On the Feasibility of E2E Verifiable Online Voting – A Case Study From Durga Puja Trial

Horia Druliac[1], Matthew Bardsley[1], Chris Riches[1], Christian Dunn[1], Luke Harrison[1], Bimal Roy[2], and Feng Hao[1]

[1] University of Warwick, United Kingdom
[2] Indian Statistical Institute, India

Abstract. India is the largest democracy by population and has one of the largest deployments of e-voting in the world for national elections. However, the e-voting machines used in India are not end-to-end (E2E) verifiable. The inability to verify the tallying integrity of an election by the public leaves the outcome open to disputes. E2E verifiable e-voting systems are commonly regarded as the most promising solution to address this problem, but they had not been implemented or trialed in India. It was unclear whether such systems would be usable and practical to the Indian people. Previous works such as Helios require a set of tallying authorities (TAs) to perform the decryption and tallying operations, but finding and managing TAs can prove difficult. This paper presents a TA-free E2E verifiable online voting system based on the DRE-ip protocol. In collaboration with the local authority of New Town, Kolkata, India, we conducted an online voting trial as part of the 2022 Durga Puja festival celebration, during which residents of New Town were invited to use mobile phones to vote for their favourite pujas (festival decorations) in an E2E verifiable manner. 543 participants attended the Durga Puja trial and 95 of them provided feedback by filling in an anonymous survey after voting. Based on the voter feedback, participants generally found the system easy to use. This was the first time that an E2E online voting system had been built and tested in India, suggesting its feasibility for non-statutory voting scenarios.

1 Introduction

India is the largest democracy in the world by population, and one of the earliest adopters of electronic voting (e-voting) for national elections. E-voting machines, known as Electronic Voting Machines (EVMs), were first used in elections in India in 1998, and today, they have replaced paper ballots in all state and general elections, and almost all local elections. Twenty years of deployment of e-voting in legally binding elections in India has built up an asset of experience and provided the world with useful lessons.

One of the most important lessons concerns the security of e-voting. With the existing design of EVM, voters must trust that the machine functions correctly and honestly, but they cannot independently verify if that is the case [25]. In

a civic poll in Uttar Pradesh in November 2017, some voters discovered that an EVM always recorded votes for a fixed candidate, irrespective of the button that was being pressed [19]. Officials attributed this to a "malfunction" and replaced the affected machines. Similar malfunctions had been reported before, e.g., during elections in Maharashtra in July 2017 [16]. These instances of EVM malfunction inevitably raise the question: how can it be assured that a similar malfunction does not happen in an election in an undetectable way?

One promising solution, widely supported by the research community, is to make an e-voting system end-to-end (E2E) verifiable [13]. Here, the E2E verifiability refers to voters being able to verify their votes are *cast-as-intended*, *recorded-as-cast* and *tallied-as-recorded*. The malfunction cases reported in Uttar Pradesh and Maharashtra show an example of not fulfilling the *cast-as-intended* requirement. Yet, despite the advancement of e-voting research, E2E verifiable voting systems had not been implemented or trialled in India. It was unclear whether such systems would be usable and practical for Indian people.

We reached out to the Election Commission of India (ECI) with a proposal to validate the feasibility of E2E verifiable e-voting in the country. Subsequent meetings with the IT secretary of the West Bengal state were arranged to discuss suitable trial opportunities. However, instead of doing a trial as part of a statutory election, it was suggested that the trial be conducted for an informal non-statutory election to build a case study first. One opportunity that arose from the discussions was to conduct an online voting trial as part of the annual Durga Puja festival celebration at New Town, Kolkata.

New Town is a modern satellite city of Kolkata in the West Bengal state. Durga Puja is one of the most important festivals in India, especially in Kolkata. From 2019, the research team had several meetings with New Town Kolkata Development Authority (NKDA), the local governing body of New Town, to gather the design requirements for an E2E verifiable online voting system to be used in the trial. In 2022, in-person celebrations for Durga Puja were revived in many parts of India including New Town after the COVID-19 pandemic. For the first time, residents of New Town were provided with an opportunity to vote for their favourite pujas (festival decorations) online in an E2E-verifiable manner.

In this paper, we present an E2E verifiable online voting system based on Shahandashti and Hao's DRE-ip protocol [21] (with adaptations). Compared with other E2E voting schemes, the DRE-ip protocol has a distinctive feature that it is free from any tallying authorities (TAs). As we will show, the removal of TAs significantly simplifies election management. The system development started from scratch and took about a year, comprising around 25,000 lines of code. Voters cast votes using mobile phones or computers through a public web interface[3]. The backend code including all cryptographic implementations is available as open-source[4]. In collaboration with NKDA, we successfully conducted an online voting trial among the New Town residents as part of the Durga Puja festival celebration. To the best of our knowledge, this is the first time that

[3] `https://newtowndurgapuja.in/`

[4] `https://github.com/DRE-ip-Implementation-Team/dre-ip-backend/`

an E2E online voting system has been implemented and trialled in India. We explain the details of the voting system design (§3), the implementation (§4), the Durga Puja trial (§5) and the survey results (§6). This trial serves to provide a case study that demonstrates the feasibility of E2E online voting, in place of vanilla (or black-box) online voting, for an informal non-statutory election. We leave the study of the suitability of E2E online voting, as an alternative choice of postal voting, for statutory elections to future research.

2 Background

In general, there are two types of voting applications: local and remote voting. The former is conducted locally at polling stations in a *supervised* environment, while the latter is conducted remotely in an *unsupervised* environment. In either case, voters may use paper ballots or electronic means. In this section, we will focus primarily on remote voting. Table 1 compares several remote voting systems in terms of the E2E verifiability requirements.

Postal voting (also called mail-in voting) is a popular form of remote voting legally permitted in many democratic countries. In the UK, about 20% of the electorate choose postal voting in general elections. In other countries, the proportion of the electorate which choose postal voting has been steadily increasing [23]. In postal voting, a voter is able to verify that a vote is cast as intended (since they manually write the candidate choice). However, a voter cannot verify how a paper ballot will be recorded and tallied after it is posted.

Internet voting allows voters to vote online through a web browser. A vanilla version of Internet voting simply asks voters to choose candidates on a web page and tallies the submitted votes on the server. Voters are required to completely trust the system, but their electronic ballots, as well as the tally, may be maliciously changed without their awareness. Such a vanilla system is also called a "black-box" e-voting system [11]. Examples include those used for 2022 IEEE Annual Election and the 2022 Tory party leadership election. In these elections, voters had no means to independently verify the tallying integrity.

A prominent online voting system that has been used for national elections is Estonian Internet voting [13]. In 2005, Estonia became the first country that allowed Internet voting for national elections, and today about 50% of the ballots are cast online. As compared with a vanilla system, the Estonian online voting system is partially verifiable – in particular, a voter can verify that their vote is cast-as-intended (by forcing the voting device to reveal the randomness used in the encryption of a ballot), but not that it is correctly recorded-as-cast and tallied-as-recorded [13]. Voters must trust the server for the tallying integrity. If the server is compromised, the attacker can freely modify the tally without being detected, as demonstrated by Springall et al. [22].

E2E verifiable e-voting systems are commonly regarded as the most promising solution to protect the tallying integrity since by design they must satisfy all three verifiability requirements (see Table 1). Well-known E2E voting systems include Chaum's voter verifiable scheme, Prêt à Voter, Scantegrity, Helios and

Scheme	Postal voting	Vanilla online voting	Estonian online voting	E2E verifiable online voting
Cast-as-intended	✓	✗	✓	✓
Recorded-as-cast	✗	✗	✗	✓
Tallied-as-recorded	✗	✗	✗	✓

Table 1: Comparison of remote voting systems in terms of verifiability

STAR-vote [13]. All of these schemes require a set of Tallying Authorities (TAs) to perform decryption and tallying operations. However, finding and managing such TAs has proved difficult [2]. This motivates designing E2E voting systems that are TA-free, e.g., DRE-i [11] and DRE-ip [21]. DRE-i chooses to pre-compute encrypted ballots but the pre-computed ballots need to be securely stored. A compromise of the storage will reveal the secrecy of all ballots. By comparison, DRE-ip encrypts ballots in real-time, and can provide a stronger guarantee of ballot secrecy: when the system is completely compromised, while the tallying integrity remains intact due to E2E verifiability, only the partial tally at the time of compromise is revealed, representing minimum information leakage [21]. While DRE-i and DRE-ip are designed to only support the simple plurality voting scheme, they can be extended to support more complex ranked-choice voting schemes such as Borda count [3], Condorcet [15] and Instant Run-off [14] without requiring TAs. In this work, we are only concerned with plurality voting.

3 System Design

This section explains the requirements for the Durga Puja online voting system, the architectural design and the authentication mechanism.

3.1 Requirements

Based on the meetings with NKDA, the following requirements were specified.

1. *E2E verifiability* - The online voting system shall be E2E verifiable (which is the main goal of the trial).
2. *Admin automation* - The election management should be automated without TAs (there are no resources from NKDA to serve as any TA).
3. *Mobile phone friendliness* - A voter should be able to vote from a mobile phone as long as it has Internet access. They can also vote from computers.
4. *Option of verification* - The option to verify votes should be available to voters, but not imposed on them as mandatory. This is for usability reasons.
5. *No registration* - No registration is required; anyone with a valid India mobile number can participate in this trial. This is to facilitate participation.
6. *Support for two categories* - There are two communities in New Town called *housing* and *block*, which correspond to two categories of voting. A voter needs to choose one of the two categories and vote within the chosen category.

Satisfying (1-4). To achieve E2E verifiability we chose DRE-ip [21], which does not require any TAs. The removal of TAs significantly simplifies election management. After the election parameters (i.e., voting questions, candidates, start/end time) are specified, the remaining management process is automated in a publicly verifiable manner. As soon as the election ends, the tally is instantly available, together with publicly available audit data to enable anyone to verify the tallying integrity. As the cryptographic operations are performed on the server according to DRE-ip, a voter can use a plain browser on a resource-constrained device such as a mobile phone to vote. In DRE-ip, the voter verification step has been naturally integrated into the voting process as a 'cancel/confirm' step, which gives every voter an option to verify votes.

Satisfying (5). To facilitate remote participation in this trial, we adopt a single-factor (token) authentication scheme. The voter provides a mobile phone number, and proves the ownership of that number by entering a 6-digit One-Time Passcode (OTP) that the system sends to that number through SMS. The system formats the provided number according to the international phone number formatting standard (E.164) and accepts only valid India numbers (international code +91). To preserve privacy, we do not store any phone number in plaintext on our system. Instead, we store an HMAC tag $t = HMAC_k(\text{“phone number”})$ where k is a HMAC key (stored in the system together with the private signing key used to sign data for publication on the bulletin board). This allows us to check if the phone number has been used before without storing it in plaintext. To address the threat of a bot guessing OTP, we add reCAPTCHA (v2) as part of the authentication process. If a voter has more than one mobile phone number, they can cast a vote using each different mobile number. This is considered acceptable in the context of this trial (there is no award for the winners, except an honourable mention of their names on the NKDA website). In a real election, proper registration is required and then “one-man-one-vote” can be enforced by checking the phone number against a list of registered numbers. Furthermore, instead of using SMS (which is being replaced by a token-based authenticator in many applications), alternative schemes based on multiple factors (e.g., password + authenticator) can be used to strengthen authentication in real elections.

Satisfying (6). To fulfill Requirement 6 and to make the system flexible and generally useful, we define *groups* of voters, each of which can vote on a subset of questions. Different groups may be mutually exclusive or overlap. In the most common case, the electorate contains only one group. The system is designed to support any number of groups. When we create an election, we can assign specific voting questions to specific groups. For the New Town Durga Puja trial, we define two mutually exclusive groups (block/housing). A voter must choose to belong to one of the two groups, and once the choice is confirmed, it cannot be changed. Voters in each group elect the best puja within their respective group.

3.2 Architecture

The voting system can be subdivided into two primary components: an election server, and a web voting interface. The server implements cryptographic func-

tions of DRE-ip using Rust, in conjunction with MongoDB for backend storage. The use of Rust has the advantages of eliminating classes of bugs (e.g., memory safety bugs) at compile-time, and supporting concurrent requests efficiently. The front-end interface adopts the open-source JavaScript framework Vue.js to provide a single-page web application. Each page is considered as a view, composed of different components. Each component is split into three sections: template (HTML), styling (CSS) and behavior (JavaScript). This allows for designing web pages in a more organized manner. Among numerous CSS frameworks available, we chose Tailwind, which is lightweight, offers the freedom to create a unique user interface, and is compatible with mobile devices. The voting interface interacts with the back-end of the system by issuing requests for resources (read/write) or voter actions to the election server. All the requests are run over HTTPS.

To allow portable deployment of the system on any platform, we use Docker, a platform-independent service for application deployment. The use of Docker separates applications into containers that can interact with each other, and ensures that services can run identically regardless of the host operating system. Dependencies are automatically managed, and environments are encapsulated. We deploy the e-voting system as a multi-container Docker application on a Virtual Private Server (VPS) in a public cloud.

3.3 Authentication

The voting system distinguishes three types of users: 1) a *coordinator*, who has the administrative right to create elections; 2) a *voter*, who has the right to cast one ballot; and 3) a *public member*, who is free to view a public bulletin board page without any authentication. In our implementation, a *coordinator* is authenticated by a username and password. A *voter* is authenticated based on proving the ownership of a mobile phone number, i.e., providing an OTP sent to that mobile phone. A *public member* does not need authentication.

4 Durga Puja Online Voting System

The Durga Puja online voting system is based on implementing DRE-ip with adaptations according to the requirements in Section 3.1. The system implementation consists of three phases: setup, voting and result, as we detail below.

4.1 Setup Phase

The original DRE-ip protocol [21] is specified in a DSA-like multiplicative group for a single-candidate election. But in our system, we choose to implement it using an additive group over an elliptic curve for better efficiency. We extend the single-candidate specification by running it in parallel for multiple candidates with an additional zero-knowledge proof (based on Chaum and Pedersen [9]) to enforce that only one candidate may be selected from a list. Let $E(F_q)$ be an elliptic curve defined over a finite field F_q where q is a large prime. The system

works in the subgroup over $E(F_q)$ of prime order p. We adopt the NIST P-256 elliptic curve, which is well supported in Rust. (Alternative curves such as Curve25519 can also be used.) The setup of an election involves deriving two generators: namely, G_1 and G_2, whose discrete logarithm relation is unknown to anyone. We define the standard generator from NIST P-256 as G_1 and obtain G_2 by using a `hash_to_curve` algorithm[5] with election-specific data (e.g., election name, start/end time, candidates etc) as the input to the hash function. On the P-256 curve (where the co-factor is 1), any non-identity point can serve as a generator. Hence, computing G_2 is straightforward.

The setup phase also involves generating a pair of digital signing keys using ECDSA for each election. The ECDSA keys are used to ensure the authenticity of the data published on the bulletin board. We choose the same NIST P-256 curve for implementing ECDSA. The public key is published on the public bulletin board before the election. During an election, all the audit data published on the bulletin board for voter verification is digitally signed by the corresponding ECDSA private key for proving the authenticity of the published data. Once the coordinator defines the parameters for an election, including the election title, start/end time, questions and candidates, the subsequent management of an election is automated. The voting phase will start and end automatically, with publicly verifiable audit data published on the bulletin board to allow every voter to verify the tallying integrity, as we explain in the next phase.

4.2 Voting Phase

A voter must first pass the authentication as described in Section 3. They then need to choose whether they vote in the "block" or "housing" group based on the requirements in Section 3.1. For each group, voters are presented with the same voting question (defined by NKDA): "Best Puja under citizen choice". They are allowed to choose only one candidate from the given list in the respective group.

The original DRE-ip protocol is specified for a single-candidate election, in which each voter casts one vote $v_i \in \{0, 1\}$ (which corresponds to 'No' and 'Yes' respectively). In our implementation, we extend it to support multiple candidates by running the single-candidate elections in parallel for all k candidates and adding a zero-knowledge proof (ZKP) to constrain that only one candidate out of k may be chosen. Each ballot is uniquely identified by an index number i. In our implementation, we chose the index number i to be incremental for each ballot. Each ballot i contains $v_{ij} \in \{0, 1\}$ $(j = 0, 1, \ldots, k-1)$ votes for k candidates. When the voter selects a candidate, their ballot is generated by the server as follows. For each candidate $j = 0, 1, \ldots, k-1$:

1. Assign $v_{ij} = 1$ if the voter has chosen the jth candidate; $v_{ij} = 0$ otherwise.
2. Compute an encrypted vote for the jth candidate, consisting of $Z_{ij} = G_1 \cdot (r_{ij} + v_{ij})$ and $R_{ij} = G_2 \cdot r_{ij}$ where $r_{ij} \in_R [0, p-1]$ is a random secret, and '$+/\cdot$' are addition/multiplication operations on an elliptic curve respectively.

[5] `https://docs.rs/hash2curve/`

3. Compute a 1-out-of-2 ZKP based on the Cramer-Damgård-Schoenmakers method [10] to prove that v_{ij} is either 0 or 1 (i.e., the encrypted vote for each candidate is well-formed).

These per-candidate votes are then collated into an overall ballot i, with an additional ZKP to prove that exactly one out of the k candidates has been chosen, i.e., $\sum_j v_{ij} = 1$. We use Chaum-Pedersen's equality ZKP [9] to realize this. All the ZKPs implemented in our system are made non-interactive by applying the Fiat-Shamir heuristics [21]. Unique indices of the election ID, the question number, the ballot number and the candidate number (where applicable) are included into the hash function to bind each ZKP with its unique context.

Casting a ballot comprises two steps (see Figure 1 for an illustration). First, the voter chooses a candidate from the list. Second, the voter has the option to either cancel or confirm the ballot. In case of cancelling the ballot, the ballot is cancelled and the voter is redirected to Step 1 to vote again. In case of confirming the ballot, the ballot is cast and the voting session is finished.

A receipt for the ballot i consists of two parts, corresponding to the above two steps respectively. The first part is generated as soon as the voter chooses a candidate. It contains Z_{ij} and R_{ij} for $j = 0, 1, \ldots, k-1$ and the ZKPs. The receipt is displayed on the voter's voting page, which they can choose to print out. The same content is also posted on the public bulletin board together with a digital signature to prove data authenticity. The voter is encouraged to check that the receipt is the same as the one published on the bulletin board. To make this check easier, we adopt a truncated hash approach used in the Gateshead trial [12] by computing a truncated hash of 50 characters (in base-32 encoding), arranged in two rows and groups of 5 characters. This truncated hash is called the "confirmation code" on the receipt. It is sufficient for the voter to check the "confirmation code" on the receipt matches that on the public bulletin board. The full cryptographic data (including R_{ij}, Z_{ij}, ZKPs and digital signatures) are available on the bulletin board for public verification.

The second part of the receipt depends on whether the voter chooses to cancel or confirm their ballot. In case of cancelling the ballot, the chosen candidate (v_{ij}, $j = 0, 1, \ldots, k-1$) is revealed on the receipt. The same content, together with the cryptographic random secrets r_{ij}, is also published on the bulletin board. With r_{ij}, anyone is able to decrypt Z_{ij} and R_{ij} and check whether the decrypted results match the v_{ij} values. This assures "cast-as-intended". A cancelled ballot does not add to the tally. After a voter cancels a ballot, they can vote again.

In the case of confirming the ballot, the receipt shows no additional information other than that the ballot has been "confirmed", and that the ballot is the encryption of either a 'Yes' vote or a 'No' vote. A voter is encouraged to check that the same receipt is published on the bulletin board. This provides assurance of "recorded-as-cast". The server keeps a running tally t_j for each candidate j, and an aggregated secret s_j. The t_j and s_j values are initially zero and are updated during the voting phase for each confirmed ballot, i.e., $t_j \leftarrow t_j + v_{ij}$, $s_j \leftarrow s_j + r_{ij}$ for $j = 0, 1, \ldots, k$. The individual values of v_{ij} and r_{ij} ($j = 0, 1, \ldots, k$) are securely erased once a confirmed ballot is cast. At

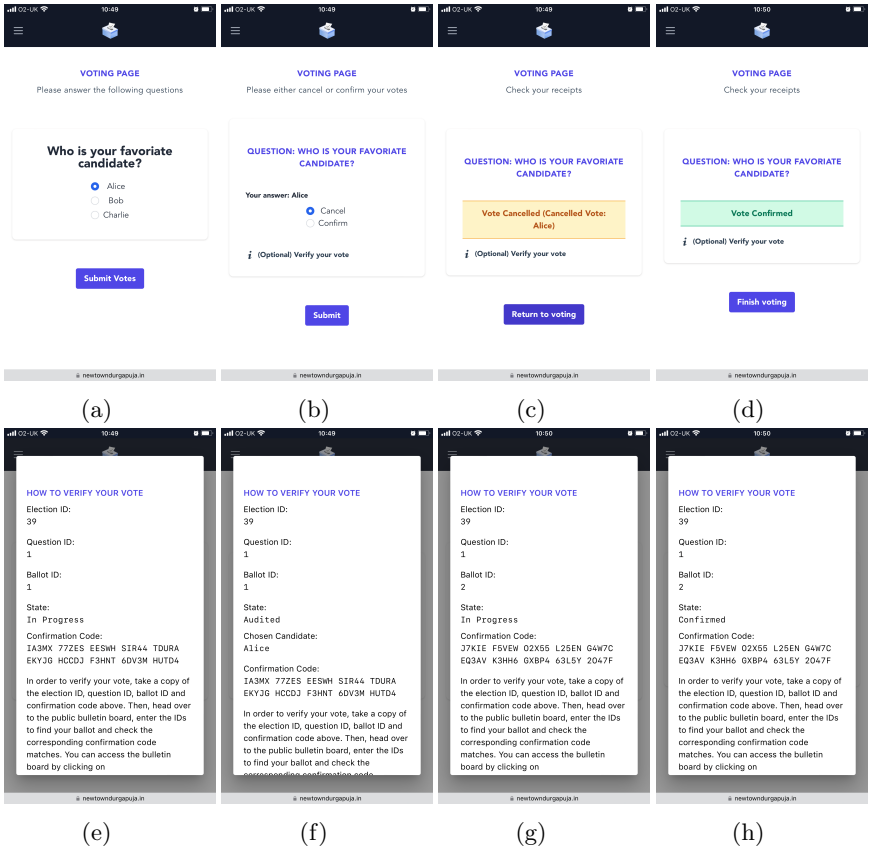

Fig. 1: Voting interfaces in an example election. (a) Step 1: choose a candidate. (b) Step 2: cancel or confirm. (c) The vote is cancelled. (d) The vote is confirmed. (e) Part-I receipt for ballot 1. (f) Part-II receipt for ballot 1 (audited). (g) Part-I receipt for ballot 2. (h) Part-II receipt for ballot 2 (confirmed)

this point, not even the system knows which candidate a particular ballot is for, and the receipts reveal no information that would enable a coercer to deduce the vote. We note that a coercer could learn the vote by being present with the voter and observing how they cast the vote; however, this is a threat that generally applies to all remote voting applications (including postal voting).

4.3 Result Phase

When the voting end time has passed, the system stops accepting new votes; any existing ballots that are neither confirmed nor cancelled are automatically cancelled. The system publishes the tally t_j and the aggregated secret s_j for all k candidates. The tally t_j for each candidate $j \in [0, k-1]$ can be verified as follows. We know that $t_j = \sum_i v_{ij}$ and $s_j = \sum_i r_{ij}$ should hold over the secret

(and now erased) v_{ij} and r_{ij} values of all confirmed ballots $\mathbb{C}$. Therefore, the following properties hold over the public values of confirmed ballots:

$$\sum_{i \in \mathbb{C}} Z_{ij} \;=\; G_1 \cdot (s_j + t_j) \quad \text{and} \quad \sum_{i \in \mathbb{C}} R_{ij} \;=\; G_2 \cdot s_j \tag{1}$$

These two properties are defined on only public values and therefore can be checked by anyone. They are equivalent to checking the correctness properties on the secret values and thus the correctness of the tally. In our implementation, once an election is finished, we publish the full audit data in JSON format on the bulletin board so anyone can download and check. To facilitate public verification, we provide an open-source tool to allow anyone to verify that all ballots are "tallied-as-recorded". One is also free to write their own verification tool since the verification algorithm is entirely public without any secret keys.

5 Durga Puja Online Voting Trial

Ethics approval. The Durga Puja trial represents a joint collaboration between the academic research team and NKDA, which is the local authority governing the New Town city. The system was developed according to the NKDA requirements in Section 3.1. After the system was developed, it was internally tested by NKDA in several iterations before they officially approved its use for the Durga Puja trial. Meanwhile, the trial was also approved by the University of Warwick's research ethics committee. As part of the ethics approval process, it was agreed that we would not provide any awards to the winning candidates (apart from an honourable mention on the NKDA website), or make payments to participants of this trial. This was to avoid potential bias in favor of the system.

The trial was agreed to run during the 2022 Durga Puja festival: starting on the 1st of October 2022, 12:00 AM and ending on the 5th of October 11:59 PM (Indian standard time). From 23 September to 30 September, NKDA issued a public call through their official website for the nomination of candidates for the 'block' and 'housing' categories respectively. In the end, 14 candidates were nominated for the 'block' category, and 18 candidates were nominated for the 'housing' category. For each category, the voting question is the same: "Best Puja under citizen choice". After the candidates, the voting question and the start/end times for the election were specified, the election setup was completed and the subsequent election management was automated.

The online voting system started accepting votes at midnight (12:00 am) on 1st October 2022, when the Durga Puja festival in New Town officially began. Over the next 5 days, 543 voters participated with 506 cast votes recorded (see Figure 2). After voting, each voter was provided with a web link to an anonymous survey using Google Form. In total, 95 voters filled in the survey.

The survey contains three sections. The first section includes questions about basic demographics, such as gender, age, education, and computer usage background. The second section includes 10 standard System Usability Scale (SUS)

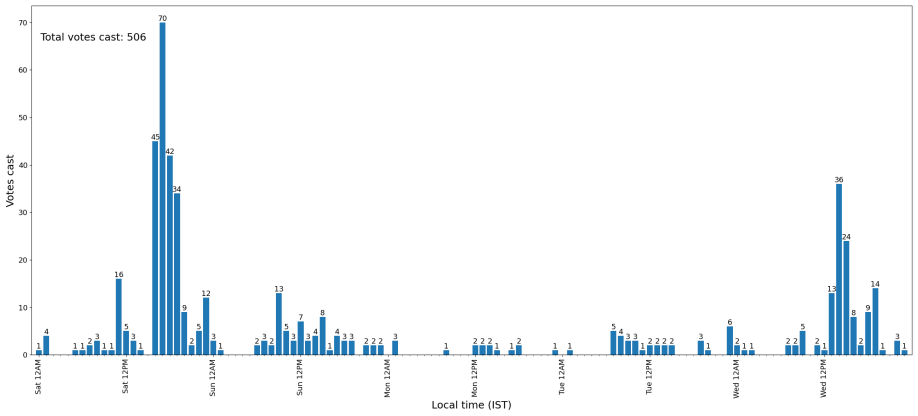

Fig. 2: Voter turnout during the 2022 Durga Puja trial

questions in order to evaluate the usability of the system. The third section includes a question asking the voter for their preference between postal voting and E2E verifiable online voting.

6 Trial Results

In this section, we analyze the voting statistics and the collected surveys for usability evaluation, as well as possible correlations with demographics.

6.1 Demographics

Among the 95 participants in the anonymous survey, 70 of them (73.7%) are male, and 25 (26.3%) are female. In terms of age, 24 are aged above 60 (25.3%); 24 between 50-59 (25.3%); 22 between 40-49 (23.2%); 16 between 30-39 (16.8%); seven between 20-29 (7.4%); one under 20 (1.1%); with one preferring not to say (1.1%). In terms of educational background, 50 hold a postgraduate degree (45.3%); 18 hold an undergraduate degree (18.9%); 14 are at college (14.7%); three people are at secondary school (3.2%); one person has not attended anything above or equivalent to secondary school (1.1%); nine preferred not to say (9.5%). We observe that the percentage of postgraduate degrees, as self-reported by participants, is higher than expected (we had a similar observation in an earlier study [12]). This is likely because some participants are not familiar with the meaning of a "postgraduate degree" and over-claim their educational background. Nonetheless, this should have little effect on our main analysis results as we do not find any significant correlation between usability scores and educational background, both with and without "postgraduate degree" (see §6.2). On the experience of computer usage, 43 report that they use computers "extensively" (45.3%); 30 use them "often" (31.6%); 13 "sometimes" use computers (13.7%); nine only "occasionally" use them (9.5%).

6.2 System Usability Scale scores and correlations

We adopt the standard SUS framework developed by John Brooke [6] for evaluating the usability of our system. The SUS framework consists of 10 statements. Respondents must indicate their agreement with these statements on a five-point Likert scale (1 = "strongly disagree", 2 = "disagree", 3 = "neutral", 4 = "agree", 5 = "strongly agree"). This returns a score that can be compared against other systems. We use the original wording for the SUS questions, except that we change one statement "I think that I would like to use this system frequently" to "I think that I would like to use this system in the future". This follows the same change made to this SUS question in the Gateshead trial [12].

Using the SUS computation method, our mean SUS score is 79.55 with a standard deviation of 17.13. This is between "good" (SUS = 73) and "excellent (SUS = 85) based on commonly used criteria [4]. We observed that the main inconvenience for some voters was solving a CAPTCHA challenge while going through the OTP authentication. In ordinary cases, a user simply needs to click a tick-box to pass the reCAPTCHA v2, but Google sometimes prompts the voter to manually solve a CAPTCHA challenge (e.g., identifying traffic lights). This is decided by the Google server based on several factors, e.g., if the user has previously authenticated to a Google account in the browser. The use of reCAPTCHA v3 would reduce the need for this user interaction, but it requires a dedicated administrator to monitor the traffic and adjust the threshold accordingly.

Based on the Spearman correlation method, we find the SUS score has little correlation with age (Spearman correlation coefficient $\rho = -0.065$, and two-tailed $p = 0.533$) and education ($\rho = 0.134, p = 0.219$)[6]. However, the SUS score is found weakly correlated with computer experience, with higher scores for users with more computer experience ($\rho = 0.319$, $p = 0.002$). This result is broadly consistent with a previous usability study of a DRE-ip implementation for a polling station voting trial at Gateshead, UK [12], in which the SUS score was found uncorrelated with the age, gender, or educational background but positively correlated with the voter's computer usage experience.

The final question in the anonymous survey asks the voter: "If you have to vote remotely, which system do you prefer?" Voters are asked to choose a preference between two remote voting methods: postal voting and E2E verifiable online voting (as used in the trial). The choices are given in a 5-point Likert scale: 1 ("strongly prefer postal voting"), 2 ("prefer postal voting"), 3 ("neutral"), 4 ("prefer online voting") and 5 ("strongly prefer online voting"). Figure 3 summarizes the preferences from the voter feedback. The vast majority (89 or 93.7%) of participants responded that they prefer or strongly prefer online voting. This preference is found weakly correlated with SUS with those rating higher SUS scores more likely to prefer online voting ($\rho = 0.38$, $p < 0.001$).

We emphasize that the strong preference for online voting over postal voting in Figure 3 is expressed within the context of an informal non-statutory election.

[6] If we remove "postgraduate degree", there remains no significant correlation between the SUS and education with $\rho = 0.276$ and $p = 0.103$.

The result may not necessarily generalize to statutory elections, where the election rules are different, and so are the voters' expectations. This will need to be studied further in the future. We note that voting as a basic democratic method is not limited to government elections; a person's democratic right can be exercised in many mundane voting scenarios, such as classroom voting, boardroom voting, and festival voting. In these scenarios, in-person voting using paper ballots is not always possible. When voting is conducted remotely, postal voting can prove costly especially when it needs to be done frequently. In that case, E2E verifiable online voting can become a possible alternative to postal voting.

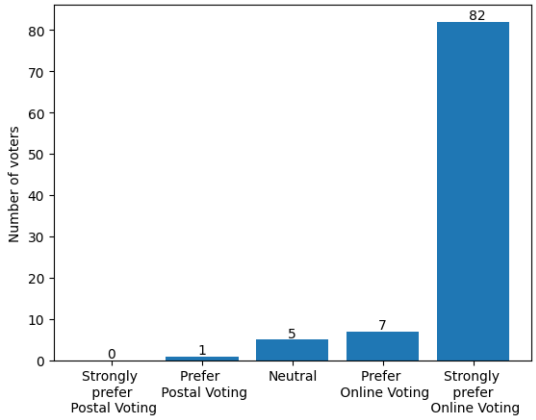

Fig. 3: Voter preference between postal voting and E2E online voting

6.3 Other statistics

Auditing rate. Our system recorded 506 cast votes during the 5-day trial. Among these votes, 12 were cancelled (or audited) votes while the remaining 494 were confirmed votes. This gives an auditing rate of 2.4%, which is similar to expectations from the literature when the voter is left to themselves to choose if they wish to confirm or cancel a ballot [12]. In practice, the auditing rate can be increased by employing dedicated auditors whose task is to cancel votes and check cast-as-intended at any random time during the voting phase [21].

Completion rate. During the trial, 543 people (with distinct Indian mobile numbers) passed the OTP authentication and chose one of the two groups. However, only 494 cast confirmed ballots. This represents a completion rate of $494/543 = 91\%$. We were not able to follow up with those who did not finish voting, but we analyze that this is most likely because after choosing one of the two groups, a voter does not find their favored candidates on the list, hence they quit voting by closing the browser.

Voting time. We record the voting time from the moment a voter enters a mobile number in the initial authentication step to the completion of voting by casting a confirmed ballot. If a voter passes OTP authentication but does not cast any confirmed vote, they are allowed to log in again later and continue voting. In that case, the voting time starts from the latest login. The average voting time is found to be 72 s (STD = 170 s) with the median duration being 34 s. The minimum voting time is 10 s, while the maximum time is 1736 s.

6.4 Performance Measurements

We ran the core DRE-ip code, written in Rust, on a computer with a 3.8 GHz CPU and 32 GB memory (featuring an AMD Ryzen 7 5800X processor). An election of 16 candidates and 10,000 ballots was created to evaluate the performance. Table 2 summarizes the time measurements averaged over 10,000 ballots. Overall, it takes a total of 13.04 ms to create a ballot (containing 16 candidates) during voting, and a total of 13.72 ms to verify every ballot during the tally verification. Verifying the tally of 10,000 ballots takes 137.2 s in total (which can be reduced by running the verification operations in parallel).

	Operation	**Time**	**No of multiplications in EC**
Create a ballot	1). Encryption	3.03 ms	32
	2). Candidate ZKP	9.80 ms	96
	3). Ballot ZKP	0.21 ms	2
Verify tally	1). Candidate ZKP	12.88 ms	128
	2). Ballot ZKP	0.43 ms	4
	3). Signature ZKP	0.40 ms	4
	4). Tally	0.01 ms	2

Table 2: The time measurements per ballot, averaged over 10,000 ballots

In the last column of Table 2, we list the number of scalar multiplications performed on the elliptic curve for each operation. During the creation of a ballot, 1) "Encryption" involves $16 \times 2 = 32$ multiplications to compute R_{ij} and Z_{ij} for 16 candidates ($j = 0, 1, \ldots, 15$); 2) "Candidate ZKP" involves $16 \times 6 = 96$ multiplications to generate 16 1-out-of-2 ZKPs; 3) "Ballot ZKP" involves 2 multiplications to generate one equality ZKP (which ensures that only one candidate out of 16 can be chosen). During the verification of tally, 1) "Candidate ZKP" involves $16 \times 8 = 128$ multiplications to verify the 16 1-out-of-2 ZKPs[7]; 2)

[7] It is possible to use only 6 multiplications (instead of 8) to verify one 1-out-of-2 ZKP by adapting a Simultaneous Multiple Exponentiation (SME) [21] method in the EC setting. However, we did not implement this optimization in our code.

"Ballot ZKP" involves 4 multiplications to verify the equality ZKP; 3) "Signature ZKP" involves 4 multiplications to verify the 2 ECDSA digital signatures per ballot (2 signatures correspond to the two stages during voting); 4) "Tally" involves 2 multiplications to verify the tally as shown in Equation 1.

In our experiment, we measured that each elliptic curve (EC) multiplication took 0.096 ms on average on our test machine. This allows us to theoretically estimate the latency based on the assumption that the EC multiplication is the dominant cost in the computation. As an example, for the "Encryption" operation, 32 multiplications correspond to a theoretical estimate of $32 \times 0.096 = 3.07$ ms, which matches closely the 3.03 ms empirical result shown in Table 2. All the other experimental measurements in Table 2 match our theoretical estimates based on the number of multiplications, except the "Tally" operation. This exception occurs because, for a large number of ballots, the cost of Equation 1 is dominated not by the multiplication, but by the addition (i.e., computing $\sum Z_{ij}$ and $\sum R_{ij}$). Hence, we need a different way to estimate the cost. Using the repeated squaring method, one multiplication approximately requires $256 + 128 = 384$ additions on the NIST P-256 curve. For summing up Z_{ij} and R_{ij} for 16 candidates, we need $2 \times 16 \times 10,000 = 320,000$ additions, which is equivalent to $320,000/384 = 833$ multiplications. For each candidate, we require 2 multiplications for the tally verification in Equation 1. So all together that is equivalent to $833 + 2 \times 16 = 865$ multiplications. Given that each multiplication costs 0.096 ms, the total cost is estimated to be $0.096 \times 865 = 83.04$ ms for $10,000$ ballots, i.e., 0.008 ms per ballot. This approximately matches the 0.01 ms experimental measurement per ballot shown in Table 2.

7 Related Work

Many case studies have been examined in the literature regarding the deployment and usability analysis of verifiable e-voting systems. These studies may largely be classified into those for polling station voting [7,8,12], where a voter must cast their vote in-person, or for remote voting [17, 24], where a voter is permitted to cast their vote from any location, for instance through the Internet. Example case studies for remote voting include the analyses of Swiss Post's voting system by Marky et al. [17] and Volkamer et al. [24]. Additionally, Acemyan et al. provide baseline data for the usability of the Helios voting system [1]. In this section, we will focus on comparing with Helios, since both Helios and our system are end-to-end verifiable, whilst the Swiss Post voting system is currently not [17].

Helios [2] is a web-based voting system, which initially utilized the Sako-Kilian mix-net [20] in Version 1.0, but later switched to use homomorphic encryption in Version 2.0. Helios 2.0 implements the Cramer-Gennario-Schonmakers (CGS) voting protocol [10], which aggregates encrypted ballots based on homomorphism and then employs a set of TAs to perform threshold decryption. It additionally incorporates Benaloh's voter-initiated auditing method [5] to ensure "cast as intended". In 2009, Helios 2.0 was used to elect the president of

Université catholique de Louvain (UCL) [2]. Since Helios 1.0 has been replaced by Helios 2.0, by Helios, we refer to Helios 2.0 thereafter.

Acemyan et al. assessed the usability of Helios through a within-subjects study of 37 participants [1]. A mock election was created through the Helios website in 2014. According to Acemyan et al., although 95% of voters perceived that they had cast the vote through Helios, only 60% actually had. This discrepancy was mainly because of the way that Helios implemented voter auditing according to Beneloh's challenge [5]. Anyone can choose candidates from a list and perform auditing by cancelling votes *without being authenticated*. If a voter opts to confirm the vote, they may think voting has finished, but they still need to pass authentication to actually cast the vote. This issue could be addressed by adding an explicit warning on the voting page to prompt the voter to log in and finish voting. In our system, voters are authenticated at the start of a voting session, hence avoiding any potential confusion.

Our E2E voting system differs from Helios in several aspects. First, instead of implementing a TA-based CGS protocol [10], we adopt a TA-free DRE-ip protocol [21]. As reported by the Helios authors in the UCL trial [2], finding and managing the TAs proved to be a "particularly difficult issue". As a mitigating measure, the Helios website now prompts the election creator to choose the Helios server as a single TA and discourages defining more TAs due to the additional complexity involved[8]. In a single TA-based Helios election, a compromise of the server (i.e., the TA's private key) breaks the secrecy of all votes. In our system, there is no TA (or TA's private key) and a compromise of the server only reveals the partial tally at the time of compromise, which minimizes information leakage [21]. For both Helios and DRE-ip, voter privacy can be enhanced through anonymity: only authenticated voters are allowed to vote but their real identities are unknown to the system. Anonymous voting can be realized in a polling station through physical means, e.g., by assigning random passcode slips to voters [12]. For remote voting, a similar procedure may be used to distribute authentication credentials to enable legitimate voters to vote anonymously.

Second, while Helios allows a voter to audit votes without authentication as specified in Benaloh's challenge [5], we require voters to be authenticated first according to the DRE-ip specification. Based on the trial, we find that this helps provide a more natural and intuitive voting process for ordinary voters, as well as a simpler voter-initiated auditing procedure as explained below.

Third, Helios requires a user to verify an audited ballot by copying and pasting the cryptographic data from a voting page to the text field of a verification page running the JavaScript code, but this can prove difficult for ordinary voters [18]. The verification of an audited ballot in our system is simpler as the voter only needs to check if the receipt of a cancelled ballot matches that published

[8] When the election creator attempts to specify more than one TA, the Helios website displays the following popup alert (as of 6/11/2023): "Adding your own trustee requires a good bit more work to tally the election. You will need to have trustees generate keypairs and safeguard their secret key. If you are not sure what that means, we strongly recommend clicking Cancel and letting Helios tally the election for you."

on the public bulletin board based on DRE-ip [21]. However, doing the same in Helios appears difficult because a user is *unauthenticated* when auditing a ballot; allowing an unauthenticated user to post data on the public bulletin board of an election can expose the system to a Denial of Service attack.

8 Conclusion

Empirical studies play an important role in e-voting research to validate not only the assumptions of a theoretical voting protocol but also the feasibility of the system from a voter's perspective. Despite extensive research on E2E verifiable e-voting, such studies remain limited: few E2E voting systems have been implemented and used in practice. In this paper, we present an E2E online voting system based on DRE-ip and conduct a trial among the New Town residents in India as part of the 2022 Durga Puja festival celebration. Based on the voter feedback, participants generally found our system easy to use. Our trial experience also shows that the removal of tallying authorities significantly simplifies not only the implementation but also the election management, making E2E e-voting more practical than before.

Acknowledgements

We would like to thank the New Town Kolkata Development Authority (NKDA) for their generous support. Special thanks go to Debashis Sen, Suvayu Ray and Pritam Thakur for making the Durga Puja trial possible. This work was supported by the Royal Society International Collaboration Award (ICA\R1\180226).

References

1. C. Acemyan et al. Usability of Voter Verifiable, End-to-end Voting Systems: Baseline Data for Helios, Prêt à Voter, and Scantegrity II. In *EVT/WOTE*, 2014.
2. B. Adida et al. Electing a university president using open-audit voting: Analysis of real-world use of Helios. *EVT/WOTE*, 2009.
3. S. Bag, M.A. Azad, and F. Hao. E2E Verifiable Borda Count Voting System without Tallying Authorities. In *ARES*, 2019.
4. A. Bangor et al. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008.
5. J. Benaloh. Simple Verifiable Elections. *EVT*, 2006.
6. J. Brooke. SUS: A quick and dirty usablility scale. *Usability evaluation in industry*, 189(3), 1996.
7. C. Burton et al. Using Prêt à voter in Victorian state elections. In *EVT/WOTE*, 2012.
8. R. Carback et al. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In *USENIX Security*, 2010.
9. D. Chaum and T. Pedersen. Wallet databases with observers. In *CRYPTO*, 1992.

10. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *EUROCRYPT*, 1997.
11. F. Hao et al. Every vote counts: Ensuring integrity in large-scale electronic voting. In *EVT/WOTE*, 2014.
12. F. Hao et al. End-to-End Verifiable E-Voting Trial for Polling Station Voting. *IEEE Security & Privacy*, 18(6):6–13, 2020.
13. F. Hao and P. Ryan. *Real-World Electronic Voting*. CRC, 2016.
14. L. Harrison, S. Bag, and F. Hao. Camel: E2E Verifiable Instant Runoff Voting without Tallying Authorities,. In *ASIACCS*, 2024.
15. L. Harrison, S. Bag, H. Luo, and F. Hao. VERICONDOR: End-to-End Verifiable Condorcet Voting without Tallying Authorities. In *ASIACCS*, 2022.
16. India News. EVM Row: RTI Reveals Case Of Voting Machine's Malfunction In Maharashtra Zilla Parishad Election. `https://tinyurl.com/bdd3td3y`, July 2017.
17. K. Marky et al. Improving the usability and UX of the Swiss internet voting interface. In *CHI*, 2020.
18. S. Neumann et al. Helios verification: To alleviate, or to nominate: Is that the question, or shall we have both? In *EGOVIS*, 2014.
19. S. Rai. Furore in Meerut after voter presses BSP on EVM, vote goes to BJP. `https://tinyurl.com/46mawdue`, November 2017.
20. K Sako and J. Kilian. Receipt-free mix-type voting scheme. In *CRYPTO*, 1995.
21. S. Shahandashti and F. Hao. DRE-ip: A verifiable e-voting scheme without tallying authorities. In *ESORCIS*, 2016.
22. D. Springall et al. Security analysis of the Estonian Internet voting system. In *CCS*, 2014.
23. J. Townsley et al. Who votes by post? understanding the drivers of postal voting in the 2019 british general election. *Parliamentary Affairs*, 2021.
24. M. Volkamer et al. Increasing security without decreasing usability: A comparison of various verifiable voting systems. In *SOUPS*, 2022.
25. S. Wolchok et al. Security analysis of India's electronic voting machines. In *CCS*, 2010.