryptions in the mixnet (mix and tallying phase).

### E. VERIFIABLE MIXNET

A verifiable mixnet is used to provide an anonymous and verifiable tally in electronic voting systems. A verifiable mixnet enables a collection of trustworthy servers to take as input an ordered set of ciphertexts $E = E_1, E_2, \ldots, E_N$ generated in a cryptosystem like ElGamal that allows an encrypted message to be re-encrypted using a new randomization value without changing the decryption process. The output is an ordered set of re-randomized encryptions $E' = E'_{\pi(1)}, E'_{\pi(2)}, \ldots, E'_{\pi(N)}$ (where $E'_{\pi(N)}$ is a re-encryption of $E_N$, and $\pi$ is a uniformly random and secret permutation), and non-interactive zero-knowledge proofs $\pi_{mix_t}$ (where $t = 1, \ldots, N$) of correct mixing. Thus, this re-randomized encryption prevents an adversary to determine the link between the output and the input ciphertexts. The link between elements from input and output is only retrieved in case of conspiring mix-nodes. Verifiability is provided by $\pi_{mix_t}$, that is checkable by any party and demonstrates that $E'$ is correctly constructed. The tallying phase (Section III-F) of SeVEP employs the verifiable mixnet proposed by [30].

### F. DIGITAL SIGNATURE

A digital signature scheme (e.g., RSA, DSA) is used to provide data integrity, data origin authentication and non-repudiation. In SeVEP, we have used the RSA signature [31], which is made up of three algorithms, (*Gen*, *Sign*, *Verify*), for generating keys, signing, and verifying signatures, respectively. *Gen* is a key generation algorithm that creates an RSA pubic key $pk$ ($pk = (n, e)$), and a corresponding RSA private key $sk$ ($sk = d$), where $n$ is a product of two large distinct prime numbers $p$ and $q$, $e$ is a public exponent (a randomly generated integer with $1 < e < \phi$, where $\phi = (p-1)(q-1)$), and $d$ is a private unique integer with $1 < d < \phi$. *Sign* is a probabilistic signature algorithm that takes a message $m$ as an input, produces a hash $H$ of $m$, and then computes a signature $S$ on hash value ($H_s$) using $sk$. *Verify* is a deterministic verification algorithm that takes $pk$ and a signature $S$ as inputs to extract hash $H_s$ from $S$. Also, it computes hash on the received message to generate another hash value ($H_v$), and compares it with $H_s$ for verification purposes. If both hashes are identical, $S$ is considered valid, otherwise invalid.

### G. MULTIFACTOR AUTHENTICATION

Multifactor Authentication (MFA) is a security process in which the end user provides two or more authentication factors to establish identity and access control. The key idea of MFA is to sum up the security of two or more factors. These factors include, passwords, representing ''something you know, i.e., knowledge factor'', or physical tokens, such as smart-cards, representing ''something you have, i.e., possession factor'', or biometric traits such as face and gait, representing ''something you are, i.e., inherence factor'', or contextual factors such as location and ambiance. In SeVEP, a three-factor authentication is used to identify and verify remote voters. In the first level of authentication, the user would provide his/her password (a knowledge factor) to gain access to the application. When the login is successful, the user is asked to enter either of these options depending on the voting device: if a voter is using a mobile phone (not a smartphone), he/she is asked to enter one time password (OTP) sent to his/her registered mobile phone via a SMS; or if a voter is using a smartphone, he/she is asked to reconstruct a graphical password via touch-screen; or if a voter is using a desktop computer to cast his/her vote, he/she is asked to reconstruct a graphical password via mouse clicks. If the voter is successful at the second level, he/she is asked to input his/her final factor, i.e., a third factor to complete the authentication process. The third level of authentication is a fusion of voice and keystroke recognition (inherence factors) techniques. This fusion module determines the scores for both types of authentication techniques, and depending on the set threshold criteria, allows the voter to access the polling system. MFA scheme (Section III-D3) in the pre-polling phase of SeVEP employs a graphical password via touch-screen scheme proposed in [32], a graphical password via mouse-clicks scheme proposed in [33], and a multimodal (fusion) scheme proposed in [34].

### H. ZERO-WATERMARKING

Zero-watermarking [35] is a type of digital watermarking that does not actually embed the watermark information into the host data, instead it uses the characteristics or the properties of the original data to construct a zero-watermark. Unlike the traditional watermarking method, the quality of the host data is not degraded as zero-watermarking does not insert a watermark physically into the original content. In SeVEP, we have proposed a zero-watermarking algorithm (Section III-D5) to provide authentication and tampering detection of polling code sheets. The algorithm uses the characteristics of the polling code sheet (PCS) to generate a zero-watermark, which is then registered with the trusted authority, and is used in the extraction algorithm to prove the authenticity and integrity of the PCS. The watermarked PCS has no difference from the original PCS, but it is protected because the constructed zero-watermark has been registered with the trusted authority.

## III. PROPOSED SYSTEM

This section describes the design and functionality of SeVEP. In Section III-A, we describe the parameters used in the SeVEP and the role of each involved entity involved. Section III-B defines the functionality requirements and the security assumptions. An attack model is described for SeVEP in Section III-C. The subsequent sections (Sections III-D, III-E and III-F) describe three key phases of SeVEP: (1) pre-polling; (2) polling; and (3) post-polling. Two phases of SeVEP, i.e., pre-polling and polling phases, are described in detail here as these two processes address the design requirements of any secure and verifiable remote

**TABLE 1. Parameters and Notations.**

| Parameter | Specification | Parameter | Specification |
|---|---|---|---|
| $V_k$ | $k$th voter | $ID_{V_k}$ | Pseudo ID of $V_k$ |
| $N$ | No. of voters | $S$ | No. of computing devices |
| $VD_S$ | Voting device | $t_p$ | Polling period |
| PO | Polling organizer | $t$ | No. of PO |
| CI | Credential Issuer | BB | Bulletin board |
| PS | Polling server | CG | Code generator |
| PCG | Polling code generator | $\mathbb{X}$ | No. of PCG |
| PF | Printing facility | $tag_{Poll}$ | Polling tag |
| PCS | Polling code sheet | $PCS_{ID}$ | ID of PCS |
| $ACK$ | Acknowledgment code | $Confirm$ | Confirmation code |
| $\widetilde{SC_{ij}}(PCG_{\mathbb{X}})$ | Short return codes | $LC_{ij}$ | Long return codes |
| $t_s$ | Time stamp | $FL$ | Poll Flag |
| $z$ | Voting attempts of $V_k$ | $\rho$ | Random permutation key |
| $i$ | No. of poll questions | $Q_i$ | $i$th poll question |
| $j$ | No. of voting options | $v_j$ | $j$th voting option |
| $l$ | No. of pre-generated keys | $r$ | Secret shared between CI and $V_k$ |
| $ballot_{V_k}$ | Ballot cast by $V_k$ | $B_{V_k}$ | Partial ballot of $V_k$ |
| $\gamma_{V_k}$ | 3-digit random number | $\pi_{XX}$ | NIZKPs of SeVEP's cryptographic operations |
| $p, q$ | Safe primes | $\mathcal{G}_p$ | Cyclic group |
| $g$ | Generator of $\mathcal{G}_p$ | $\mathbb{Z}_p^*$ | Message space |
| $(c_1, h_1)$ | Ciphertext of $v_j$ | $(d_1, e_1)$ | Ciphertext of $\gamma_{V_k}$ |
| $F_{\mathrm{DDH}}$ | PRF based on DDH assumption | $WM$ | Watermark |
| $BC$ | Barcode of PCS | $GS$ | Group size |
| $SK$ | Secret key of zero-watermarking | $TP$ | Text partitions of PCS |
| $(K_{pPF}, K_{sPF})$ | Key pair of PF | $(K_{pPS}, K_{sPS})$ | Key pair of PS |
| $(K_{pCG}, K_{sCG})$ | Key pair of CG | $(K_{pPCG}, K_{sPCG})$ | Joint key pair of PCGs |
| $(K_{pPO}, K_{sPO})$ | Joint key pair of POs | $K_{sess_{\mathbb{X}}}$ | Session keys of PCGs |

polling system. The post-polling phase is similar to other voting schemes in the literature that employ mixnets to preserve anonymity of votes.

### A. SYSTEM PARAMETERS AND ENTITIES

In this section, system parameters and a description of each entity of the system are provided.

#### 1) SYSTEM PARAMETERS

Table 1 describes the relevant terms and parameters used in SeVEP to benefit our readers.

#### 2) SYSTEM ENTITIES

Fig. 1 illustrates the model of SeVEP that contains the following basic entities:

1) **Voter**: The voter ($V_k$) is a user who obtains valid credentials from the credential issuer to cast a vote ($k = 1, \ldots, N$, where $N$ is equal to maximum voters allowed in polling). $V_k$ use these credentials in the first level of MFA scheme to get an access to SeVEP.
2) **Voting device**: The voting device ($\mathrm{VD}_S$) is a computing device that is selected by $V_k$ to cast a ballot. $V_k$ can use as many as $S$ computing devices to cast his/her vote. Besides $\mathrm{VD}_S$, $V_k$ uses another computing device as a validation device to receive return and confirmation codes.
3) **Polling organization**: The polling organization (PO) is a trusted entity that is responsible of setting up the poll (poll questions and their corresponding voting options, etc.), tallying the votes and publishing the results of the poll. *PO* consists of $t$ polling organizers: $\mathrm{PO}_1 \ldots \mathrm{PO}_t$. It is assumed that out of $t$ POs, there is one main PO who manages the remaining POs. Also, in a scenario where a voter complaints about receiving a damaged or manipulated PCS, the main PO performs zero-watermarking extraction algorithm.
4) **Credential issuer**: The credential issuer (CI) is a trusted party that is in-charge of registering and authenticating the voter via MFA scheme.
5) **Bulletin board**: The bulletin board (BB) is a publicly verifiable entity where the results of various steps of the polling process including the final polling result are published by the authorized entities. After the polling phase is completed, a voter obtains access to the BB to verify that his/her vote is included in the final tally, and view all the results of the polling and post-polling phases published on the BB by the involved entities. All the entities of SeVEP have read-only access to BB, whereas some parties have write-only and append-only access to BB. No party is allowed to delete the existing data.
6) **Polling server**: The polling server (PS) verifies the ballots cast by the authenticated voters, updates, records and stores these ballots into the ballot box.
7) **Code generator**: The code generator (CG) is an entity responsible of managing multiple polling code generators (PCG). SeVEP assumes six $PCG_{\mathbb{X}}$ ($\mathbb{X} = 1, \ldots, 6$) responsible for generating return codes, acknowledgment and confirmation codes to be used in the polling phase. Also, a mapping table is generated by each PCG to map long-length return codes to small-length return codes. By adding more PCGs, the computational

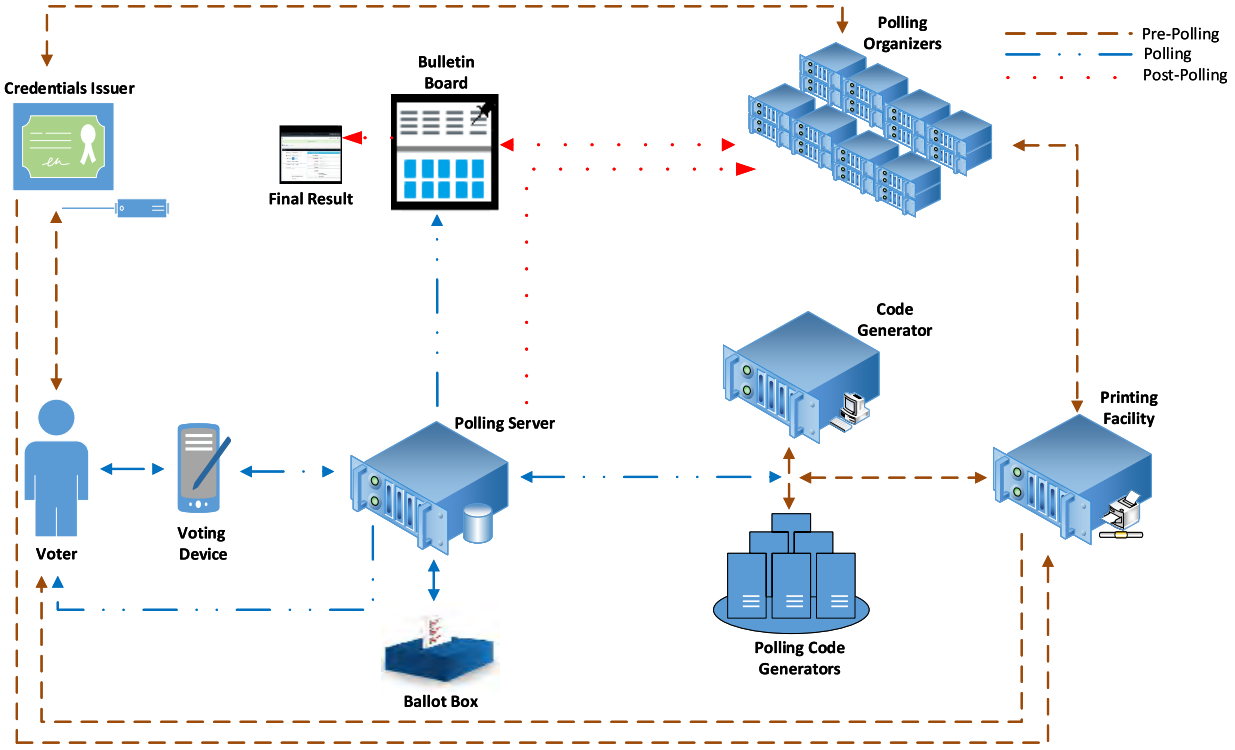


**FIGURE 1.** Overview of SeVEP.

costs associated with the generation of the codes can be reduced considerably.

8) **Printing facility**: The printing facility (PF) manages the printing of polling code sheet (PCS) that contains voting options along with their corresponding return codes, polling code sheet identity ($PCS_{ID}$), acknowledgment and confirmation codes, and a barcode. Additionally, PF is responsible for embedding a zero-watermark into each PCS. PF then encloses three watermarked PCSs into a sealed envelope, places a unique polling tag onto that envelope, and delivers it to the authenticated voter only in cooperation with CI.

### B. DESIGN REQUIREMENTS AND ASSUMPTIONS

In this section, the design requirements, and general and security assumptions of SeVEP are described.

- **Authenticity:** The voter must provide his/her authentication factors (knowledge, possession and inherence factors) required in a three-round multifactor authentication scheme to get authenticated. Only authorized voters are provided with an access to the polling page.
- **Flexibility:** The poll should contain a variety of question formats. Also, a voter may receive different number of voting options from other voters.
- **Multi-voting:** The authenticated voter can use any light-weight computing device as a voting device to cast his/her votes for a maximum of three times. Only the last cast vote (within an allowed polling period) is considered valid. A voter is prohibited from double voting.
- **Privacy:** Once the voter has cast his/her vote, it should not be linked with his/her identity.
- **Voter coercion:** During polling, a coercer should not be able to coerce a voter to cast a vote for a specific voting option.
- **Authenticity of polling code sheets:** A mechanism should be provided that guarantees integrity and authenticity of the polling code sheets sent to the voter during poll registration phase.
- **Fairness:** No entity can gain any knowledge about the partial ballot before the end of the polling.
- **Cast-as-intended verifiability:** After casting a vote, there should be mechanism that allows a voter to verify that the vote cast by his/her untrusted voting device indeed corresponds to what he/she cast in the polling phase, and it has not been modified.
- **Tallied-as-recorded verifiability:** After completion of the tallying phase, the poll results should be published on the BB by the POs such that the voter, auditors or any third party observers are able to verify that the votes tallied correspond to the cast votes. Also, a voter should be able to verify that his/her vote was correctly included in the tallying phase.
- **Cost effectiveness:** The polling system should be efficient (decreased computational overheads) and scalable (should be able to support 100 to 10 million voters).

#### 1) SYSTEM ASSUMPTIONS

The general design assumptions of SeVEP are as follows:

- Each poll question (in any available format) consists of multiple options, and the voter should select only one option per question. The selected options must be ordered sequentially.
- A voter is allowed to cast his/her vote three times within the allowed polling period ($t_p$) using his/her voting device.
- The polling phase of SeVEP assumes that the voter has two computing devices, i.e., one to cast his/her vote (a voting device), and another to open an email containing return and confirmation codes (a validation device).
- The return, acknowledgment and confirmation codes are composed of 6, 6 and 8 alphanumeric digits, respectively. Each digit is encoded with extended ASCII values, thus, allowing upper-case (A–Z) and lower-case letters (a–z), digits (0–9) and special symbols.
- Each PCS contains return codes corresponding to 5 fixed voting options for each poll question.
- PCSs are assumed to be cooperatively generated beforehand by CG, six $PCG_{\mathbb{X}}$, and PF, i.e., it is an offline process that is carried out before the start of the polling.
- For a proof-of-concept, it is assumed that each $PCG_{\mathbb{X}}$ has an access to a database of 50 pre-generated keys of length 1024-bits to assist ten million users.
- Each voter has access to the general parameters and the public keys, which are made available by PS, POs, PF, CG and six $PCG_{\mathbb{X}}$.
- During registration with SeVEP, it is assumed that each voter provides his/her phone number and email address to CI.
- Six $PCG_{\mathbb{X}}$ are assumed in generation of the polling code sheets to achieve efficiency. With an increase in the number of PCGs, the computational costs incurred in generation of temporary cryptographic keys for generating return codes is reduced.

#### 2) SECURITY ASSUMPTIONS

In this section, we define the security assumptions of SeVEP.

- CI is a trusted entity that is responsible of authenticating a voter via a 3-layer multifactor authentication scheme. It is assumed that CI does not form a coalition with any other party to break voter's privacy by revealing his/her personal details (email, phone number, etc.). Also, CI generates a random number and shares it with an authenticated voter for the generation of a pseudo-identity using an interactive protocol [36], [37].
- A polling tag is used during the polling phase that allows PS to identify different votes (maximum 3) sent by a voter within $t_p$.
- The same pseudo-identity (issued by CI at the time of registration) should be used by the voter in all three rounds of the poll.
- During polling, a TLS channel is assumed between a voting device and PS.
- Three polling code sheets, watermarked and sealed in an envelope, are provided to an authenticated voter through a postal mail by PF. Optionally, the voter could collect the envelope from a specific delivery point (a locker pickup) by inputing his/her username and password into the locker machine.
- A semi-passive device fingerprinting mechanism is carried out by CI to determine the type of voting device used by the voter. In semi-passive device fingerprinting, it is assumed that after the voter is authenticated in the first layer of MFA, a connection is established between him/her and CI, who estimates a device's clock skew to fingerprint it.
- After the delivery of PCSs, PF destroys all the information (digital or paper) related to these sheets such that only the voter knows the contents of PCSs.
- The existence of a PKI is assumed such that any entity who uses the public key of another participant knows that this key belongs to a legitimate party. The RSA and ElGamal key generation is performed offline to generate key pairs.
- Cryptographic primitives and constructions used in SeVEP are secure and verifiable.
- During a polling phase, the confirmed ballots published by PS on the BB remain secret. After the mixing and tallying phase is completed, a voter gets an access to the BB to verify that his/her vote is included in the final tally.

### C. THREAT MODEL

This section describes the main attacks that may be performed on three key phases of SeVEP. These attacks can be aimed to break either the security or the privacy properties of the system.

#### 1) VOTER COERCION

Under this type of attack, the coercer may persuade a voter into voting for his/her chosen voting options. This attack is possible if a voter provides any form of voting receipt, which is sent to him/her by the polling organizers to allow individual verifiability.

#### 2) DOUBLE VOTING

Remote polling is vulnerable to electoral fraud due to the possibility of double voting, i.e., an authenticated but a malicious voter may attempt to cast multiple ballots in the same poll in a way where all his/her votes are counted in the final tally.

#### 3) VOTE MODIFICATION BY A MALICIOUS VOTING DEVICE

Remote polling is an attractive option to the voters since it allows them to cast their votes through their computing devices in their homes without going to a polling booth. However, the voting devices that are used to cast ballots in such systems may affect the final outcomes of the polling due to two possibilities: either the device is affected with a malware, or is controlled by a malicious entity (hacker). In either case, the voting options selected by a voter can be covertly modified before these are submitted to the polling organizers, and

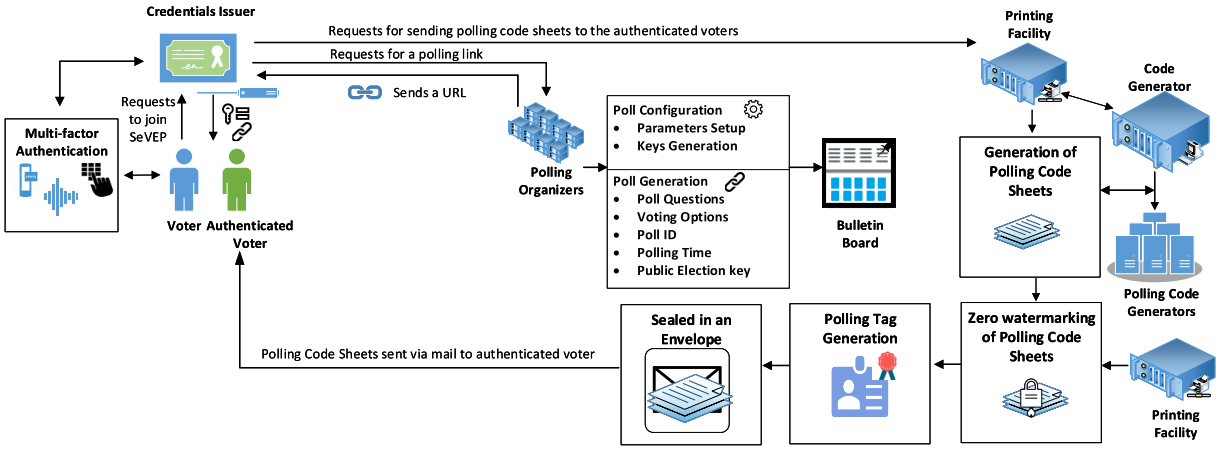


**FIGURE 2.** Pre-Polling Phase.

thus, falsely recorded and counted by the polling organizers undetectably (without the voter's knowledge).

#### 4) COALITION OF MALICIOUS ENTITIES

In the following, possible coalition attacks between malicious participants of SeVEP are presented:

- $VD_S$ and PS may collude to affect vote's privacy.
- PS and CG may form a coalition to infer the voting options selected by the voter.
- After the completion of the ballot processing phase, i.e., the voter has received a confirmation code, PS may collude with CG to replace the voter's confirmed ballot in the ballot box with the colluded vote.
- POs, PF and CI may collaborate to de-anonymize the voter.

#### 5) MODIFICATION OF A PCS

An authorized but a malicious voter may attempt to cast votes more than the allowed limit (3) within the polling period by modifying the content of PCS, and then, registering a complaint to the polling organizers of receiving an impaired PCS from PF.

#### 6) ATTACKS ON STORED BIOMETRIC SAMPLES

The following problems can undermine the security of biometric templates (voice and keystrokes pattern) stored at CI's end:

- If biometric templates are stored in a database in plain format and an attacker gains access to that database, those biometric templates can be assumed to be compromised forever.
- Database records containing biometric templates could be stolen or modified by an attacker to create spoofs, and thereby fraudulently enroll him/her into the system as an authorized user.

The security of the system against these attacks is discussed in Section IV-A.

### D. PRE-POLLING PHASE

Fig. 2 shows the pre-polling phase of SeVEP. In this initial phase, the following tasks are executed: (1) POs of SeVEP configure the poll by generating the poll site, cryptographic keys and system parameters; (2) A flexible poll setup is provided by POs; (3) A voter gets registered to SeVEP after getting authenticated via a 3-layer multifactor authentication scheme; (4) Only the authenticated voters receive PCSs from PF on CI's request; (5) PCSs sent to the voter are watermarked by PF using zero-watermarking; and (6) A polling tag is generated by PF to identify different votes of a same voter.

#### 1) POLL CONFIGURATION

In this phase, the polling cryptographic parameters $(p, q, g)$ to be used in ElGamal cryptosystem and homomorphic PRF are defined and published. A cyclic $\mathcal{G}_p \subseteq \mathbb{Z}_p^*$ of quadratic residues modulo a safe prime $p = 2q + 1$ is chosen as a common group for all the cryptographic operations used in SeVEP. The key pairs of PF $(K_{pPF}, K_{sPF})$, PS $(K_{pPS}, K_{sPS})$, CG $(K_{pCG}, K_{sCG})$, and PCGs are generated. Also, a joint public encryption key $(K_{pPCG})$ and a shared secret decryption key $(K_{sPCG})$ are created by six PCGs using distributed cryptosystem. Similarly, POs create a joint public encryption key and a shared secret decryption key for ElGamal cryptosystem. Each PO creates its share of the key and posts the public part along with the proofs at BB. BB checks the proofs and combines the shares to form a public election key $(K_{pPO})$ to be used by the voters to encrypt their votes before casting them. A vote encrypted under $K_{pPO}$ can only be decrypted by $K_{sPO}$ if all POs collaborate. POs and PS are provided with "write" and "append" access to the BB. CG is provided with "write-only" access to the BB. The voters are provided with "read-only" access to the BB.

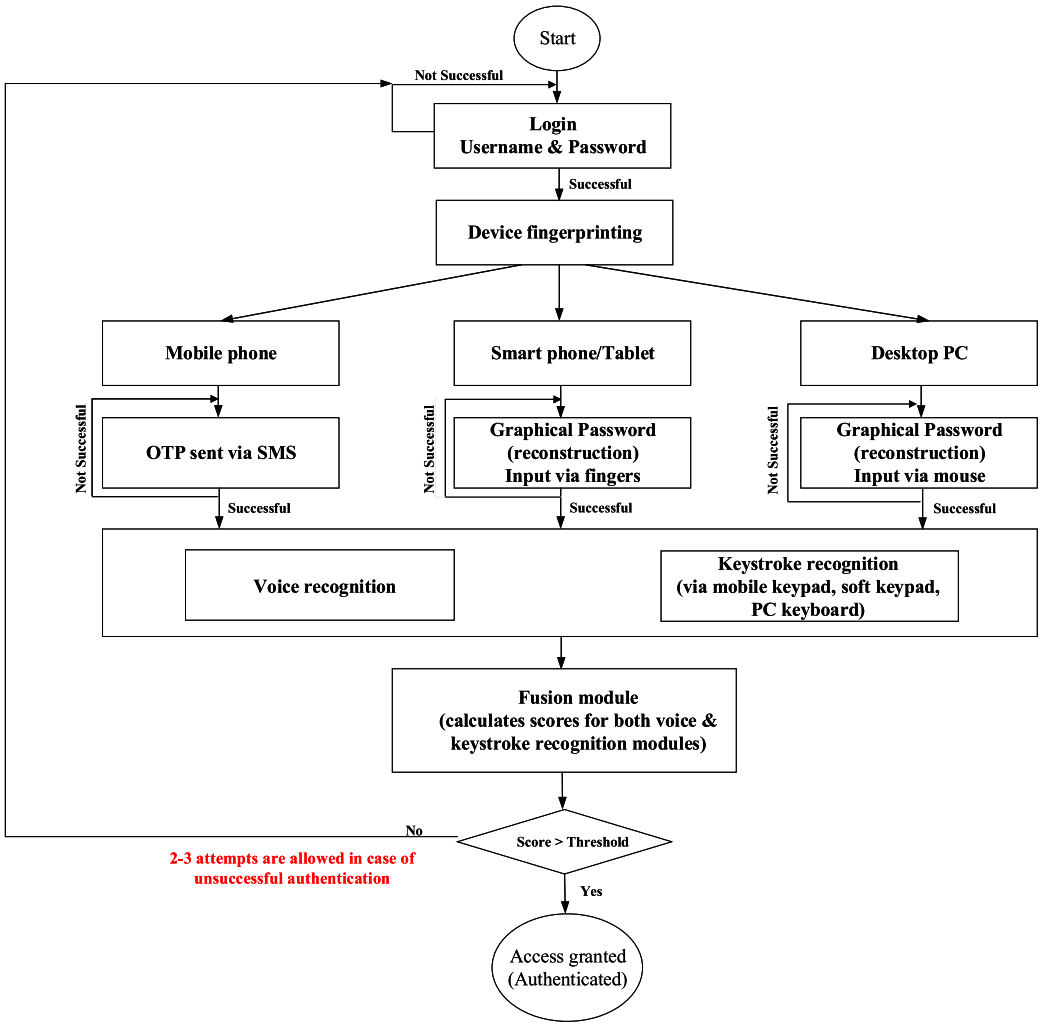


**FIGURE 3.** Multifactor authentication scheme.

### 2) FLEXIBLE POLL SETUP

A poll site is generated by POs that contains a unique poll identifier, poll questions (in various formats), voting options (represented by small bit-length prime numbers $\in \mathcal{G}_p$) for each poll question, polling time period ($t_p$), and public election key ($K_{pPO}$). CI sends the link of this poll site to the voters after a successful authentication.

In SeVEP, a flexible poll setup is provided in the sense that POs may choose different formats for the poll questions, e.g. rate the best option (1–5), choose the best option from the given range (A–E), and choose the preference (Yes/No/No Comment/Partial Yes/Partial No), etc. Additionally, the voters may receive poll questions with varying voting options, e.g., 3 instead of fixed 5 options. Also, there may be a scenario, where the voters receive shuffled polling options for each poll question.

### 3) MULTIFACTOR AUTHENTICATION

SeVEP employs MFA scheme that requires three authentication factors from the voter in order to establish his/her identity and provide access control. Fig. 3 illustrates the MFA scheme to identify and verify remote voters.

When a voter sends a registration request to SeVEP, CI sends him/her a url of web-page containing instructions to join SeVEP. During the first step of registration, the voter must choose his/her credentials (username and password), inputs a graphical password (either through a touch-screen interaction or with a mouse), and records a voice sample (a passphrase chosen by the voter) and a typing pattern (a word typed either through a mobile keypad or a desktop keyboard). All these factors along with the mobile number of the voter are registered with CI, who then sends him/her a link of a login page, where he/she can input these credentials (in three levels) to gain access to the polling site of SeVEP. In the first step, the voter inputs his/her credentials (username and password) as a first authentication factor to login into SeVEP. If the given credentials are verified, semi-passive device fingerprinting [38] is performed to recognize the computing device that the voter is utilizing to log in, e.g., it can be a mobile phone, or a smartphone/tablet, or a desktop PC, etc.

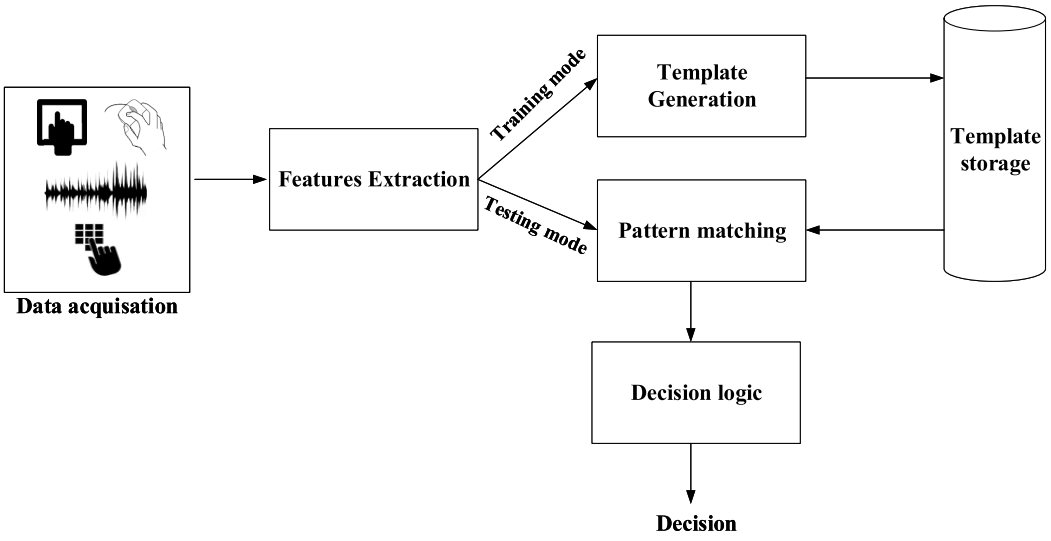


**FIGURE 4.** Generic authentication scheme.

In case of a mismatch, the voter is asked to re-enter the correct username and password. In SeVEP, we have considered three possible scenarios of voters based on the results of device fingerprinting: (1) a voter uses his/her mobile phone to cast votes; (2) a voter has a smartphone or a tablet to cast his/her votes; and (3) a desktop PC is used as a voting device by the voter.

In first scenario, the voter is required to enter an additional second-factor password, i.e., a one-time password (OTP). The OTP is a security token that consists of a string of numbers (numeric) or a combination of alphabets and numbers (alphanumeric) characters. This OTP is delivered by the CI as a text message (SMS) on the voter's registered mobile number. For security purposes, the OTP received on the mobile phone will be valid for 1 hour only after which the voter would have to obtain a new OTP. The codes are uniquely generated for each voter and can be used only once. After the voter inputs the OTP into the desired password field, it is forwarded to CI, who verifies the received OTP. If verified, the voter will be required to enter multimodal biometric features via a fusion module, where the voter will be authenticated based on voice and keystrokes recognition, else the voter is required to repeat this step.

The second scenario requires a voter to enter a graphical password for second-factor authentication. The voter gets authenticated from his/her multitouch interactions, i.e., drawing behavior through extraction of touch characteristics. A graphical password uses images, graphics or colors instead of letters, numbers or special characters. In SeVEP, a pure recall-based graphical password technique is considered in which the voter has to reproduce his/her password without being given any hint related to his/her registered password. Fig. 4 shows the working flow of a graphical password technique.

The following steps are followed in the graphical password technique:

- **Data acquisition:** During registration phase, a voter is prompted to input his/her gesture, i.e., draw a random password (a combination of three items: text (A–Z), numbers (0–9), and ASCII characters) 3–5 times on a screen.
- **Features extraction:** Touch events such as $x$ and $y$ coordinates of the touch point, size of the touch area, time-stamp of the touch event, and pressure (how hard the finger was pressed on the screen) are extracted from the input. Pressure and size are in the range of $[0, 1]$, whereas, $x$ and $y$ coordinates are in the range of screen resolution.
- **Template Generation & Storage:** The extracted features are then processed (positioning, sampling, and min-max normalization) and stored as authentication template for the voter. To model the user's touch characteristic, Dynamic Time Wrapping (DTW) [39] or Guassian Mixture Module (GMM) [40] or Hidden Markov Model (HMM) [41] can be used.
- **Pattern matching:** In the testing mode, first the system recognizes the individual items and verifies the combinational password. If it is correct, the system then verifies the drawing behavior by calculating the distance between the input sample and the stored authentication template using DTW or GMM or HMM.
- **Decision logic:** Depending on the results obtained in pattern matching, a decision is made that either the identity of the voter is verified successfully or not.

After the voter's reconstructed password matches with the stored graphical password, he/she is required to enter his/her biometric features through a fusion module, in which he/she will be authenticated based on his/her voice sample and keystroke dynamics. In case there is not a successful match

between the input and stored password sample, the voter is required to repeat this step.

Finally, in the last scenario, the voter is authenticated from his/her interactions with the graphical user interface on his/her PC through extraction of characteristics of the mouse input device. SeVEP uses a pure recall-based graphical password technique, likewise in the second scenario. The graphical password technique [33] employed by SeVEP for the authentication of a desktop voter consists of the following steps (illustrated in Fig. 4):

- **Data acquisition:** In the registration phase, a voter is requested to input his/her gestures in a uni-stroke manner, i.e., draw a random password (a combination of three items: text (A–Z), numbers (0–9), and ASCII characters) 3–5 times using mouse clicks.
- **Features extraction:** Mouse events such as horizontal and vertical coordinates, elapsed time starting from the origin of the gesture till the voter enters his/her password, horizontal velocity, vertical velocity, tangential velocity, tangential acceleration, tangential jerk, slope angle of the tangent, curvature and curvature rate of change are extracted from the sample. These features exhibit strong reproducibility and discriminative capabilities.
- **Template Generation and Storage:** The extracted features are then processed (positioning and min-max normalization) and stored as authentication template for this voter. Edit distance (DTW) or nearest-neighbor distance technique can be used to model user's mouse input characteristics.
- **Pattern matching:** During the first stage of the testing mode, the system recognizes the individual items and verifies the password. If it is correct, the system then verifies the drawing behavior by calculating the distance between the input sample and the stored authentication template using DTW or nearest-neighbor distance technique.
- **Decision logic:** The output of the pattern matching module is a similarity score (indicating the proximity of train sample with the test sample), which is used by the decision logic module to make the final decision of the voter's identity.

Upon successful verification by CI, the last step of MFA is performed in which the voter gets authenticated based on his/her voice sample and keystroke dynamics. In case of an unsuccessful match between registered (trained) and current input samples (tested), the voter is required to repeat this step.

In our proposed MFA scheme, fusion is used to combine information from multimodal modules (such as voice (Fig. 4) and keystroke recognition (Fig. 4)) to improve accuracy, robustness, precision, and efficiency of SeVEP. All the phases of voice and keystroke recognition modules are similar to the ones used in graphical password techniques except for the pattern matching stage, where the resulting similarity scores of each module (voice and keystroke) are combined to obtain a single fused score, which is known as score-level fusion, that asserts the level of the claimed identity. Since the scores obtained from two different modules may not belong to the same distribution or range, therefore, the scores are transformed to a common domain before they can be combined. This process is called score-normalization. Here, min-max normalization is used to normalize the score within the range [0, 1]. The scores are then fused using either sum fusion, product fusion or weighted average fusion [34]. If the calculated score exceeds the set threshold value, the voter is authenticated and is given an access to SeVEP polling site, else the voter is directed to the login page, i.e., the first level of MFA scheme. In case the voter exceeds the criteria set for possible MFA attempts (3), he/she gets blocked from SeVEP by CI.

A security analysis of MFA scheme is provided in Section IV-A.

#### 4) GENERATION OF POLLING CODE SHEETS

The code generator (CG), six polling code generators ($PCG_{\mathbb{X}}$), and the printing facility (PF) cooperatively generate polling code sheets (PCS) by employing numerous cryptographic operations (e.g., encryption, hmac, etc.) based on their respective key pairs and unique session keys. For generation of PCS, as a proof-of-concept, we have assumed that the main PO generates 4 polling questions ($Q_i$ with $i = 1, \ldots, 4$) with each $Q_i$ having fixed (5) voting options ($v_j$ with $j = 1, 2, 3, 4, 5$) that are represented by small bit-length prime numbers. Before initiating the process of generating PCS, the main PO sends these voting options to each polling code generator ($PCG_{\mathbb{X}}$) of SeVEP. As an example, the following voting options (represented by prime numbers) for 4 poll questions are assumed: $v_{1j} = \{11, 13, 37, 19, 2\}$, $v_{2j} = \{7, 29, 31, 41, 47\}$, $v_{3j} = \{43, 53, 5, 59, 71\}$, and $v_{4j} = \{23, 61, 67, 17, 83\}$.

A key is randomly selected by $PCG_{\mathbb{X}}$ from a database of 50 pre-generated cryptographic keys ($l = 1, \ldots, 50$). Also, a unique symmetric key ($K_{sess_{\mathbb{X}}}$) of length 256-bits is generated by each $PCG_{\mathbb{X}}$. Fig. 5 illustrates the generation of PCSs.

Following steps are performed between CG, $PCG_{\mathbb{X}}$ and PF to generate a PCS for the authenticated voter of SeVEP.

- Using PRF based on DDH assumption and a selected secret key, each $PCG_{\mathbb{X}}$ calculates partial return codes ($RC_{ij}(PCG_{\mathbb{X}})$) for each voting option $v_j$ of $Q_i$:

$$
\begin{aligned}
RC_{ij}(PCG_1) &= F_{\mathrm{DDH}}(K_{sa_l}, v_{ij}),\\
RC_{ij}(PCG_2) &= F_{\mathrm{DDH}}(K_{sb_l}, v_{ij}),\\
RC_{ij}(PCG_3) &= F_{\mathrm{DDH}}(K_{sc_l}, v_{ij}),\\
RC_{ij}(PCG_4) &= F_{\mathrm{DDH}}(K_{sd_l}, v_{ij}),\\
RC_{ij}(PCG_5) &= F_{\mathrm{DDH}}(K_{se_l}, v_{ij}),\\
RC_{ij}(PCG_6) &= F_{\mathrm{DDH}}(K_{sf_l}, v_{ij}),
\end{aligned}
$$

  where $F_{\mathrm{DDH}}(K, v_{ij}) = \mathcal{H}_1(v_{ij})^K$. Each $PCG_{\mathbb{X}}$ sends $RC_{ij}(PCG_{\mathbb{X}})$ to CG.
- CG computes a product ($RRC_{ij}$) of the received partial return codes, e.g., a product of the voting option

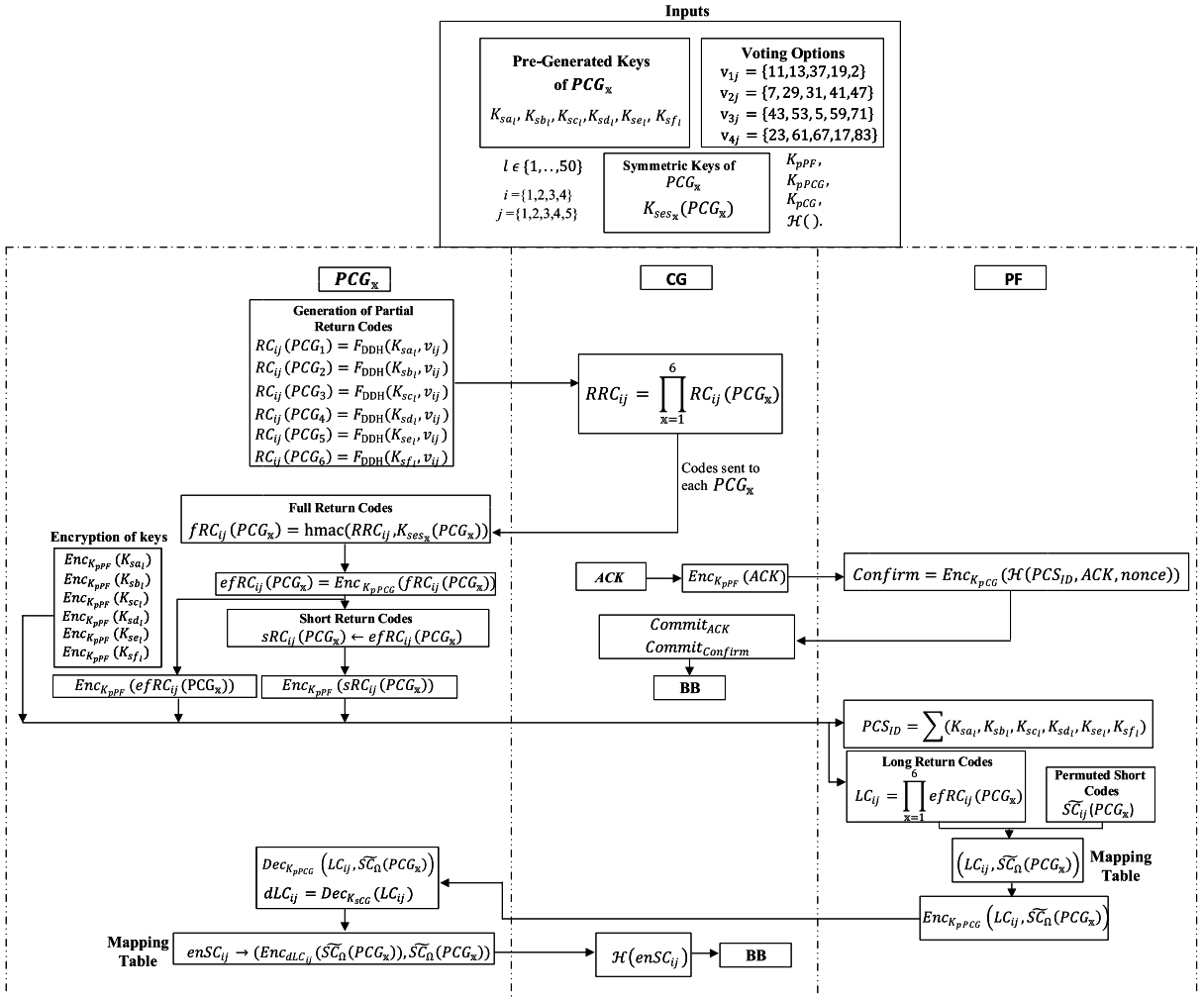


**FIGURE 5.** Generation of Polling Code Sheets.

"1" of $Q_1$ ($RRC_{11}$) is computed as: $RRC_{11} = \prod_{\mathbb{X}=1}^{6} RC_{11}(PCG_{\mathbb{X}})$. CG sends $RRC_{ij}$ to each $PCG_{\mathbb{X}}$.

- Using the received codes $RRC_{ij}$ and the unique symmetric key, each $PCG_{\mathbb{X}}$ computes a full return code in the following way:

$$fRC_{ij}(PCG_{\mathbb{X}}) = hmac(RRC_{ij}, K_{ses_{\mathbb{X}}}).$$

- Each $PCG_{\mathbb{X}}$ encrypts $fRC_{ij}(PCG_{\mathbb{X}})$ with the joint public key ($K_{pPCG}$) to obtain encrypted return codes ($efRC_{ij}(PCG_{\mathbb{X}})$) that are sent to PF.
- Random codes $sRC_{ij}(PCG_{\mathbb{X}})$ of small length (64-bits) are generated by each $PCG_{\mathbb{X}}$. Each $sRC_{ij}(PCG_{\mathbb{X}})$ corresponds to the long return codes $efRC_{ij}$ of length 1024-bits:

$$sRC_{ij}(PCG_{\mathbb{X}}) \leftarrow efRC_{ij}(PCG_{\mathbb{X}}).$$

- Both these codes ($sRC_{ij}(PCG_{\mathbb{X}})$ and $efRC_{ij}(PCG_{\mathbb{X}})$) are encrypted by each $PCG_{\mathbb{X}}$ using the public key ($K_{pPF}$) of PF. The encrypted codes are then sent to PF.
- Each $PCG_{\mathbb{X}}$ encrypts its secret key (that was used in the computation of partial return codes) with $K_{pPF}$, and sends the encrypted keys to PF.
- In the meanwhile, CG generates a random 6-digits Acknowledgment ($ACK$) code of length 64-bits (each digit is encoded with Extended ASCII). $ACK$ is used by a voter in the polling phase to provide confirmation of the received return codes. CG encrypts $ACK$ with $K_{pPF}$, and sends $Enc_{K_{pPF}}(ACK)$ to PF.
- Upon receiving the sets of the return codes (both long and short), the encrypted keys, and the encrypted $ACK$ code, PF performs decryption on these items with $K_{sPF}$ to compute the following:
  - $Dec_{K_{pPF}}(Enc(K_{sa_l}, K_{sb_l}, K_{sc_l}, K_{sd_l}, K_{se_l}, K_{sf_l}))$ is performed to obtain cryptographic keys, which are used to generate a PCS ID ($PCS_{ID}$):

$$PCS_{ID} = \sum(K_{sa_l}, K_{sb_l}, K_{sc_l}, K_{sd_l}, K_{se_l}, K_{sf_l}).$$

  - $Dec_{K_{pPF}}(Enc(ACK))$ to obtain plaintext $ACK$.
  - $Dec_{K_{pPF}}(Enc(efRC_{ij}(PCG_{\mathbb{X}})))$ is performed to compute a long code ($LC_{ij}$) for each voting option:

$$LC_{ij} = \prod_{\mathbb{X}=1}^{6} efRC_{ij}(PCG_{\mathbb{X}}).$$

  - $Dec_{K_{pPF}}(Enc(sRC_{ij}(PCG_{\mathbb{X}})))$ is performed to obtain plaintext short return codes $SC_{ij}(PCG_{\mathbb{X}})$, which are permuted with a random permutation key $\rho$ to generate permuted codes $\widetilde{SC_{ij}}(PCG_{\mathbb{X}})$.
  - PF randomly selects $\widetilde{SC_{ij}}(PCG_{\mathbb{X}})$, and pairs each permuted code with the encrypted long return code such that it obtains $i \times j$ pairs of return codes to create a mapping table, e.g., in our proof-of-concept, $4 \times 5 = 20$ pairs of codes are generated: $(LC_{ij}, \widetilde{SC_{ij}}(PCG_{\mathbb{X}}))$.
  - PF encrypts each entry of the mapping table with $K_{pPCG}$, and sends it to each $PCG_{\mathbb{X}}$.

- PF is also responsible for generating a confirmation number (*Confirm*) that is used as a proof of a vote been confirmed by the voter:

$$Confirm = H(PCS_{ID}, ACK, nonce).$$

  *Confirm* is encrypted with $K_{pCG}$ and is sent to CG.
- The short return codes $(\widetilde{SC_{ij}}(PCG_{\mathbb{X}}))$ along with the corresponding voting options (depending on the poll format, following options are possible: $\{Yes, No, NoComment, PartialYes, PartialNo\}$ or $\{A, B, C, D, E\}$ or $\{1, 2, 3, 4, 5\}$), $PCS_{ID}$, $ACK$, and *Confirm* are printed by PF as a PCS.
- Upon receiving the mapping table containing encrypted long codes and short return codes from PF, each $PCG_{\mathbb{X}}$ performs distributed decryption on the encrypted table (one time to decrypt the pair, and a second time to decrypt $LC_{ij}$ to obtain long return codes $(dLC_{ij})$).
- CG encrypts $\widetilde{SC_{ij}}(PCG_{\mathbb{X}})$ with the corresponding long return codes $dLC_{ij}$ to obtain $Enc_{dLC_{ij}}(\widetilde{SC_{ij}}(PCG_{\mathbb{X}}))$, which is then paired with the plaintext $\widetilde{SC_{ij}}(PCG_{\mathbb{X}})$ to create a mapping table containing $i \times j$ pairs of return codes:

$$enSC_{ij} = (Enc_{dLC_{ij}}(\widetilde{SC_{ij}}(PCG_{\mathbb{X}})), \widetilde{SC_{ij}}(PCG_{\mathbb{X}})).$$

  This mapping table is shared between CG and $PCG_{\mathbb{X}}$.
- CG computes the hash of each pair in the mapping table, and publishes it on BB as commitments to the return codes.
- Also, CG computes commitments to *ACK* and *Confirm*, and publishes these values on the BB. Since SeVEP allows a voter to cast his/her votes at most three times $(1 \leq z \leq 3)$, he/she is provided with three PCSs. This implies that there would be 3 tables of commitments to the return codes, and 3 values of $Commit_{ACK_z}$ and $Commit_{Confirm_z}$ for each voter on the BB.

#### 5) ZERO-WATERMARKING OF A PCS

PCS is considered a text file that is printed by the PF. CI requests PF to send three PCSs to the authenticated user only. In SeVEP, we have proposed a novel zero-watermarking algorithm for authentication of PCS, which is performed by PF before sending PCSs to the eligible voters. The proposed algorithm utilizes structural components in PCS to watermark it. PF logically embeds the watermark in PCS and generates a watermark key, which is registered with the polling organizer. Here PCS is not modified to embed the watermark, rather the characteristics of PCS are used to generate a watermark key. The algorithm provides security against content-preserving modifications, and simultaneously detects malicious tampering. PF embeds a watermark (a combination of polling authority's logo, a pseudo-identity of the user (issued by CI), and a numeric barcode printed on the PCS) into each PCS using zero-watermarking. Once the PF performs zero-watermarking of each PCS, it seals these watermarked PCSs into an envelope and sends it to the authenticated user only.

##### *a: EMBEDDING ALGORITHM*

The embedding algorithm is performed by PF in which PCS, a logo of the polling authority, a PI of the user, and a parameter *GS* (group size) are considered as inputs, and a secret key (*SK*) is obtained as an output, as shown in Algorithm 1.

**Algorithm 1** ***(Zero-Watermark Embedding Algorithm)***

**for** each PCS **do**
  Convert the logo: $\text{logo}_{PO} \leftarrow \text{Norm}(\text{logo})$.
  Generate a zero-watermark (*WM*): $WM = \text{XOR}(\text{logo}_{PO}, BC, PI)$ {*BC* is the barcode of the PCS.}
  Identify colons (:) in each row (from the start of PCS till the end).
  Generate text partitions (*TP*) using the colons obtained in Step 1.
  Generate groups using *GS*: $Group_{NO} = \frac{TP}{GS}$, where $Group_{NO}$ is a row or column vector containing $\frac{TP}{GS}$ strings.
  **for** each $Group_{NO}$ **do**
    Count occurrence of ASCII values in the group.
    $\text{LIST}_{\text{GN}} \leftarrow$ [ASCII value, No. of occurrences, Position in $Group_{NO}$]
  **end for**
  LIST $\leftarrow$ Combine $\text{LIST}_{\text{GN}}$ of all groups.
  **for** each character *c* of *WM* **do**
    **if** $c \in$ LIST **then**
      PF looks for $\text{LIST}_{\text{GN}}$ that contains *c*.
      PF extracts *GN* and Position from $\text{LIST}_{\text{GN}}$.
      $SK[i] \leftarrow 0$
      $SK[i+1] \leftarrow$ (GN, Position)
    **else**
      $SK[i] \leftarrow 1$
      $SK[i+1] \leftarrow Enc_{\text{ASCII95}}(c)$
    **end if**
    Set: $i = i + 2$
  **end for**
**end for**
RETURN *SK*

In Algorithm 1, the logo of the polling authority is converted into ASCII values within the range 33–127 using the min-max normalization (Norm). This range contains 10 digits

(0–9), 33 special symbols, 26 capital alphabets (A–Z), and 26 small alphabets (a–z). Afterwards, colons are used as the separators to generate text partitions ($TP$). These partitions are then combined together to form groups based on group size($GS$), where $GS$ is selected by PF. A $Group_{NO}$ is generated in either a row or column vector form that contains $\frac{TP}{GS}$ strings containing ASCII values. Then, occurrence of each ASCII character is counted in string of $Group_{NO}$. A list $LIST_{GN}$ is created that is populated with frequently occurring ASCII character along with its position in a row or column vector, and number of times it has occurred. Along with the most frequent occurring character, this list also contains the characters that have occurred at least two times. The rest are not included in the list. Afterwards, a final list is created by combining $LIST_{GN}$ of all the groups. A secret key ($SK$) is generated by using characters of a watermark and $LIST$ as described in the embedding algorithm. $SK$ is registered at $PO$, along with $WM$, logo, current date and time, $GS$, poll ID, and encrypted barcode of the PCS.

In Algorithm 1, PF performs the following steps to perform ASCII95 encryption on the characters of $WM$:

- Using 8-bit ASCII encoding, the ASCII value of the character is coded to a binary string. This string is then filled with zeros to obtain a 32-bit binary string.
- The binary string is converted to Base 10, i.e., $(X)_{10}$.
- The number obtained is converted to Base 95, i.e., $(X)_{95}$.
- Each value in $(X)_{95}$ is replaced by an ASCII character of code (value+33) to obtain the encrypted value.

#### b: EXTRACTION ALGORITHM

The extraction of zero-watermark is performed by the PO, who uses this algorithm to resolve conflicts such as complaint received by the voter of receiving a tampered or damaged PCS. The algorithm takes the altered or damaged PCS, a secret key $SK$, length of $WM$ and a parameter $GS$ (group size) as inputs, and extracts the watermark ($WM'$), as described in Algorithm 2.

Likewise Algorithm 1, Algorithm 2 uses colons as the separators to generate text partitions ($TP'$), which are then combined together to form groups based on $GS$. Each generated $Group'_{NO}$ is either a row or column vector containing $\frac{TP}{GS}$ strings of ASCII values. After creating groups, for each group, PO counts the occurrence of each ASCII character in all the strings. Then, PO populates a list ($LIST'_{GN}$) with frequently occurring ASCII character, its position in a row or column vector, and times it has occurred. $LIST'_{GN}$ contains the most frequent occurring character as well as the characters that have occurred at least two times. Then, $LIST'_{GN}$ of all the groups are combined to form a $LIST'$ as done previously in Algorithm 1. A watermark is extracted by using the contents of $SK$ and $LIST'$ as described in the extraction algorithm.

After performing Algorithm 2, the PO sends $WM'$, an encrypted barcode, and a poll ID to the PF, who performs decryption on the received encrypted barcode ($D_{BC}$), and then calculates $XOR$ ($WM'$, $\text{logo}_{PO}$, $D_{BC}$) to obtain credentials of the user ($ID_{V_k}$). The credentials are then sent to the PO, who can then take an appropriate action against the user in cooperation with CI.

**Algorithm 2** ***(Zero-Watermark Extraction Algorithm)***

**for** each PCS **do**
  Identify colons (:) in each row (from the start of PCS till the end).
  Generate text partitions ($TP'$) using the colons obtained in Step 1.
  Generate groups using $GS$: $Group'_{NO} = \frac{TP}{GS}$, where $Group'_{NO}$ is a row or column vector containing $\frac{TP}{GS}$ strings.
  **for** each $Group'_{NO}$ **do**
    Count occurrence of ASCII values in the group.
    $\text{LIST}'_{\text{GN}} \leftarrow$ [ASCII value, No. of occurrences, Position in $Group'_{NO}$]
  **end for**
  $\text{LIST}' \leftarrow$ Combine $\text{LIST}'_{\text{GN}}$ of all groups.
  Set: $L = Length(WM)$, $index = 1$, $i = 1$, and $k = 1$.
  **while** $index < L$ **do**
    **if** $SK[i] == 0$ **then**
      {where, $i = 1, 3, 5...$}
      Extract the value stored in $SK[i + 1]$, i.e., (GN, Position).
      Against $GN$, PO looks into $LIST'_{GN}$, and extracts the character placed at Position from $LIST'_{GN}$, i.e.,
      $WM'[i + 1] \leftarrow c'$ placed at (GN, Position) of $LIST'_{GN}$.
    **else**
      $WM'[i+1] \leftarrow Dec_{\text{ASCII95}}(c)$ stored in $SK[i+1]$
    **end if**
    Set: $i = i + 2$, $k = k + 1$, and $index = index + 1$
  **end while**
**end for**
RETURN $WM'$

In Algorithm 2, PO performs the following steps to perform ASCII95 decryption on the characters stored in $SK[i + 1]$:

- For each character, obtain an ASCII code and subtract 33 from it.
- Convert the number to Base 10, i.e., $(X)_{10}$.
- The decimal number is converted to Base 2, i.e., a binary string.
- A decrypted character is obtained by encoding the binary code with the 8-bit ASCII encoding.

### 6) GENERATION OF A POLLING TAG

SeVEP allows a voter to cast three votes within a permitted polling time while preventing double voting by him/her. To do so, PF generates a polling tag ($tag_{Poll}$) to identify different votes ($1 \leq z \leq 3$) sent by a single voter.

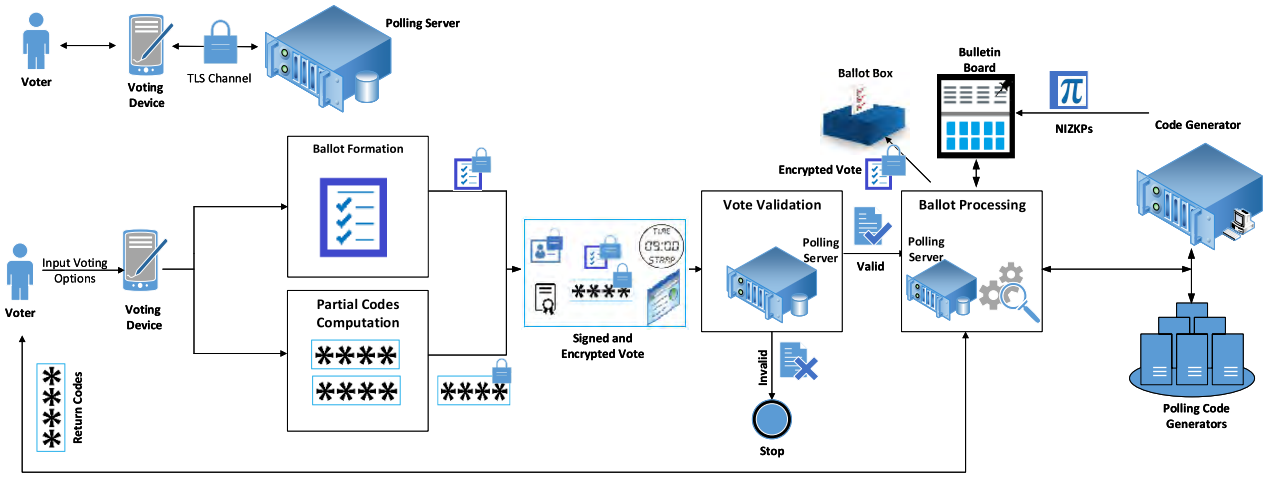


**FIGURE 6.** Overview of Polling Phase.

After the voter is authenticated by CI, it sends a request to PF to deliver three PCSs to the authenticated voters only. During a PCS generation process, PF encodes the $PCS_{ID}$ of each PCS, time-stamp, and a voter's pseudo-ID ($ID_{V_k}$) into a numeric barcode, which is then printed on the PCS as an identifier. After receiving the PCS delivery request from CI, PF performs zero-watermarking for each PCS, computes $tag_{Poll}$ by generating a hash of three barcodes, and seals these watermarked PCSs into a single envelope. Then, PF prints $tag_{Poll}$ on the envelope and sends it to the voter via a postal mail. Also, PF sends $tag_{Poll}$ to the PS, who stores it against the user's $ID_{V_k}$. In three (allowed) voting attempts, the voter must attach $tag_{Poll}$ to his/her vote, which is then verified by the PS.

### E. POLLING PHASE

After the completion of the pre-polling phase, each $V_k$ may cast his/her ballot using his/her $VD_S$. $VD_S$ generates a ballot with the selected voting options of each polling question ($Q_i$), and submits it to PS. $V_k$ can cast his/her ballot at most three times ($1 \leq z \leq 3$). Following are the steps followed by each $V_k$ to casts his/her ballot (as illustrated in Fig. 6):

- $VD_S$ sets up a TLS connection with PS, who authenticates $VD_S$ upon successful verification of his/her credentials from CI. Once authenticated, PS sends a time-stamp ($t_s$) with the current time to $VD_S$.
- $V_k$ selects one option for each poll question, i.e., he/she inputs $v_j$ of each $Q_i$ into his/her $VD_S$.
- $VD_S$ computes a partial ballot as a product of options ($v_{ij}$) selected by $V_k$:

$$B_{V_k} = \prod_{i=1}^{4} v_{ij}.$$

  $VD_S$ encrypts $B_{V_k}$ with the joint public key of POs ($K_{pPO}$) to obtain the ElGamal ciphertext:

$$(c_1, h_1) = Enc_{K_{pPO}}(B_{V_k}).$$

  Also, $VD_S$ generates NIZKP ($\pi_{enc}$) to prove knowledge of the randomness used for computing the encryption of $B_{V_k}$.
- Additionally, a 3-digit (alphanumeric) random number is entered by $V_k$ into his/her $VD_S$ for each voting option ($\gamma_1, \gamma_2, \gamma_3$).
- $VD_S$ concatenates three digits to create $\gamma_{V_k}$:

$$\gamma_{V_k} = \gamma_1 || \gamma_2 || \gamma_3.$$

  Then, $VD_S$ encrypts $\gamma_{V_k}$ with $K_{pPO}$ to generate the ElGamal ciphertext:

$$(d_1, e_1) = Enc_{K_{pPO}}(\gamma_{V_k}).$$

  Both the ciphertexts are then concatenated by $VD_S$: $(c_1, h_1)||(d_1, e_1)$. Also, a NIZKP ($\pi_{enc_{con}}$) is generated by $VD_S$ to prove that $(c_1, h_1)||(d_1, e_1)$ is equivalent to the concatenation of two ElGamal encrypted ciphertexts under $K_{pPO}$. $VD_S$ then signs $(c_1, h_1)||(d_1, e_1)$, and $t_s$:

$$\text{Sign}_{K_{sV_K}}((c_1, h_1)||(d_1, e_1), t_s, \pi_{enc}, \pi_{enc_{con}}).$$

- In addition to voting options and random alphanumeric codes, $V_k$ also inputs $tag_{Poll}$ and $PCS_{ID}$ into $VD_S$. Using $PCS_{ID}$, $VD_S$ computes partial codes corresponding to voter's voting options ($v_{ij}$) using a pseudorandom function (PRF) based on DDH assumption with homomorphic properties. Each computed partial return code is then encrypted with the public key of CG ($K_{pCG}$). Also, NIZKPs ($\pi_{PCS_i}$) are generated for each computed partial return code by $VD_S$.
- The final ballot submitted to PS by $VD_S$ consists of the following items:

$$\begin{aligned} &ballot_{V_k} \\ &= Enc_{K_{pPS}}(ID_{V_k}), (c_1, h_1)||(d_1, e_1), \pi_{enc}, \\ &\quad tag_{Poll}, t_s, \pi_{enc_{con}}, t_{V_k}, Enc_{K_{pCG}}(F_{\text{DDH}}(PCS_{ID}, v_{1j})), \\ &\quad \pi_{PCS_1}, Enc_{K_{pCG}}(F_{\text{DDH}}(PCS_{ID}, v_{2j})), \pi_{PCS_2}, \\ &\quad Enc_{K_{pCG}}(F_{\text{DDH}}(PCS_{ID}, v_{3j})), \pi_{PCS_3}, \\ &\quad Enc_{K_{pCG}}(F_{\text{DDH}}(PCS_{ID}, v_{4j})), \pi_{PCS_4}, \end{aligned}$$

$$\text{Sign}_{K_{sV_K}}((c_1, h_1)||(d_1, e_1), t_s, tag_{Poll}, \pi_{enc_{con}}),$$

where $t_{V_k}$ is the time of voting according to the system clock of $VD_S$.

#### 1) VOTE VALIDATION

After the ballot ($ballot_{V_k}$) is cast by $VD_S$, a vote verification process is initiated by PS to validate the received vote before processing it.

- PS performs decryption on $Enc_{K_{pPS}}(ID_{V_k})$ to obtain the plaintext identity of the voter, i.e., $ID_{V_k}$.
- PS looks into its database for a possible entry of ballot $ballot_{V_k}$ for $V_k$. If found, then PS checks the value of the flag ($FL$) that indicates the number of entries of $V_k$.
- If $FL = 0$, i.e., $ballot_{V_k}$ is not found against $V_k$'s record, PS continues the validation process by verifying the digital signature, proofs ($\pi_{enc}, \pi_{enc_{con}}$), and $tag_{Poll}$ contained in $ballot_{V_k}$.
- Also, PS verifies that $t_s$ used in $ballot_{V_k}$ is equal to the one sent to $V_k$. If verified, PS creates a new entry for $V_k$, stores his/her $ballot_{V_k}$, and sets $FL = 1$.
- A case $FL = 3$ implies that there are already three entries of $V_k$ (i.e., the voter has cast his/her vote three times). In this case, PS halts the polling process.
- If PS finds that there is already an entry of $V_k$ and $FL < 3$, it updates $t_s$, checks that the new time is more recent than that of an old entry, and verifies that same $tag_{Poll}$ is being used by $V_k$ to cast the vote.

#### 2) BALLOT PROCESSING

Once the votes are validated by PS, the following steps are performed to process the validated ballots:

- After creating or updating $V_k$'s voting record, PS encrypts pseudo-identity of the voter ($ID_{V_k}$) with $K_{pCG}$ ($Enc_{K_{pCG}}(ID_{V_k})$). PS sends this encrypted identity, encrypted partial return codes ($Enc_{K_{pCG}}(F_{\text{DDH}}(PCS_{ID}, v_{ij}))$, and NIZKPs ($\pi_{PCS_1}, \pi_{PCS_2}, \pi_{PCS_3}, \pi_{PCS_4}$) to CG.
- CG performs decryption on the received encrypted items with its secret key ($K_{sCG}$) to obtain the plaintext of voter's identity ($ID_{V_k}$) and the clear-text of partial return codes.
- CG sends partial return codes and NIZKPs to each $PCG_{\mathbb{X}}$.
- NIZKPs are verified by each $PCG_{\mathbb{X}}$, and upon successful verification, full return codes are computed using the keyed-PRF and a symmetric key (the same key used to compute the return codes during pre-polling generation of PCS):

  $$NRC_{ij}(PCG_{\mathbb{X}}) = hmac(F_{\text{DDH}}(PCS_{ID}, v_{ij}), K_{ses_{\mathbb{X}}}).$$

  Each $PCG_{\mathbb{X}}$ encrypts $NRC_{ij}(PCG_{\mathbb{X}})$ with $K_{pPCG}$ to obtain $eNRC_{ij}(PCG_{\mathbb{X}})$:

  $$eNRC_{ij}(PCG_{\mathbb{X}}) = Enc_{K_{pPCG}}(NRC_{ij}(PCG_{\mathbb{X}})).$$

  Each $eNRC_{ij}(PCG_{\mathbb{X}})$ is sent to CG.
- CG computes long return codes $NLC_{ij}$ using the received $eNRC_{ij}(PCG_{\mathbb{X}})$:

  $$NLC_{ij} = \prod_{\mathbb{X}=1}^{6} eNRC_{ij}(PCG_{\mathbb{X}}),$$

  and sends these long codes to each $PCG_{\mathbb{X}}$.
- Upon receiving the long codes, each $PCG_{\mathbb{X}}$ performs lookup on its stored mapping table (sent by PF during pre-polling PCS generation) to extract the corresponding short return codes. These short return codes are encrypted by each $PCG_{\mathbb{X}}$ with $NLC_{ij}$, and is then sent to CG.
- CG extracts the short return codes ($\widetilde{SC_{ij}}(PCG_{\mathbb{X}})$) corresponding to the received encrypted codes by looking into its stored mapping table (shared with $PCG_{\mathbb{X}}$).
- Upon finding the matching entries, CG encrypts the corresponding $\widetilde{SC_{ij}}(PCG_{\mathbb{X}})$ with $K_{pPS}$, and sends the encrypted short return codes to PS.
- PS performs decryption on $Enc_{K_{pPS}}(\widetilde{SC_{ij}}(PCG_{\mathbb{X}}))$ to obtain the plaintext short return codes, which are then sent to the voter's email address in the form of self-destructing email.
- Upon receiving an email from PS, $V_k$ opens it in his/her validation device, and verifies whether the received short codes correspond to the printed short return codes of the PCS (used by $V_k$ in the polling phase to cast the ballot).
- If all the received codes match with the printed ones, $V_k$ inputs $ACK$ into $VD_S$ to finalize the ballot processing phase.
- $VD_S$ encrypts $ACK$ with $K_{pCG}$ and sends encrypted code ($Enc_{K_{pCG}}(ACK)$) to PS.
- PS sends the encrypted $ACK$ to CG, who decrypts it with $K_{sPS}$ to obtain a clear-text $ACK$. CG checks $ACK$ to confirm whether it is a valid opening for the $Commit_{ACK_z}$. If valid, CG checks the index of $Commit_{ACK_z}$ since there are three published commitments for each voter. CG checks the number corresponding to index "z" of $Commit_{Confirm_z}$ and extracts the corresponding $Confirm$ code. CG encrypts $Confirm$ with $K_{pPS}$, and sends it to PS.
- PS decrypts $Enc_{K_{pPS}}(Confirm)$ and sends the plaintext $Confirm$ code to $V_k$ via an email. $V_k$ checks on his/her validation device whether the received $Confirm$ code matches with the printed one on his/her PCS. If matched, the vote confirmation process is deemed successful.
- PS adds $ballot_{V_k}$ as a confirmed ballot into its ballot box. Only the validated votes with "confirmed" codes are considered in the tallying phase.

After the completion of the ballot processing phase, PS generates a hash of a ballot and publishes it on the BB. Till the publishing of the final results on the BB, the hashes of the ballots remain hidden from the voters.

### F. POST-POLLING PHASE

After the polling phase is completed, i.e., the polling period ($t_p$) is expired, PS no longer accepts the votes from the voters.

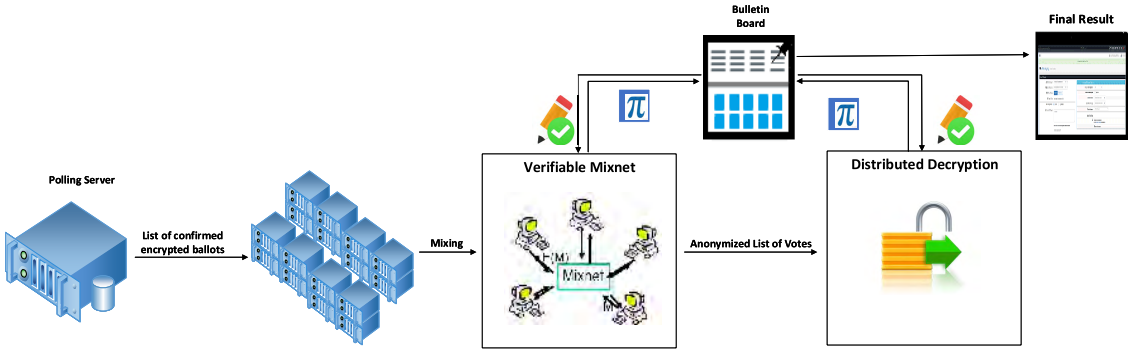


**FIGURE 7.** Overview of Post-Polling Phase.

Fig. 7 illustrates the post-polling phase of SeVEP. The list of ciphertexts $mathcalC_k = (c_k, h_k)||(d_k, e_k)$, which are stored in the ballot box with "Confirmed" status, is sent by PS to the main PO, who is responsible for mixing and tallying the received votes. The main PO initiates the mixing process to anonymize the votes such that it is impossible to link ciphertexts with the voters.

The mixing phase is carried out before the tallying process because the former provides vote's privacy whereas the later posts the polling results on the BB. A verifiable mixnet based on ElGamal encryption with the shuffle size equal to the number of POs, i.e., $t$, is instantiated to mix $\mathcal{C}_k$. Permutations ($\rho'$) and re-encryption randomizations ($\varphi$) are selected at random by each PO in the mixing phase. Here, $\rho'$ is used to only change the order of the ciphertexts contained in $\mathcal{C}_k$ without changing its content, whereas $\varphi$ is used in re-encryption of the ciphertexts to make them unlinkable. In order to provide the proof of correctness of the mixing, each of these values ($\rho'$ and $\varphi$) must be generated explicitly. Each PO performs mixing by permuting and re-encrypting the ballots ($\mathcal{C}_k^t = (c_k, h_k)||(d_k, e_k)$), and forwarding it to the next mixer (PO).

### MIXING AND TALLYING PHASE

The following steps are performed by POs in the mixing and tallying phase:

- $\mathcal{C}_k^1$ is input to the first $PO_1$, who selects a random secret permutation value ($\rho'^{(t_1)}$) to permute the input list to obtain a following new list:
$$\overline{\mathcal{C}_k^1} = \left\{(c^1_{\rho'^{(t_1)}k}, h^1_{\rho'^{(t_1)}k})||(d^1_{\rho'^{(t_1)}k}, e^1_{\rho'^{(t_1)}k})\right\}.$$
- $PO_1$ re-encrypts the new list using a randomization value ($\varphi^1$) to obtain $\mathcal{C}_k'^1$.
- $PO_1$ generates the NIZKP of correct mixing ($\pi_{mix_1}$) and submits it along with the mixing result to the BB.
- BB verifies the received proof ($\pi_{mix_1}$) and, upon successful verification, posts the mixed vote list for the next PO to mix.
- $\mathcal{C}_k'^1$ is input to the next $PO_2$, who performs permutation and re-encryption on $\mathcal{C}_k'^1$ to obtain a new list of ciphertexts. This process is continued by each PO involved in the mixing phase.
- The mixing phase is completed when the last PO, i.e., $PO_t$ outputs its list of ciphertexts and the corresponding proof to the BB.
- Upon the completion of the mixing phase, BB verifies all the proofs ($\pi_{mix_1}, \ldots, \pi_{mix_t}$) provided by the POs.
- After successfully verified by the BB, each PO can download an anonymized list of mixed ciphertexts to perform decryption using its share of a secret key.
- A list of plaintext ballots ($\mathcal{B}_{V_k}$) is obtained after the joint-decryption performed by $t$ $PO$s.
- In joint-decryption process, each PO generates a NIZKP of correct decryption ($\pi_{dec_t}$) and publishes it on the BB.
- The received NIZKPs of correct decryption shares ($\pi_{dec_t}$) are validated by the BB.
- Upon successful verification, the main PO performs prime factorization on the plaintext ballots ($\mathcal{B}_{V_k}$) to output the factors $v_{ij}$.
- Each factor is evaluated by the main PO to obtain the corresponding voting option.
- The voting options and the associated 3-digit number ($\gamma_{V_k}$) are published on the BB against each polling question.

## IV. RESULTS AND DISCUSSION

In this section, we provide an analysis of SeVEP in terms of security, privacy, and performance.

### A. SECURITY AND PRIVACY ANALYSIS

This section analyzes the security and privacy properties of SeVEP, and details how it addresses the design requirements and the threat model presented in Sections III-B and III-C, respectively.

#### 1) VOTER COERCION RESISTANCE

In SeVEP, coercers are unable to determine if the coerced voter has obeyed his/her instructions or not due to the fact that SeVEP allows multiple voting within $t_p$, i.e., only the last ballot cast by the voter is considered valid. Also, SeVEP provides multiple PCSs to the authenticated voters before the polling period. Thus, enabling the voters to always update their votes by using a different PCS, and

embedding a new time-stamp and a constant polling tag ($tag_{Poll}$) in the updated ballot before the poll is closed. Therefore, the coercer has no way of knowing if the vote cast in his/her presence and the return codes shown to him/her represents the ballot that was actually counted for that voter.

Alternatively, it can be assumed that the coercer is communicating with the voter during polling, and forces the coerced voter to hand over his/her PCSs to be ensured that the coerced voter voted for his/her voting choices. In this scenario, a coerced voter can generate a ballot that would contain a ciphertext of his/her choice of voting options and partial return codes corresponding to those options. Upon receiving the return codes from PS, the coerced voter can forge the received codes that would satisfy the coercer, e.g., by swapping the received return codes (corresponding to their intended choices) with the return codes of the options demanded by the coercer. Since the coerced voter receives these codes via self destructing email, he/she cannot be forced by a coercer to show him/her the received return codes. Upon successful matching between the codes (received codes and the printed codes on the PCS), the coercer would provide *ACK* to the coerced voter, who would send it to PS. Upon verification, PS sends *Confirm* code to the coerced voter, who sends the code to the coercer. Thus, coercer would be satisfied that his/her vote is recorded-as-intended in the ballot box at PS's end. After the completion of the tallying and mixing phase, if a coercer checks for his/her cast vote and does not find it, he/she may make a complaint to POs about missing vote. In such a case, POs would request him/her to give his/her polling credentials, e.g., polling tag, digital signature, pseudo-identity, time-stamp, NIZKPs, etc. Since the coercer only has PCSs of the coerced voter and no other information, he/she would be unable to submit the required polling credentials to prove that his/her vote is not included in the final tally.

Furthermore, if the three-attempt vote casting possibility is not considered enough to prevent vote coercion, then the polling system could be designed to allow the coerced voter to cast another vote after a temporary ban (e.g., one hour). In such a scenario, the polling system would enable the voter to obtain a new polling card sheet after passing through a multifactor authentication scheme and providing valid polling credentials (polling tag ($tag_{Poll}$), pseudo-ID ($ID_{V_k}$), and $PCS_{ID}$ of the three polling card sheets that were used in previous three attempts under a coercion attack). Upon successful multifactor authentication, the polling server will then check the voter's polling credentials in its database, and the status of the corresponding flag ($FL$), i.e., it verifies if $FL = 3$ or not.

#### 2) DOUBLE VOTE PREVENTION

In SeVEP, double voting by a single authenticated voter is prevented by a polling tag ($tag_{poll}$), which allows PS to identify different votes ($\leq 3$) of that voter. During vote validation phase, after the verification of voter's credentials such as his/her pseudo-identity ($ID_{V_k}$), digital signature ($Sign$) and NIZKPs, PS verifies that $tag_{poll}$ embedded in the ballot matches with the one sent to the voter in the poll registration. Since SeVEP allows the voter to cast his/her ballot three times within $t_p$, the valid ballot must always contain the same $tag_{poll}$ as described in assumptions (Section III-B2). Upon an unsuccessful match of $tag_{poll}$, PS halts the polling process even if all other credentials ($ID_{V_k}$, $Sign$, and NIZKPs) are verified.

Alternatively, it can be assumed that the authenticated voter is malicious, and he/she may attempt double voting by using different identity. SeVEP prevents this possible attack by restricting the use of a constant pseudo-identity during three rounds of polling, i.e., if the voter is using $ID_{V_k}$ to cast his/her vote, then $ID_{V_k}$ remains constant in all three rounds of polling. However, in case the voter requests for a new pseudo ID, all his/her previous votes ($\leq 3$) shall be revoked by PS on CI's request.

#### 3) CAST-AS-INTENDED VERIFIABILITY

SeVEP provides vote integrity through a verifiable mechanism that prevents vote's manipulation or modification by the malware-affected or controlled voting device. In the proposed cast-as-intended mechanism, return codes are used that enable a voter to detect whether his/her voting device is infected with malware or controlled by a hacker. For example, a possible attempt made by a malicious voting device to modify the ballot contents, and submit it on voter's behalf would be unsuccessful due to the fact that when the voter receives the return codes from PS, those codes would not match with the voter's intended voting options. Furthermore, the voter uses his/her validation device to open the email (containing the return codes corresponding to voter's intended voting options) sent by PS during ballot processing phase, thus, preventing the malicious voting device to obtain any information about these received return codes. Due to mismatch between the received and printed return codes (on a PCS), the voter will cast his/her vote using a different voting device.

#### 4) TALLIED-AS-RECORDED VERIFIABILITY

In SeVEP, POs publish the output of the mixing and tallying phase (voting options along with three-digit random codes, associated NIZKPs, and hashes of the confirmed ballots) on the BB so that a voter, any other participant, or an auditor can check whether the votes are counted correctly or not. The voters can verify the votes by generating hashes of their submitted ballots, and then compare them to the ones displaying on the BB. Moreover, the published three-digit random code (only known by the voter) on the BB confirms to the voter that his/her vote has been recorded correctly.

#### 5) FAIRNESS

The final tally of the polling and the partial polling results are published by the BB after the completion of the mixing and tallying phase as described in assumptions (Section III-B2).

After the completion of the polling phase (when PS no longer accepts any votes from the voters), no one is able to compute a partial tally except the POs, who are assumed to be a trusted entity (Section III-A2). Even though POs are assumed to be trusted, distributed decryption is employed such that all POs compute their decryption shares to output a plaintext ballot. Upon completion of the mixing and tallying phase in collaboration with the BB, the plaintext votes and all NIZKPs are published on the BB. Hence, fairness is achieved in SeVEP as long as the BB correctly follows the polling procedure.

#### 6) INTEGRITY OF A PCS

A possible attempt by an authenticated voter to cast multiple votes by manipulating contents of PCS is unsuccessful due to the following reasons: (1) three polling card sheets are sent to the voter in a paper form. In case a PCS is smudged with printing ink, or there are printed marks (black vertical or horizontal line defect), PF discards this PCS; and (2) the text-based PCSs are watermarked using zero-watermarking in which the contents of PCS (6-digit return codes, 6-digit $ACK$, 8-digit $Confirm$, and 16-digit $PCS_{ID}$), PI of the voter along with other information (a logo of PO) are considered as inputs; and (3) a secret key $SK$ based on the text of PCS is constructed by PF, who sends this $SK$ along with the related information (zero-watermark $WM$, logo, current date and time, $GS$, poll ID and encrypted barcode) to PO, who stores all the received information to be used in case of extracting the zero-watermark from PCS. Upon receiving a complaint from a voter that his/her one or all three PCS(s) are damaged either due to an ink spread, or overlapping characters, PO requests that voter to send his/her impaired PCS(s) and poll ID to it, so that an appropriate action can be taken w.r.t the complain. A damaged PCS due to an ink spread is not considered by PO, who discards the complaint of the voter. In the later case, PO performs an extraction algorithm to extract $WM'$. Then, PO compares the extracted $WM'$ with the stored $WM$. If during the embedding procedure, there would have been an overlapping of characters, the $WM$ would have been constructed using those characters. Thus, upon mismatch between $WM$ and $WM'$, PO requests PF to send him/her the PI of that voter, who would then be penalized by PO in cooperation with CI.

#### 7) PRIVACY AND SECURITY ATTACKS

This section discusses several attack scenarios presented in Section III-C4, which may occur during the execution of three phases of SeVEP.

- A possible coalition between $VD_S$ and PS to manipulate voter's voting choices could not affect vote's privacy due to the fact that even if a malicious PS verifies the manipulated encrypted vote in the vote validation phase (Section III-E1), at the next stage, i.e., the ballot processing phase (Section III-E2), the voter would get to know about the malicious activity by finding a mismatch between the received return codes and the printed codes as the received codes would not correspond to his/her intended voting choices. $VD_S$ cannot know about the received return codes by any chance due to the fact that the voter opens the received return codes on his/her validation device.
- A possible collusion attack by PS and CG/PCGs to infer the voter's selected voting options cannot be successful for three reasons: (1) the voting options are encrypted with the joint public key of the polling organizers; (2) the security of the PRF function computed over the voting options to generate the partial return codes is based on the hardness of DDH assumption [25]; and (3) the relation between the return codes and the voting options is only known to the voter due to the fact that the process of PCS generation is distributed among different participants (PF, CG, PCG), and not a single entity has a complete knowledge about the contents (return codes, $ACK$, $Confirm$, $PCS_{ID}$) of PCS. A mapping table stored at CG and $PCG_{\mathbb{X}}$'s end only contains short and long return codes without any information about the voting options. Also, after delivery of PCS to the authenticated voter, PF destroys all PCSs (as described in the security assumptions (Section III-B2)).
- When the confirmation code is sent to the voter, PS may attempt to form a coalition with CG/PCG to replace the confirmed vote with the colluded vote, i.e., by replacing the encrypted voting options with their chosen options (voting options along with 3-digit random number), and partial return codes computed by brute forcing. However, this attack is infeasible due to two reasons: (1) at the end of the post-polling phase, when the polling results are published by the BB, the voter could compute the hash of the published vote, and in case of mismatch, complain to the POs of vote manipulation; and (2) instead of performing hash of the published vote, the voter searches for 3-digit (alphanumeric) random numbers that he/she used for each voting option, and upon not finding these numbers on the BB, he/she could complain to the POs of malicious activity.
- During zero-watermarking of PCS, PF uses $ID_{V_k}$ to construct a zero-watermark. While performing extraction of the zero-watermark, POs obtain $ID_{V_k}$ from PF. Thus, PF and POs may attempt to de-anonymize the voter. $ID_{V_k}$ is generated using a one-way cryptographic hash function $H()$. An attempt of de-anonymization attack by any malicious entity is withstood by the collision resistance of the hash function, i.e., it is computationally infeasible to find a pair $(x, y)$ such that $H(x) = H(y)$. Moreover, for a hash function with $w$-bit hash values, $2^{w/2}$ calculations are required to find a collision with probability $1/2$, which is infeasible for $w \geq 128$. In SeVEP, we have considered SHA-1 with $w = 160$ bits for high security such that it is computationally infeasible for an attacker to compute $2^{80}$ calculations to find a real identity from a pseudo ID. Furthermore, a malicious entity does not know the secret number $r$ shared by the

voter with CI, who is assumed to be a trusted entity of SeVEP, and thus, would not break the privacy of the voter by forming a coalition with PF and POs.

#### 8) BIOMETRIC PROTECTION

It is assumed that biometric samples (voice samples and keystroke patterns) stored at CI's end are protected using hybrid template protection method, which aims to combine the benefits of both feature transformation and biometric cryptosystem approaches.

- In feature transformation method, the unprotected biometric template ($T$) of a voter to be enrolled in the system is transformed into a protected template ($T'$) via a transformation function ($\mathbb{F}$). The transformation function is characterized by a set of voter-specific parameters, which are normally derived from a random external key or password ($\mathbb{K}$). Thereafter, only the protected template, $\mathbb{F}(T, \mathbb{K})$, is stored in the system database. During authentication, the same transformation function, $\mathbb{F}$, and its governing parameters, $\mathbb{K}$, are applied to the unprotected query feature set $Q$, such that matching between the enrolled and query templates occurs in the transformed space, i.e., $\mathbb{F}(T, \mathbb{K})$ is matched against $\mathbb{F}(Q, \mathbb{K})$. The main advantage of this approach is that the key is voter-specific, which incorporates diversity into the protected biometric templates, i.e., in case a protected template is compromised, it can easily be revoked and replaced with a new one by applying a different voter-specific key to the same unprotected biometric data. Furthermore, matching can be done in the transformed domain, which means that biometric templates can remain secure even during authentication.
- Biometric cryptosystems are based on cryptographic security mechanisms that allow cryptographic privacy protection for biometric reference data. A biometric cryptosystem is a key-generating system, where a key is generated directly from the biometric data itself. Error correcting codes are used to deal with the intra-class variance inherent in multiple samples of the same biometric characteristic, and hash functions are used to impart security to the protected biometric templates.

This hybrid biometric template protection scheme addresses the security attacks described in Section III-C6:

- In case an attacker gains an access to the template, the compromised template can be cancelled and replaced with a different protected template from the same original template.
- In the event that a user's protected template is stolen from the database, that stolen template cannot be used to recover the corresponding original template.

### B. PERFORMANCE ANALYSIS

In this section, the performance evaluation of pre-polling and polling phases of SeVEP is provided in terms of computational and cryptographic overheads.

#### 1) COMPUTATIONAL COSTS OF PRE-POLLING AND POLLING PHASES

The pre-polling phase (Section III-D) and the polling phase (Section III-E) of SeVEP are implemented in Matlab and Java programming languages on a workstation equipped with an Intel i-7 processor at 3.5 GHz and 8 GB of RAM to compute the costs of involved cryptographic operations.

In the pre-polling phase, poll configuration, generation of a PCS and a polling tag are implemented in Java, whereas zero-watermarking of PCS is performed in Matlab. The computational costs associated with the multifactor authentication have been left for the future work. The results presented in Table 2 correspond to the average of 100 runs of each operation of the pre-polling phase.

**TABLE 2. Computational costs of pre-polling phase.**

| Process | Entity | Time (ms) |
|---|---|---|
| Poll configuration | PO | 6,519 |
| Generation of $ID_{V_k}$ | CI+$V_k$ | 200 |
| Generation of a PCS | CG+$PCG_{\mathbb{X}}$+PF | 4,086 |
| Generation of a polling tag | PF | 5 |
| Zero-Watermarking of a PCS | PF | 900 |
| Total time | | 8,710 |

The poll configuration process is performed by PO, who performs the following operations in almost 6.50 seconds: generation of safe primes, RSA key pairs for PS, PF, CG, and $PCG_{\mathbb{X}}$, a unique poll ID, a poll public key, four poll questions and corresponding five voting options (small bit-length unique prime numbers), and ElGamal key pair using distributed ElGamal (Section II-A). The pseudo-identity of the voter, generated by a voter with the help of CI using the protocol of [37], is computed in 0.20 seconds. A single PCS containing return codes, *ACK*, *Confirm*, $PCS_{ID}$, and a barcode (1-D) is generated by CG, six PCGS and PF in 4.10 seconds. In order to compute these codes, a joint El-Gamal public key is computed by six PCGs using ElGamal key distribution. Also, each PCG is assumed to have an access to a pool of 50 pre-generated keys of length 1,024-bits. A polling tag is generated by PF in 0.005 seconds by taking hash of three barcodes printed on three PCSs. Here, we have only considered the computational cost of computing hash and excluded the costs of generation of two PCSs. After the generation of PCS, a zero-watermark is embedded into the PCS by PF in 0.90 seconds using the zero-watermarking (Algorithm 1).

The vote validation and ballot processing processes of the polling phase are implemented in Java programming language to compute the computational costs. The overheads presented in Table 3 correspond to an average of 100 runs of each cryptographic operation (PRF with DDH assumption with 1024-bits key, hmacwithSHA256, AES-256, ElGamal and RSA encryption/decryption with 1024-bits) of the polling phase. For the calculation of these costs, it was assumed that the voter has cast his/her vote only once during $t_p$.

It can be seen from the results presented in Tables 2 and 3 that on average, a voter requires less than 6.50 seconds to

**TABLE 3.** Computational costs of the polling phase.

| Phase | Entity | Operations | Time (ms) |
|---|---|---|---|
| $VD_S$ joins with PS | $VD_S$ | TLS | 1100 |
| Polling Phase | $VD_S$ | ElGamal Enc of votes and a random no. | 512 |
| | $VD_S + PS + CG$ | RSA Enc/Dec of voter ID | 2/18 |
| | $VD_S + CG$ | Partial return codes Gen | 61 |
| | $VD_S$ | RSA Enc/Dec of partial return codes | 15/112 |
| | $VD_S$ | RSA Sig on ballot contents | 65 |
| | $VD_S$ | NIZKPs Gen | 108 |
| | PS | RSA Sig/NIZKPs Ver | 47/602 |
| | $PCG$ | Full return codes Gen | 11 |
| | $PCG$ | ElGamal Enc of long and short codes | 1912 |
| | $CG + PS$ | RSA Enc/Dec of short codes | 9/21 |
| | $CG + VD_S$ | RSA Enc/Dec of $ACK$ | 0.2/5 |
| | $CG + VD_S$ | RSA Enc/Dec of $Confirm$ | 0.1/6 |

get registered to SeVEP (excluding multifactor authentication process), and 4.00 seconds (excluding the costs of processes involved in pre-poll registration, i.e., computing safe primes, ElGamal key distribution, RSA keys generation etc., and the costs of communication between the involved entities) to cast his/her vote, thus, demonstrating the practicality of the proposed system.

### 2) CRYTOGRAPHIC COSTS OF PRE-POLLING AND POLLING PHASES

In the poll configuration process of the pre-polling phase, generation of a RSA key pair requires 2 modular exponentiations, thus, resulting in 18 exponentiations for total 9 key pairs (PF, CG, six PCGs, and PS). Similarly, generation of ElGamal public key requires 1 modular exponentiation, which sums up to 2 exponentiations (one for $t$ POs and other for six PCGs). The generation of voter's pseudo-identity in cooperation with CI takes 1 modular exponentiation. In PCS generation, the cryptographic costs are calculated for the following operations: (i) each PCG computes a partial return code, which requires 1 exponentiation and $M$-modular multiplications of each voter selection to a secret (pre-generated) key, thus, resulting in 6 exponentiations; (ii) six full return codes are encrypted with ElGamal encryption algorithm (with the joint public key of PCG) that requires 2 exponentiations each, which sums up to 12 exponentiations; (iii) the short and the encrypted long return codes, the secret key of each PCG, and *ACK* are encrypted with RSA encryption algorithm (with the public key of PF) that requires 1 exponentiations to generate a ciphertext, thus, resulting in 19 modular exponentiations. Similarly, RSA decryption algorithm requires 1 exponentiation to decrypt the ciphertext, thus, resulting in 19 exponentiations. $PCS_{ID}$ is generated by summing up six secret keys of PCGs, which result in 1 modular exponentiation; (iv) each pair of the mapping table, and *Confirm* code is encrypted with ElGamal encryption algorithm (with the joint public key of PCG and public key of CG, respectively) that requires 2 exponentiations, thus, resulting in total 42 exponentiations; and (v) the entries of the mapping table, the encrypted long return codes, and *Confirm* code are decrypted using ElGamal decryption that requires 1 modular exponentiation, which results in total 41 exponentiations.

In the pre-polling phase, the PCS generation is an expensive process in terms of cryptographic costs, but this process is performed by CG, six PCGs and PF in an offline mode, thus, these costs are not included in the total overheads of the voter.

In the polling phase, the product of selected voting options, a three-digit random number, and the long and short return codes are encrypted with ElGamal encryption algorithm, which requires 2 exponentiations each, thus, resulting in 20 modular exponentiations. The voter's pseudo-identity, the partial codes, the short return codes, *ACK*, and *Confirm* codes are encrypted with the RSA encryption algorithm that requires 1 exponentiation to generate a ciphertext, thus, resulting in total 11 exponentiations. Similarly, RSA decryption algorithm requires 1 exponentiation to decrypt the ciphertext and, therefore, sums up to 11 exponentiations. The computation of partial return codes requires 1 exponentiation and $M$-modular multiplications of each voter selection (to $PCS_{ID}$), which sums up to 4 exponentiations (with one option per 4 questions). $VD_S$ computes two following NIZKPs: (1) Schnorr identification protocol; and (2) Chaum-Pederson protocol. The generation of first proof requires 1 modular exponentiation (total of 2 exponentiations for generating ElGamal ciphers) and its verification requires 2 exponentiations, which sum up to 4. Generation of Chaum-Pederson proof requires 2 modular exponentiations, which implies that it takes 8 exponentiations for 4 partial return codes. The verification of Chaum-Pederson protocol requires 4 exponentiations, which result in total 16 modular exponentiations. The RSA algorithm used to digitally sign the ballot contents requires 1 modular exponentiation for signature generation and 1 modular exponentiation for signature verification, thus resulting in total 2 exponentiations.

It can be observed from the given cryptographic costs of the polling phase that $VD_S$ does not need to perform the most expensive cryptographic operations, i.e., ElGamal Enc of long and short return codes (total 16 exponentiations), and NIZKP verification (total 20 exponentiations), which demonstrates the feasibility of implementation of the polling phase on the light-weight computing devices such as smartphones.

**TABLE 4.** Comparison with other e-voting schemes.

| E-Voting Systems | Security Properties | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Voter MFA | Multi-voting | Vote Privacy | Coercion Resistance | Cast-as-intended Verifiability | Recorded-as-cast Verifiability | Tallied-as-recorded Verifiability | Robustness | Untrusted Platform |
| Adida [12] | No | Yes | No | No | Yes | Yes | Yes | No | No |
| Haenni and Koenig [11] | No | No | No | No | Yes | No | No | No | No |
| Haenni *et al.* [18] | No | No | Yes | Yes | No | Yes | Yes | Yes | No |
| Galindo *et al.* [16] | No | No | Yes | No | Yes | Yes | Yes | Yes | No |
| Gøjsteen [17] | Yes | Yes | Yes | No | Yes | No | Yes | Yes | No |
| SeVEP | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

### C. COMPARISON WITH STATE-OF-THE-ART E-VOTING SYSTEMS

This section carries out a comparative analysis of SeVEP with state-of-the-art e-voting systems that provide similar security properties of e-voting as SeVEP. The comparison is performed in terms of security properties and cryptographic costs.

#### 1) COMPARISON IN TERMS OF SECURITY PROPERTIES

The security properties of SeVEP are compared against relevant e-voting systems [11], [12], [16]–[18]. The comparison, summarized in Table 4, considers the desirable properties described in Section III-B. In Table 4, a cell contains "No" when the corresponding security property is not guaranteed by the voting system.

From Table 4, the following observations can be made with respect to security properties:

- In terms of authentication, all the systems except [17] and SeVEP, consider the traditional password-based authentication technique to authenticate a user, which is vulnerable to guessing, brute-force, and phishing attacks. In [17], an authentication platform (commonly used for online services) is employed that uses two-factor authentication (2FA), i.e., password and OTP to authenticate the voter. Though OTP used in 2FA provides security against traditional security attacks, it is vulnerable to malware and session replay attacks. SeVEP mitigates this vulnerability by adding a third layer of authentication based on biometric recognition.
- All the voting systems except SeVEP, [12] and [17] allow single voting only. In [12], the voter has an option to either audit or submit his/her votes. If the voter chooses to audit, he/she can review the displayed ciphertext to verify that the voting platform correctly saved his/her votes. The voter can select different options until he/she decides to cast his/her ballot by sealing it using a different ciphertext. The system proposed in [17] provides its voters an alternative vote casting channel such as postal mail or a local polling station to cast a vote if they had encountered any problem during online voting. The other voting systems based on return codes do not offer multiple voting whereas SeVEP that is based on return-codes allows the voter to vote three times within the allowed polling period. In [15], multi-voting is assumed by the authors such that the voter can choose to cast his/her vote from a different voting device computer) in case of incorrect generation of the return codes.
- Vote privacy is provided by all the voting systems except for [11] and [12]. In [12], the attackers may copy and re-cast previously cast votes, and a corrupted authority may distribute identical receipts to different voters while modifying their actual votes undetected. In [11], the trusted hardware device can learn about the voting choices made by the voter.
- Except for [18] and SeVEP, the remaining voting systems do not offer coercion resistance. The scheme in [18] is based on oblivious transfer, where the security of the mechanism is provided by the fact that the voting authorities (though they may know all the codes) do not know which codes are actually transfered to the voter, thus, providing somewhat protection against vote buying. SeVEP offers coercion resistant by allowing the voter to vote three times using a different code sheet within an allowed polling period (the security against coercibility is discussed in Section IV-A1).
- All the systems except [18] provide cast-as-intended verifiability by either employing voting cards or return codes. Haenni *et al.* [18] have proposed cast-as-intended verification mechanism based on oblivious transfer protocol, however, Brelle and Truderung [23] discovered a serious flaw in the proposed scheme: an attacker can cast invalid ballots and provide valid return codes to the voters. These invalid ballots are accepted by the voting server, tallied, and only discovered and rejected after tallying, when the link between the ballot and the voter has been closed. This flaw can be removed by adding NIZKPs in ballot formation.
- All voting systems except [11] satisfy recorded-as-cast and tallied-as-recorded verifiability.
- Security of a voter and the voting system against malicious attacks (either by external or internal attackers) is provided by all the voting systems except [12] and [11], which pose a range of vulnerabilities, e.g., [12] is susceptible to a vote stealing attack that may allow an attacker to surreptitiously cast a ballot on voter's behalf. In [11], voters must trust the hardware device for vote privacy. In case of a tampered device, voters would have no means of verifying the cast votes.
- Except for SeVEP, all systems assume that the device used for voting is either trusted or uncompromised.

**TABLE 5. Comparison in terms of cryptographic costs with other e-voting schemes.**

| E-Voting Systems | Cryptographic Costs | |
|---|---|---|
| | Voting Device | Voting Authorities |
| Haenni *et al.* [18] | 7 | 8 |
| Galindo *et al.* [16] | 15 | 23 |
| Gøjsteen [17] | 12 | 20 |
| SeVEP | 11 | 22 |

### 2) COMPARISON IN TERMS OF CRYPTOGRAPHIC COSTS

In Table 5, comparison between SeVEP and the three most relevant e-voting systems is presented. In this comparison, we have selected only those voting systems that provide verifiability based on the return codes or voting codes. Table 5 compares the cryptographic costs (in terms of modular exponentiations) of each system's vote casting phase (polling phase) that involves a voting platform (device) and the voting authorities.

In the light of the results presented in Table 5, the overall cryptographic costs of [16], [17] and SeVEP are similar. The voting system in [18] incur a few cryptographic costs both at the voting device and the authorities end. However, the cast-as-intended mechanism can easily be attacked due to missing NIZKPs in the vote casting phase. The security flaw can be prevented by adding NIZKPs, which would degrade the protocol's performance (it would require quadratic time computations).

Among the analyzed systems, the proposed system, SeVEP is the one that satisfies the desirable security properties and offers computationally feasible solution for light-weight devices.

## V. CONCLUSIONS AND FUTURE WORK

This paper described SeVEP, an electronic polling system for small to medium sized Internet-based public opinion systems that provides privacy of vote, voter's anonymity, voter's authentication, auditability, poll integrity, security against coalition of malicious parties, double-voting prevention, fairness, and coercion resistance, and prevents malware-infected voting device from manipulating the authenticated voter's voting choices. In addition, SeVEP provides cast-as-intended verifiability based on cryptographic primitives, which are used to design a complex voting interaction between the voting device, the polling server, the code generator and six polling code generators during the polling phase. Compared to the other state-of-the-art e-voting systems, SeVEP ensures voter's authenticity via multifactor authentication scheme, supports multiple voting, prevents double voting through a polling tag, offers verifiability in the presence of an untrusted voting device, requires less trust assumptions on involved entities, and offers computationally feasible solution for implementation on portable communication devices.

As future work, we intend to develop a working prototype of SeVEP, and evaluate its scalability and usability in real-world deployment. Since the polling phase of SeVEP requires voter's participation, real feedback from usability tests could help in improving the design of SeVEP. In addition, the voters will be provided a poll containing more than 4 questions ($\leq$ 10) will be designed to evaluate the performance in terms of the response time during the polling phase.

## REFERENCES

[1] Association Headquarters. (2018). *What are Public Opinion Polls?* Accessed: Jan. 31, 2019. [Online]. Available: https://www.historians.org/about-aha-and-membership/aha-history-and-archives/gi-roundtable-series/pamphlets/em-4-are-opinion-polls-useful-(1946)/what-are-public-opinion-polls

[2] A. Wells. *UK Polling Report*. Accessed: Jan. 31, 2019. [Online]. Available: http://ukpollingreport.co.uk/

[3] R. Finle and C. Finley. (1999). *Survey Monkey*. Accessed: Jan. 31, 2019. [Online]. Available: https://www.surveymonkey.com/

[4] M. Blumenthal. (2018). *Trump Approval Poll*. Accessed: Jan. 31, 2019. [Online]. Available: https://www.surveymonkey.com/curiosity/trump-approval-poll/

[5] A. Schneider, C. Meter, and P. Hagemeister. (2017). "Survey on remote electronic voting." [Online]. Available: http://arxiv.org/abs/1702.02798

[6] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[7] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2008, pp. 354–368.

[8] J. Benaloh, R. Rivest, P. Y. A. Ryan, P. Stark, V. Teague, and P. Vora. (2013). *End-to-End Verifiability*. Accessed: Jan. 31, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/end-end-verifiability/

[9] F. Zagórski, R. T. Carback, D. Chaum, J. Clark, A. Essex, and P. L. Vora, "Remotegrity: Design and use of an end-to-end verifiable remote voting system," in *Applied Cryptography and Network Security*, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds. Berlin, Germany: Springer, 2013, pp. 441–457.

[10] D. Chaum *et al.*, "Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes," in *Proc. Electron. Voting Technol.*, 2008, pp. 1–13.

[11] R. Haenni and R. E. Koenig, "Voting over the Internet on an insecure platform," in *Design, Development, and Use of Secure Electronic Voting Systems*, D. Zissis and D. Lekkas, Eds. Hershey, PA, USA: IGI Global, 2014, pp. 62–75.

[12] B. Adida, "Helios: Web-based open-audit voting," in *Proc. Secur. Symp. (SS)*, 2008, pp. 335–348.

[13] A. Kiayias, M. Korman, and D. Walluck, "An Internet voting system supporting user privacy," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2006, pp. 165–174.

[14] J. P. Allepuz and S. G. Castelló, "Cast-as-intended verification in Norway," in *Proc. 5th Conf. Electron. Voting*, 2012, pp. 49–63.

[15] J. P. Allepuz and S. G. Castelló, "Internet voting system with cast as intended verification," in *E-Voting and Identity*, A. Kiayias and H. Lipmaa, Eds. Berlin, Germany: Springer, 2012, pp. 36–52.

[16] D. Galindo, S. Guasch, and J. Puiggalí, "2015 Neuchâtel's cast-as-intended verification mechanism," in *E-Voting and Identity*, R. Haenni, R. E. Koenig, and D. Wikström, Eds. New York, NY, USA: Springer-Verlag, 2015, pp. 3–18.

[17] K. Gjøsteen, "The norwegian Internet voting protocol," in *E-Voting and Identity*, A. Kiayias and H. Lipmaa, Eds. Berlin, Germany: Springer, 2012, pp. 1–18.

[18] R. Haenni, R. E. Koenig, and E. Dubuis, "Cast-as-intended verification in electronic elections based on oblivious transfer," in *Electronic Voting*, R. Krimmer, M. Volkamer, J. Barrat, J. Benaloh, N. Goodman, P. Y. A. Ryan, and V. Teague, Eds. Cham, Switzerland: Springer, 2017, pp. 73–91.

[19] R. Joaquim, C. Ribeiro, and P. Ferreira, "VeryVote: A voter verifiable code voting system," in *E-Voting and Identity*, P. Y. A. Ryan and B. Schoenmakers, Eds. Berlin, Germany: Springer, 2009, pp. 106–121.

[20] A. Qureshi, D. Megías, and H. Rifà, "VSPReP: Verifiable, secure and privacy-preserving remote polling with untrusted computing devices," in *Future Network Systems and Security*, R. Doss, S. Piramuthu, and W. Zhou, Eds. Cham, Switzerland: 2018, pp. 61–79.

[21] C. Z. Acemyan, P. Kortum, M. D. Byrne, and D. S. Wallach, "Usability of voter verifiable, end-to-end voting systems: Baseline data for Helios, Prêt à voter, and scantegrity II," in *Proc. Electron. Voting Technol. Workshop/Workshop Trustworthy Electron. (EVT/WOTE)*, 2014, pp. 26–56.
[22] F. Karayumak, M. M. Olembo, M. Kauer, and M. Volkamer, "Usability analysis of Helios—An open source verifiable remote electronic voting system," in *Proc. Electron. Voting Technol./Workshop Trustworthy Electron. (EVT/WOTE)*, 2011, p. 5.
[23] A. Brelle and T. Truderung, "Cast-as-intended mechanism with return codes based on PETs," in *Electronic Voting*, R. Krimmer, M. Volkamer, N. B. Binder, N. Kersting, O. Pereira, and C. Schürmann, Eds. Cham, Switzerland: Springer, 2017, pp. 264–279.
[24] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *J. Cryptol.*, vol. 20, no. 1, pp. 51–83, 2007.
[25] M. Naor, B. Pinkas, and O. Reingold, "Distributed pseudo-random functions and KDCs," in *Advances in Cryptology—EUROCRYPT*, J. Stern, Ed. Berlin, Germany: Springer, 1999, pp. 327–346.
[26] H. Krawczyk, M. Bellare, and R. Canetti. (1997). *HMAC: Keyed-Hashing for Message Authentication*. Accessed: Jan. 31, 2019. [Online]. Available: https://tools.ietf.org/html/rfc2104
[27] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO*, A. M. Odlyzko, Ed. Berlin, Germany: Springer, 1987, pp. 186–194.
[28] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.
[29] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology—CRYPTO*, E. F. Brickell, Ed. Berlin, Germany: Springer, 1993, pp. 89–105.
[30] B. Terelius and D. Wikström, "Proofs of restricted shuffles," in *Progress in Cryptology—AFRICACRYPT*, D. J. Bernstein and T. Lange, Eds. Berlin, Germany: Springer, 2010, pp. 100–113.
[31] J. Bäck. (2011). *RSA-PSS—Provable Secure RSA Signatures and Their Implementation*. Accessed: Jan. 31, 2019. [Online]. Available: https://rsapss.hboeck.de/rsapss.pdf
[32] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin, "The design and analysis of graphical passwords," in *Proc. USENIX Secur. Symp. (SSYM)*, vol. 8, 1999, p. 1.
[33] C. Shen, Z. Cai, X. Guan, Y. Du, and R. A. Maxion, "User authentication through mouse dynamics," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 16–30, Jan. 2013.
[34] J. R. M. Filho and E. O. Freire, "Multimodal biometric fusion—Joint typist (keystroke) and speaker verification," in *Proc. Int. Telecommun. Symp.*, Sep. 2006, pp. 609–614.
[35] H. Ishizuka, I. Echizen, K. Iwamura, and K. Sakurai, "A zero-watermarking-like steganography and potential applications," in *Proc. 10th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Aug. 2014, pp. 459–462.
[36] A. Qureshi, D. Megías, and H. Rifà-Pous, "Framework for preserving security and privacy in P2P content distribution systems," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1391–1408, 2015.
[37] A. Qureshi, D. Megías, and H. Rifà-Pous, "PSUM: Peer-to-peer multimedia content distribution using collusion-resistant fingerprinting," *J. Netw. Comput. Appl.*, vol. 66, pp. 180–197, May 2016.
[38] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 2, pp. 93–108, Apr./Jun. 2005.
[39] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. Knowl. Discovery Data Mining (AAAIWS)*, 1994, pp. 359–370.
[40] D. A. Reynolds, "Gaussian mixture models," in *Encyclopedia Biometrics*. New York, NY, USA: Springer, 2009.
[41] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

**AMNA QURESHI** received the Ph.D. degree in network and information technologies from the Universitat Oberta de Catalunya, in 2014, where she joined the Internet Interdisciplinary Institute, in 2015, as a Researcher. Her research interests include intellectual property protection of multimedia content in peer-to-peer networks (content distribution), privacy preservation of the users of these networks, security of the cloud infrastructure operations for the purpose of achieving an end-to-end confidentiality, and design and analysis of security and privacy techniques for smart cities.

**DAVID MEGÍAS** is currently an Associate Professor with the Universitat Oberta de Catalunya (UOC), Barcelona, Spain, and also the current Director of the Internet Interdisciplinary Institute, UOC. He has authored many research papers in international conferences and journals. He has participated in different national research projects both as a Contributor and as a Principal Investigator. He has also experience in international projects, such as the European Network of Excellence of Cryptology of the 6th Framework Program of the European Commission. His research interests include security, privacy and protection of multimedia contents, privacy in decentralized networks, and information security.

**HELENA RIFÀ-POUS** received the Ph.D. degree in telecommunications engineering from the Universitat Politècnica de Catalunya, in 2008. From 2000 to 2007, she was with Safelayer Secure Communications as a Research Project Manager, focused on PKI projects mainly for the public administration. She has been an Associate Professor with the Computer Science Department, Universitat Oberta de Catalunya (UOC), since 2007. She has been the Director of the Master Programme in Security of Information and Communication Technologies, UOC, since 2011. Her research interests include network security and privacy in distributed and mobile networks, key management, information hiding, and security and privacy in multimedia content distribution.