An Analysis of Voter Fraud and Proposed Methodology for Recording Numerical Anomalies in Polling Data

Harrison T. Granger

A Senior Thesis submitted in partial fulfillment
of the requirements for graduation
in the Honors Program
Liberty University
Spring 2021

Abstract

Debate surrounding the security and accessibility of voting procedures in the United States has caused the matter of electoral integrity to become a point of contention. Confusion surrounding the Federal government's response to allegations of fraud has decreased the population’s trust in the government and increased voter apathy. The context for an empirical methodology is established with a history of e-voting, voter security, and voter-fraud sensationalism. This research methodology is then described and implemented with a software application that analyzes the proximity of voters to polling locations to study the way people vote and how fraud may be committed at polling stations. The development cycle, design, and use cases of this software are described, and a variety of improvements are suggested as motivation for future application.

An Analysis of Voter Fraud and Proposed Methodology for Recording Numerical Anomalies in Polling Data

The balance between a fair and free election and a secure election is a point of dispute for many politicians and their constituents. Proponents of voter ID laws and polling security often find themselves at odds with voting-rights activists – a strange conflict provided that the voter apathy and government distrust that both groups experience stem from the same cause. Conflicts at the state and federal levels have caused a lack of empirical sensibility during elections, leading to a variety of consequences in American politics including a lack of voter confidence in the voting system, increased demand for updated voting technology, and sensationalized responses from American media outlets surrounding the topic of voter fraud. Provided this context, a research methodology is proposed that analyzes numerical anomalies in polling data by matching multiple votes from the same electronic identities of citizens and mapping these occurrences on a geographical map for empirical analysis.

Historical Context and Analysis

Distrust in election results among the population stems from a lack of transparency in the voting process, resulting in apathetic voters at the polls. Though there have been suggestions for technological enhancements to the voting process, media sensationalism has mobilized the contention between advocates of voting security and voting accessibility, convoluting empirical discourse surrounding the topic. This context establishes the foundation for a research methodology proposal that will introduce transparency to the voting process by recording and presenting numerical anomalies in polling data.

Voting history

Voter apathy has existed in populations of American democracy throughout the history of the United States (Meehan, 2004). Among the voting population, youth and minority citizens have the lowest rates of participation, attributed to an ignorance of the voting system and a distrust of the government (Alvarez et al., 2018). In the general population, government distrust has grown over the last century due to increasing globalization and a lack of electoral transparency. Those prone to voter apathy fail to vote because the vote is perceived to be worthless (Hardin, 2013). Exit-poll research conducted at an Ohio polling station in 2012 during local elections showed that 18% of voters were "not too confident" or "not at all confident" that their vote would be counted fairly (Claassen et al., 2013). The presumption of unfair and fraudulent elections has endured in the United States. Without an understanding of how votes are processed, many individuals refuse to trust the election process.

Distrust in the electoral system is a consequence of the conflict between the attempts to guarantee voting accessibility and the desire to maintain polling security. During the Gilded Age of American history, a period lasting from the 1870s to about 1900, bribery, fraud, and corruption were prevalent at ballot boxes during elections (Argersinger, 1985). Though voting measures in the 20th century such as private balloting at polling locations and required voter registration reduced overt polling fraud, voters' trust in the election process was not wholly restored (Argersinger, 1985). During the 1950s and '60s, political scientists noticed that certain eligible populations of voters failed to turn out at the polls. To remedy this problem, legislators proposed lesser voting requirements to incentivize voter turnout (Abraham, 1952). Lesser requirements soon became proactive measures, and soon states began implementing early voting and allowed more exemptions for more individuals to vote through the postal service in a bid to

increase participation (Giammo, 2010). As the accessibility of the polls increased, existing apathetic voters who distrusted the electorate system noted the apparent lack of security measures in the name of accessibility (Claassen et al., 2013). In this sense, voting accessibility and voting security share an inverse relationship – reducing a registration measure to encourage voter participation could make defrauding a ballot easier. Increasing the measures of security could make casting a ballot more difficult. In this way, while many constituents ask for transparency by regulation, the very problem of mobilizing youth and minority voters has led to proposals of reducing voting requirements, causing increased tension for existing apathetic voters who distrust the system (Claassen et al., 2013). The tension that exists between the freedom to vote and the facilitation of a secure election has since plagued American confidence at the polls.

The debate surrounding polling transparency has caused a variety of parties to advocate for the implementation of technology in the voting process. The hope is that voting with the assistance of technology will enhance transparency while also increasing accessibility, in turn reducing the government distrust that causes voter apathy at the polls. Officials have since considered adding novel ways of e-voting as technology has matured (Alvarez & Levin, 2018). Though there are many exciting implementations, each has its vulnerabilities that may be encountered.

E-Voting

E-Voting refers to voting that harnesses the power of technology to enhance the polling process. Various e-voting implementations have been attempted all over the world. In their study "Biometric technology for voter identification: The experience in Ghana", Effah and Debrah write that government officials attempted to implement a biometric ID that would only allow a

citizen to vote after undergoing a vetting process (2018). The process required the citizen to provide proof of citizenship, take a biometric scan of all ten fingers, and provide a profile picture to obtain an ID card with a barcode and ID number (Effah & Debrah, 2018). A Russian state instituted optical scans to automate security aspects of the polling process (Bader, 2014). In Venezuela, citizens use Smartmatic direct-recording electronic (DRE) machines like those used in some regions of the United States (Levin, 2009). The Baltic republic of Estonia was the first country to introduce remote voting in 2005, and many other countries have since attempted to follow suit (Gibson et al., 2016). Proponents of e-voting suggest that the United States can similarly implement technology into the voting process to increase the transparency and accessibility of elections.

There are several ways to implement an e-voting system. DRE machines, internet balloting, and blockchain internet balloting have all been suggested as implementations of e-voting to enhance the integrity and transparency of the election process while also providing greater access to the polls. Many countries have already adopted DRE machines. DRE machines have a touchscreen and record a voter's preference immediately, storing the information in a database for later processing (Krimmer, 2012). Though DRE implementation is the most logical advancement following scanned paper ballots, DRE's render the election process hardly more accessible than the paper ballot because voters must still go to polling stations. Internet balloting enhances the accessibility of DRE's by taking all voting to the internet via a government website (Gibson et al., 2016). This approach has been trialed in many countries at various events abroad and in the United States – for example, the Iowa Democratic Caucus trialed online voting via a cellphone app (Gibson et al., 2016). Gibson and fellow researchers write that proponents of this strategy suggest that online voting provides ubiquitous access to a world that is now connected

by the internet, while opponents of online voting point out that vulnerabilities in the system could open the process to a dangerous amount of fraud and speculation. The cellphone application trialed in the Democratic Iowa caucus, for example, misreported results and caused a multi-day delay in election day results (Gibson, et al., 2016). With these vulnerabilities in mind, many advocates of the blockchain cryptography technology have suggested enhancing the security of online polling by incorporating data encryption with the blockchain. To do this, records would simply be stored in the blockchain as records called blocks. Each block in the blockchain has an encrypted hash pointing to the previous block, a timestamp, and a data sector. If an individual attempts to remove a data block from the blockchain or add data to it – adding fraudulent votes, for example – the chain of encryption is interrupted causing the chain to break, halting the process (Tarasov & Tewari, 2017). Proponents suggest that by incorporating the blockchain into the growing online election process, ballot integrity can be protected. In addition to these methods of e-voting, there are a variety of other implementations and suggestions for enhanced security measures including iris scans, fingerprint scans, facial geometry, and voice recognition (Cherney & Chaikina, 2019). Though e-voting takes many forms, there are just as many ways to defraud e-voting implementations.

There are two kinds of voting fraud: voter-initiated and voter-targeted fraud (Benson, J. 2009). Voter-initiated fraud is a variation of fraudulent activity that involves a perpetrator attempting to malign the voting process by manipulating polling numbers. Voter-targeted fraud is less noteworthy but just as damaging. Voter-targeted fraud occurs when an individual is targeted by false information to change their voting habits (Benson, J. 2009). While voter-targeted fraud is not uncommon, e-voting has the potential to populate voter-targeting campaigns through the far-reaching internet (Amer & El-Gendy, 2013) Though supporters of e-voting tout

accessibility as a major improvement over the traditional ballot, Amer and El-Gendy write that perpetrator of fraud are included as beneficiaries of this accessibility (2013). DRE machines were found to be easily exploited by a web-based payload. In a study performed by University College London, a PDF document was found to be suitable for carrying a malicious payload onto a DRE machine which was then compromised (Estehghari & Desmedt, 2010). If the voting process were to be taken online, client and server communication issues could cause havoc on voting day. Something as simple as a regional power outage could stifle thousands of votes by removing internet access for a day (Al-Ameen & Talab, 2012). E-voting may provide exciting new ways to vote, but it cannot fully end fraudulent activity at the polls. Even with its benefits, an alteration to the polling process is incredibly difficult because of the tension between the public, politicians, and activist groups. This tension is a result of media sensationalism and campaigns on social media.

Network-Media Sensationalism and Social Media

The network media in the United States has reduced the subject of election fraud to a sensationalized news story, polarizing public opinion surrounding the issue. This polarization creates an environment that is hostile to empirical discussions regarding voter fraud (Udani, Kimball, & Fogarty, 2018). This disregard for empirical sensibility is furthered by how the news media mobilizes this hostility. In the United States and Europe, young and minority voters are predisposed to trust networks that align with their political views and distrust networks that do not (Esser & Vreese, 2007). Because youth and minority voters will soon be the largest constituency, partisan news outlets use the topic of voter fraud as a mobilization technique to motivate their constituents to vote and cast doubt on an opponent's election validity (Fogarty, Kimball, & Kosnik, 2016). Media sensationalism of voter fraud increases voter’s distrust in the

election process and trivializes the occurrences of voter fraud. The catalyst for this desensitization in the general population is social media.

Social media acts as a loudspeaker for the message of voter fraud – decreasing confidence in voters and convoluting accounts of active fraud (Berlinski et al., 2021). Polarization among Americans regarding voter fraud is worsened by the way social media interacts with its users – social media applications are designed to provide tailored information to a user, creating an echo-chamber of information that further solidifies an individual's ideologies (Deb et al., 2019). Deb, Luceri, Badaway, and Ferrara write in their 2019 article "Perils and challenges of social media and election manipulation analysis: The 2018 us midterms" that social media is designed to propagate biased information. This bias affirms the preferences of apathetic voters, casting doubt on the electoral process and decreasing the confidence of American voters (Deb et al., 2019). There is a history of voters not trusting the electoral process in the United States. Provided the history of voter fraud and method in which information is contaminated online, a research methodology is proposed to enhance the transparency of voting numerical anomalies at polling locations. This methodology is proposed to adapt as technology in the voting process changes.

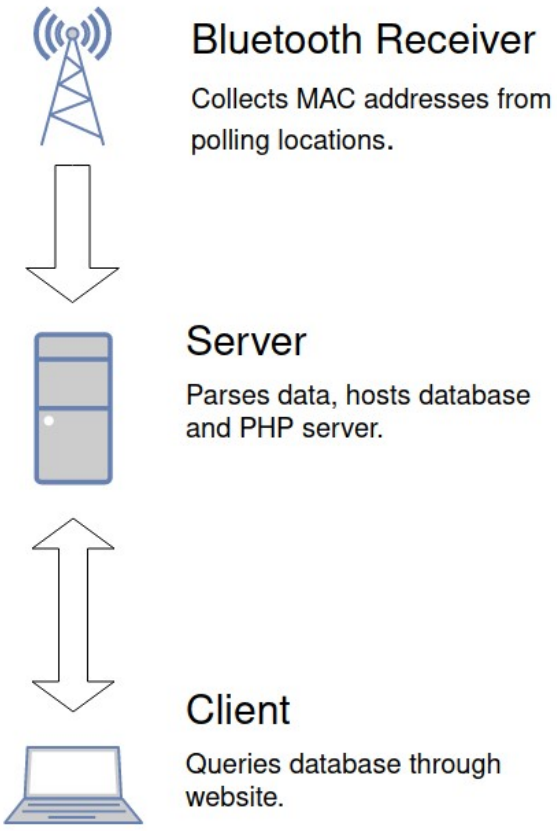
Research Methodology

An empirical methodology was developed to study the way ballot fraud occurs. The study begins by placing class 1 Bluetooth receivers at *n* polling locations, where *n* defines the number of locations participating in a polling event. Every electronic device that appears in the radius of contact for the transmitter is logged by recording the device's unique Media Access Control (MAC) address. At the end of the polling period, the MAC addresses are loaded onto a local terminal and parsed. Each MAC address record is assigned a coordinate value that

corresponds to the geographic location of the polling center. These records are then compared, and any record with different coordinate values but identical MAC addresses are saved. These values imply that a subject may have gone to more than one polling location. These records are then transformed and drawn onto a map, with a point for each polling location coordinate and a line drawn representing a connection between the two. All records of interest are then uploaded to a central database, with the database accessible to users via a PHP server. A webpage front-end interfaces with the database backend to allow the public to view polling location results to provide a transparent description of any numerical and polling anomalies that may occur. The following sections analyze the application of this methodology with a piece of software that was created, referred to henceforth as ‘The Application’. The following sections detail the development cycle of this software including system components, specifications, and motivation for future use. Figure 1 shows a basic diagram describing the functional relationship between different components of The Application.

Figure 1

Scanning Topology



Note. Basic network components detail the flow of data in the data collection process.

Application Development

The Application was developed on an Ubuntu-based Linux system for operation in the command line. There were a variety of languages and libraries used to support the development of The Application and many important specifications that were required for the project to be successful. These development components formed the foundation of the application development process.

Development Process

Five phases make up the development process: gathering requirements, analysis, design, coding and testing, and maintenance. Each category is critical in The Application's success for the next phase of the development process. If the Application failed to meet one of the following milestones at any point, the process was repeated to ensure consistency and accuracy during development.

Gathering requirements. The first step of the development process was to gather all requirements necessary for the project. This was done by surveying users. The requirements were then formatted into specifications that were formally recorded and analyzed for implementation.

Analysis. The next step of the development process was to analyze the research and development field and study the current state of similar applications. No other applications exist in a similar capacity to the proposed designs formulated by the requirements.

Design. The design process finalized the proposed ideas from the requirements and analysis stages of development. A final design was proposed for the application by studying the state of the field and implementing acquired user specifications that encompassed a user front-end, database back-end, and lightweight computational core.

Code and test. Once a final design for The Application had been decided, development began. Developers programmed and performed testing within Microsoft's Visual Studio Code application with the default debugger extension alongside native debugging utilities. C++ scripts were debugged with g++, and Python 3.9 scripts were debugged with debugpy. Validation testing was used to ensure The Application was producing accurate output.

Maintenance. Developers performed maintenance by monitoring system performance and validating test results for consistency. Large sets of output were further analyzed for accuracy after they had been added to the database.

The development process established the requirements and method of designing The Application. Furthermore, this process was used to designate the type of host system that would house The Application, the programming languages to be used, the types of libraries and packages to be imported, and the design specifications that would apply.

Application Components

Host Systems. The Application is hosted on a 64-bit architecture computer with 12 double-threaded CPU cores running Pop!_OS version 21.04.

Languages and Scripts. All referenced scripts were created as a part of this study and are hosted publicly at: https://github.com/HarrisonGranger/MerrySort. All scripts and libraries are called from a central shell script operated on the host machine. BASH was chosen as the scripting shell because of its extensibility and modern features, as well as its prevalence on most operating systems. From this BASH script, a variety of other tools and scripts are called. First, The Application calls AWK to format the incoming data stream of timestamps and MAC addresses into columns of records. These records are handed to a C++ executable binary that was compiled from a script called *sort.cpp.* C++ file streams are used in this script to read all processed data and identify records with different coordinate values and the same MAC address values. C++ 11 was chosen over other general-purpose languages such as Java or Python for its speed, lightweight nature, and prevalence on a variety of operating systems. Further, C++ does not limit file size of inputs. This is determined by the operating system and compiler, so independent of these constraints, The Application can take any number of records. Once the

records that contain duplicate coordinates and identical MAC addresses are retrieved, The Application calls a Python 3.9 script called *maps.py* to transform sorted data sets into plots for mapping. Then, *maps.py* draws the data points onto a Google map. The Application saves a copy of this map as an HTML file on the local computer. Python 3.9 was chosen for this task because of the extensive library of packages Python can access. Though often slower than other general-purpose programming languages such as Java or C++, Python can map data sets with far less overhead in this scenario.

Once all computations have taken place, The Application calls a Python 3.9 script called *upload.py* to upload the records to a MySQL database called msBE. Python 3.9 was used here for its ability to integrate with databases. A different language such as C# could have been used, but Python allowed for a large library of functions without necessitating the NET Core that C# requires. In tandem with the *upload.py* script, The Application uses *query.py* to interface with the database and pull information from previous scans. Finally, a group of PHP scripts comprises the front-end of The Application. These scripts include *index.php, connect.php, footer.php, header.php,* and *maps.php*. The first PHP script, *index.php,* is used as the starting point for front-end operations. This script is responsible for calling the other scripts into action. The second script, *connect.php,* is used to create a SQL object that interfaces with the msBE database. *footer.php* and *header.php* both include the code necessary to instantiate a webpage. This code includes HTML and calls the CSS scripts necessary to stylize the web page. The three CSS scripts called are *destyle.css, style.css, and variables.*css. Each is responsible for maintaining the visual format of the front-end. The final PHP script, *maps.php,* uses a file globe method to include all created HTML maps with corresponding timestamp online for download. This way, a user can download any dataset in the database with the corresponding map. Throughout the

development process, the languages and scripts relied heavily on libraries for a variety of functions.

Libraries and Packages. The Application uses a variety of libraries and packages to perform a range of advanced operations. The C++ script *sort.cpp* uses the Fstream library for file creation and detection of file size. The *maps.py* script uses the CSV, Gmplot, and OS packages. *maps.py* uses the CSV package to format incoming data and transform records from the CSV (comma separated values) convention to map plots. *maps.py* uses the Gmplot package to place these map plots onto a Google map. Finally, *maps.py* uses the OS package to save a copy of the HTML map onto the local computer. The *upload.py* script uses the PyMySQL and CSV packages. *upload.py* uses the PyMySQL to interface with the msBE SQL database and add data to the tables of the database. *upload.py* uses the CSV package to format data from the *sort.cpp* script for upload. The *query.py* script uses the same packages but for different applications. The PyMySQL package is used to query the database rather than add data. *query.py* uses the CSV reader to process input parameters from the user, allowing for the specification of querying options. These scripts and libraries are a necessary part of meeting the specifications of The Application during development.

Specifications. Specifications for The Application include functional requirements as well as operational requirements. First, The Application must be able to apply custom coordinates to each data file received at runtime. This would allow for a user to run ‘what if’ scenarios with data. Second, The Application must be able to sort data received from *n* number of files, where *n* denotes the upper bound of data to be received as input. This is a necessary component of the development process as polling events often comprise millions of records. It is critical that The Application can handle any amount of polling data. As a part of the efficiency

needed to take in this amount of data, the sorting algorithm used by The Application must have a computational complexity of $O(n^2)$. This is necessary for realistic computational time. Provided an excessive number of records, any algorithm with a computational complexity greater than $O(n^2)$ could approach exponential time as it calculates the results. This would be unacceptable.

Once the data points have been calculated and processed, the data coordinates must be graphed onto an HTML-based map. This is important because an HTML page can be viewed on any web browser. This is a critical method of keeping the results of computation available to the public. Once the data has been mapped, processed data must be stored in a database following SQL convention. By using a database to store the processed records, redundancy is increased. In tandem with this requirement, processed data must be retrieved by interfacing with SQL-styled queries. Furthermore, only authorized users must have access to the database itself. A user must not have direct access to the database. A user may access the database through the front-end of The Application, but not directly to the database. For this reason, a graphical interface must be provided to the user for the best user experience and accessibility of information. To ensure this data is accessible, data sets and graphs must be able to be downloaded with the specified graphical interface. These requirements and specifications ensure the accessibility, integrity, and confidentiality of the information processed with The Application. It was ensured that these core principles were maintained during the development process.

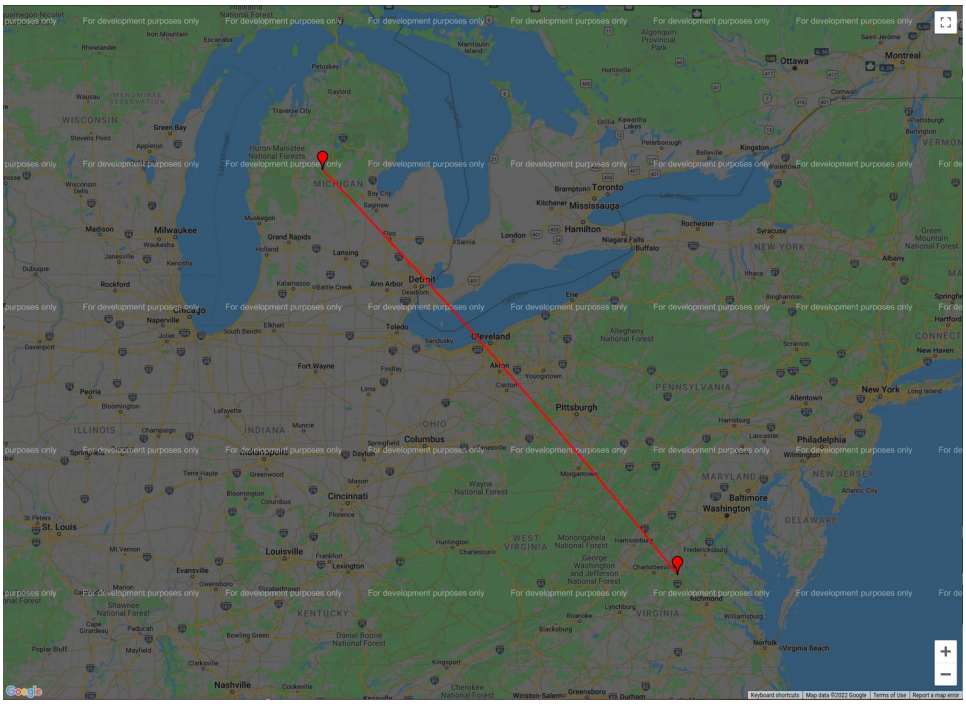
The development process for The Application began by framing important components and milestones of development including host system information, scripts, libraries, languages, and design specifications. These components form the foundation of the way that users interact with the software.

Application Operation

The Application first requests the path to a directory where the files reside. The script iterates over each file, requests a pair of coordinates from the user to apply to that file, and then trims the contents with AWK and appends the results to a universal data file. Once finished, the bash script calls a C++ script called *sort.cpp* that iterates through the temporary text file, placing any instances of identical MAC address with different coordinates into a formatted temporary file. Once finished, the bash script calls a Python script, *maps.py*, to create coordinates from the temporary file. The Python script uses the Gmplot package which employs the Google Maps JS API to place these coordinates onto a Google map. First, a Python script, *maps.py*, splices the text input and turns each row into a list. Then, the script calls the Gmplot graphing function to place the created list of coordinates onto a map, plotting a point for each identical MAC address with multiple coordinate sets and drawing a line between both points. Each marker is labeled with the corresponding coordinate set. This is repeated until the text file has been mapped. Once finished, the Python script saves the google map as an HTML file, as seen in Figure 2.

Figure 2

Geographical Mapping of Voting Inconsistency

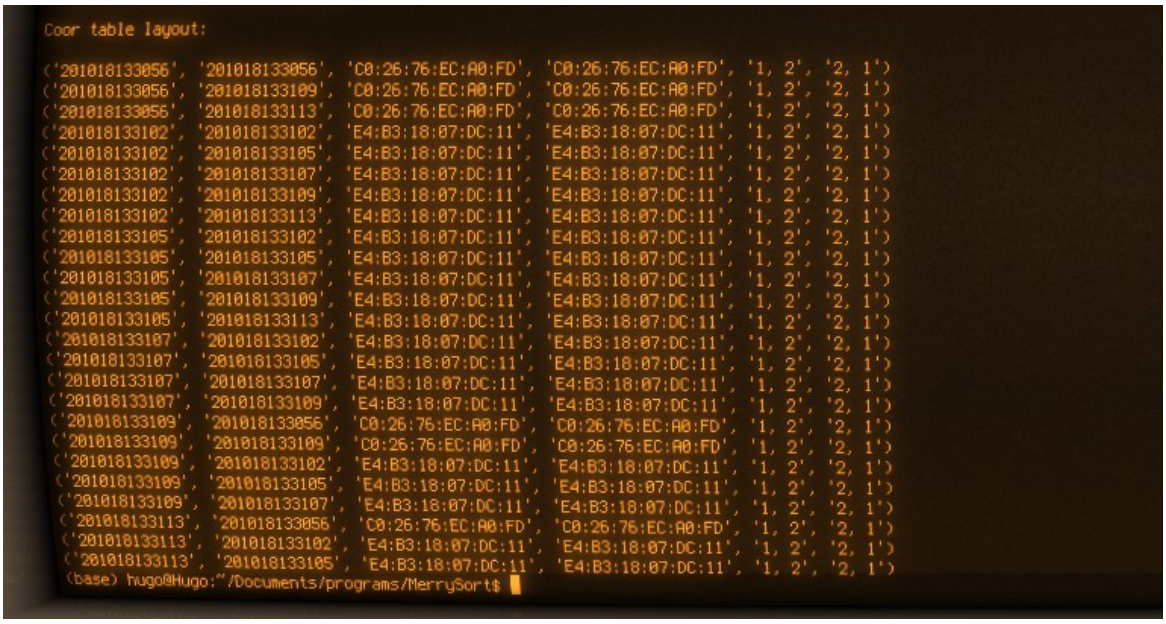


Note. A red line connects locations of possible fraud. Screenshot captured March 2020.

The script then calls *upload.py*, which uses the PyMySQL library to connect to the local MySQL database msBE. MySQL integration allows data that would otherwise reside in a temporary text file to be stored in a database. The Application stores all results of the coordinates temporary file into the database by iterating over and inserting each line into the database. Figure 3 shows the result of querying the database from the command line.

Figure 3

Retrieving Values from the SQL Database on the command line



Note. Screenshot capture March 2020. Bare columns are retrieved: timestamp of the duplicate votes, MAC address associated with the vote, and respective coordinates of each occurrence. Example coordinates of ‘1’ and ‘2’ are used here.

The addition of the database was added for a few reasons: first, it was necessary to store data besides plaintext files for security. Storing data in a text file increases the chance of accidental deletion and corruption, so storing the data in a database increases redundancy. Second, storing data within a MySQL database provides a more efficient way to manage data. Without the option to query the database, a user would be required to search through thousands of lines in a plaintext file for coordinates or a specific timestamp. With a database backend, the user can query for timestamps, coordinates, and other criteria. Once the data has been inserted, a user is able to query data in two ways: either use *query.py* and query the database directly from the command line or utilize the PHP-created website and download maps through a graphical

interface. When querying on the command line, current functionality allows a user to provide The Application with criteria such as a range of timestamps or coordinates. *query.py* then fetches all records that match the user's specification and returns them as standard system output. When accessing the database through the PHP front-end, fetching records is as easy as clicking a download link. Once the user has finished processing data, The Application deletes all temporary files and closes. All records remain in the database for future queries.

Application Configuration and Settings

The Application has a variety of settings and configuration features that can be enabled. These items include the querying options on the command line, configuration options when running The Application for specific output, and PHP configuration settings for the front-end. Configuration options when querying the database on the command line are shown in Table 1. The five commands include an option to query every record in the table, query a range of records based on record characteristics, query specific MAC addresses and coordinate values, and delete records from the table. Though relatively simple, these commands allow the user to see frames of data in files that would otherwise take an entire line to query individually. Commands are executed in the standard command line format with character references to command options.

Table 1

SQL Interfacing Commands

Command	SQL Interpretation
-A	SELECT *
-S	BETWEEN (starting value)
-E	BETWEEN (ending value)
-M	WHERE (MAC address values)
-C	WHERE (Coordinate values)

Note. All commands begin by calling the name of the executable script.

There are various options in the Bash script that allow for customization of data output to the console. The Application utilizes the input configuration tool getopts to record three primary settings: read-only mode, loud mode, and color selection. The -p option is used to provide a text-only display in the case of a machine without the necessary dependencies to support a graphical interface. When this option is specified, the map would not be created and only a text output is printed to the console and then stored in the database. The -l option denotes “loud mode” and forces printing of the debugging information to the console. Finally, there are 11 color modes, ranging from -1 to 9 (Table 2). These modes allow manual control over the color of the line drawn between coordinate points. If no color is specified with the –c option, the color value defaults to 0, or red. The Application also supports artificial randomization for the lines drawn between points in the case of multiple points in close proximity, providing for lines of different colors at each position for enhanced readability.

Table 2

Color Options for Line Graphing

Mode #	Color of Line
-1	Random
0	Red
1	Green
2	Cornflowerblue
3	Yellow
4	Purple
5	Black
6	Orange
7	Grey
8	White
9	Pink

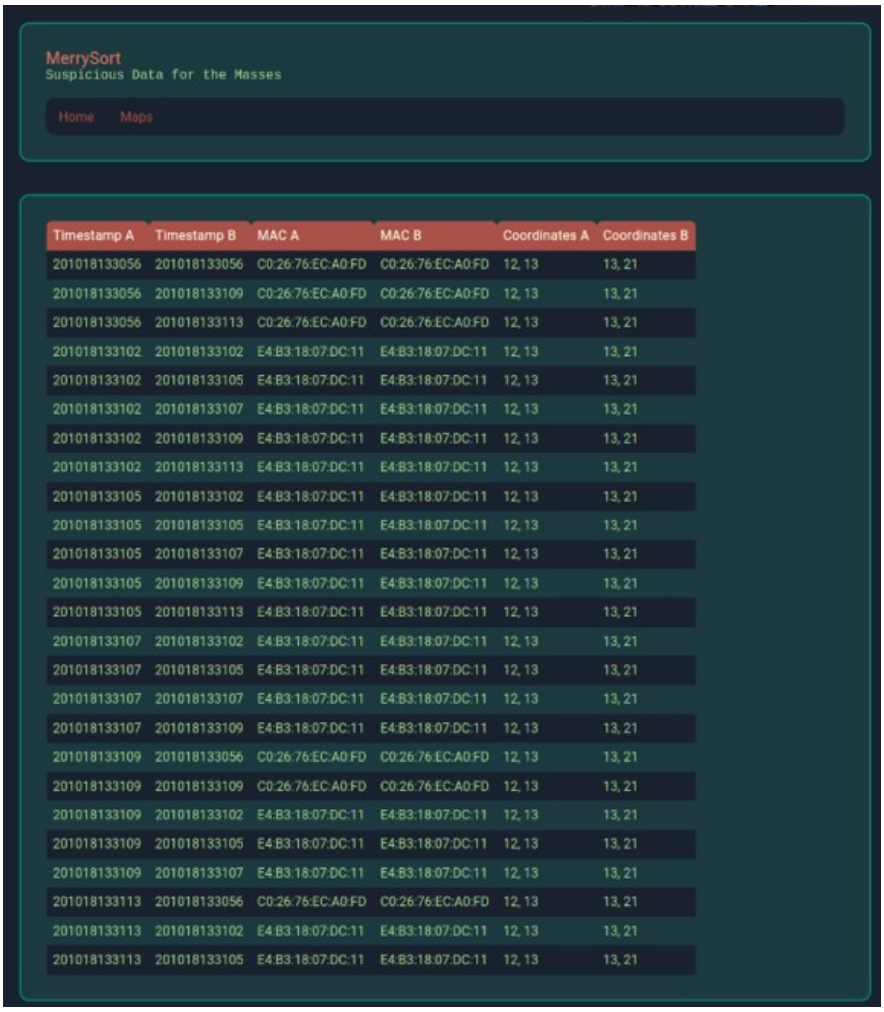
Final configuration can be performed with the PHP server. To initialize the website, a temporary PHP server can be initialized on a host machine. To do this, the following command is run:

```
php -S 127.0.0.1:8000
```

This command opens a PHP server on the local host at port 8000. To access it, the IP address and port are entered into the web browser. The result can be seen in Figure 4, displaying the homepage of The Application.

Figure 4

Retrieving Data from the Database with the PHP Frontend



Timestamp A	Timestamp B	MAC A	MAC B	Coordinates A	Coordinates B
201018133056	201018133056	C0:26:76:EC:A0:FD	C0:26:76:EC:A0:FD	12, 13	13, 21
201018133056	201018133109	C0:26:76:EC:A0:FD	C0:26:76:EC:A0:FD	12, 13	13, 21
201018133056	201018133113	C0:26:76:EC:A0:FD	C0:26:76:EC:A0:FD	12, 13	13, 21
201018133102	201018133102	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133102	201018133105	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133102	201018133107	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133102	201018133109	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133102	201018133113	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133105	201018133102	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133105	201018133105	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133105	201018133107	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133105	201018133109	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133105	201018133113	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133107	201018133102	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133107	201018133105	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133107	201018133107	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133107	201018133109	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133109	201018133056	C0:26:76:EC:A0:FD	C0:26:76:EC:A0:FD	12, 13	13, 21
201018133109	201018133109	C0:26:76:EC:A0:FD	C0:26:76:EC:A0:FD	12, 13	13, 21
201018133109	201018133102	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133109	201018133105	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133109	201018133107	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133113	201018133056	C0:26:76:EC:A0:FD	C0:26:76:EC:A0:FD	12, 13	13, 21
201018133113	201018133102	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21
201018133113	201018133105	E4:B3:18:07:DC:11	E4:B3:18:07:DC:11	12, 13	13, 21

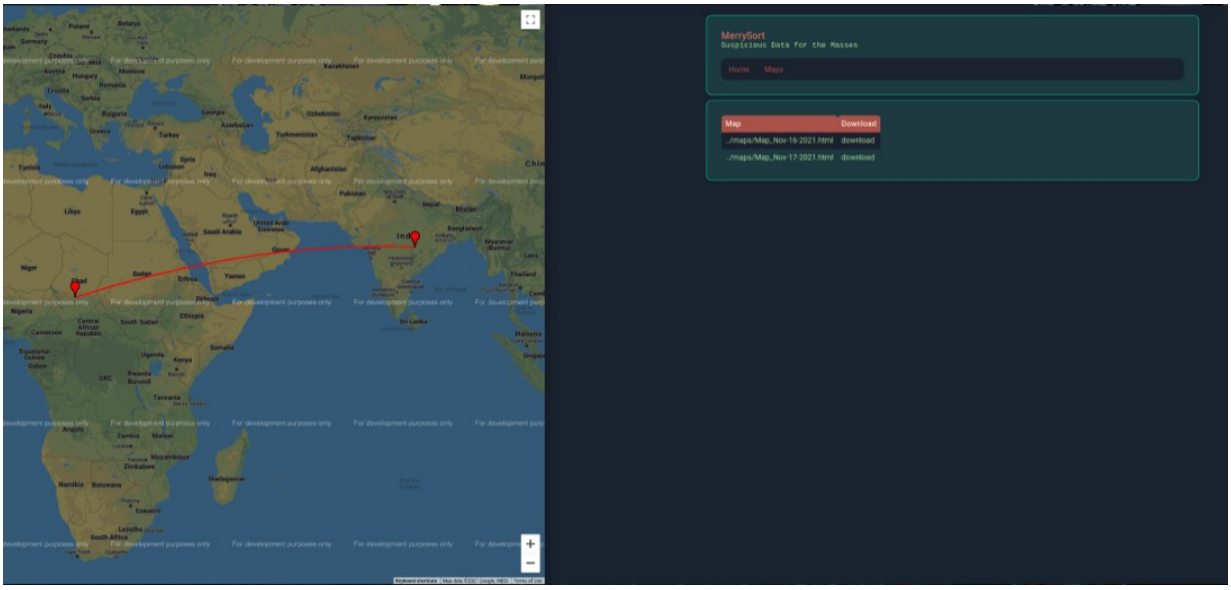
Note. The data shown on the command line in Figure 3 is comparable to the demonstration here; however, the GUI allows users to grab individual records by selection. The trial website is named “MerrySort”, a reference to a fellow researcher.

There are two additional items of interest for users on the front-end. First, the homepage stores all data in the database and automatically updates this information when there is a change in the database. The second page holds a running list of all the Google maps processed from past

trials and allows users to download them locally. Each map is named with a timestamp and iteration number as seen in Figure 5**.**

Figure 5

Downloading Data Sets



Note. The right half of the screen shows the auto-updating portion of the website that houses documents ready for download. The left half shows the result of downloading a file.

*The file shown is simply an example and has no relevance to actual geographical locations.

These configuration options allow users to display collected data with peers. They also enable long-term data storage and data obfuscation as the user is no longer directly interacting with the database and code. The addition of these options to the database, map, and front-end brings security, accessibility, and style to The Application.

Discussion

Apathetic voters exist because there is an assumption that votes are not counted fairly (Hardin, 2013). This distrust stems from a precedent of voter suppression and fraud that has coexisted with the democratic system (Alvarez et al., 2018). For this reason, it is imperative that data be published to incentivize maximum participation in the electoral process. The Application provides an effective way to collect polling data without posing a risk of interfering with the electoral process. This data can be used to reduce the amount of misinformation online. In the case of changes to the voting process, a variety of updates are suggested.

Effah and Debrah reported in their study that the polling security experience in Ghana demonstrated that the weakest element of any security chain is the human subject (2018). Even with scanning of numerous extremities and identification cards, mass fraud was still executed (Effah & Debrah, 2018). By logging proximal data, the proposed methodology does nothing to alter the voting process. Rather, the proposed methodology offers a way to log electronic events in proximity of the voting location. If an attacker were to manipulate scanner information, either with a frequency overload or similar attack, the voting process itself would not be affected because the proposed methodology has no connection to mechanics of the voting system. In the scenario of a malicious attack against one of the Bluetooth receivers, collected data could be used to show details of such manipulation and even provide the identity of the perpetrator. Timothy Nosco and colleagues present in their research that hackers typically use a depth-first search when selecting a target to compromise (2020). In other words, an attacker focused on disrupting an election will likely aim to manipulate the polling equipment, rather than scanning for additional connected resources; this alternative attack would be classified as a breadth-first search (Nosco et al., 2020). Directed authentication as used in Ghana provides attackers with a

clear target to compromise – monitoring proximal data reduces the chance that an attacker will be able to directly inhibit the voting process. In this way, logging electronic activity in the vicinity of polling locations provides an empirical way to log information while avoiding the missteps of directed authentication. The processing of The Application would further help combat false information propagated through social media.

When the results of an election get scanned and logged, it is important that people are made aware of the results. Paul Mena writes in the article “Cleaning Up Social Media: The Effect of Warning Labels on Likelihood of Sharing False News on Facebook” that implementing flags for false information has a beneficial effect in reducing the spread of misinformation (2020). Publishing the results of the poll scans would allow social media companies to implement the data into their flagging software and, in effect, reduce the amount of misinformation spread about election results. Additionally, all logged data will be published to a website hosted by The Application. Propagation of this information will benefit from social media platforms because of the resharing activity of the average user online (Mena, 2020). In this sense, publishing empirical data processed by The Application online can be used to combat misinformation. As with any theoretical implementation, this application operates on a variety of assumptions.

The methodology proposed operates on a variety of assumptions. These assumptions are grounded in the notion that the predominant form of voting in a country or region takes place at in-person voting centers. Furthermore, the process assumes that the polling locations are not frequented for more than one purpose during the election process. If individuals were to go to the location of polling twice, once to vote and another time to perform some other tasks, the methodology proposed would flag their repeated location as noteworthy behavior. To enhance

the application of the proposed methodology, motivation for future additions to The Application is provided.

Voter registration or personal ID could be paired with the MAC address and coordinate sets to reduce the possible ambiguity of gathered MAC address and coordinate pairs. Once a connection is made between two instances of voting from the same MAC address, the records of registration could be queried, and the perpetrators' identity could be further investigated. To do this, a column could be added to the database to hold voter registration numbers or ID information. Each time registration is processed at a polling center, the database would add the information to the associated row with MAC address and coordinate data. When The Application finds instances of identical MAC address appearances at different coordinate points, the database could be queried to display the associated ID number or registration information. Similarly, information like date, names, and administration credentials could be added. The goal of adding these columns is to enhance the precision of data gathered at polling locations. These suggestions still operate under the assumption that voters attend elections at a physical polling station. When the technology to take elections online matures to the point of implementation, alterations to accommodate these changes can be made to the Application.

There are many ways The Application can be expanded upon as technology at polling centers evolves. If voting were to go entirely online, The Application could take both MAC address and coordinate pairs as well as IP address and username pairs recorded from client machines as they contact the central polling server. This way, users could be tied to their online votes in a way that is similar to how users are associated in a geographical capacity. This method assumes that users are not able to spoof their IP addresses and that data from the connection socket on the host server is made available to additional processes on the host machine. This has

the possibility of opening a range of cybersecurity risks. These risks are no different from the longstanding concerns that have plagued online voting in the past, but it should be noted that these security issues will need to be addressed. One way to reduce the risk of IP address spoofing and exposing voter information is to use a virtualized private network (VPN). A VPN would provide both convenience in its application and also thorough encryption as a VPN is able to obfuscate important header protocol information (Singh & Gupta, 2016). As previously noted, increasing the security of an online election with such measures would increase the complexity of the voting process. Many eligible voters would be unable to set up such a connection without sufficient knowledge of the technology. For this reason, future studies should investigate the way online elections can be monitored while remaining secure and accessible.

There are many incremental changes that can be applied to the proposed methodology as the landscape of democracy changes, yet the foundational assumption of this study remains the same: A functioning democracy requires that members of the population be given access to a fair and free election that encourages participation without encouraging distrust in the political process. The way to alleviate the voter apathy that causes inherent distrust in the electorate process is to increase the transparency of elections. To provide this transparency, a methodology is proposed that monitors polling locations and logs voter MAC address and coordinate pairs. Logged data is open to the public to encourage empirical discourse and provide a platform to discuss polling anomalies. Transparency provided by the proposed methodology can be used in the evaluation of news media consumed by the public. Incremental updates in the proposed methodology can account for the ever-evolving technology that powers elections, preserving the freedom of the vote while ensuring the integrity of the voting process.

References

Abraham, H. J. (1952). What cure for voter apathy? *National Municipal Review, 41*(7), 346–357. https://doi.org/10.1002/ncr.4110410706

Al-Ameen, A., & Talab, S. A. (2012). E-voting systems vulnerabilities. *2012 8th International Conference on Information Science and Digital Content Technology, 1*(1), 67–73. IEEE.

Alvarez, R. M., Levin, I., & Li, Y. (2018). Fraud, convenience, and e-voting: how voting experience shapes opinions about voting technology. *Journal of Information Technology & Politics, 15*(2), 94–105. https://doi.org/10.1080/19331681.2018.1460288

Amer, M., & El-Gendy, H. (2013). Towards a fraud prevention EVoting system. *International Journal of Advanced Computer Science and Applications, 4*(4). https://doi.org/10.14569/IJACSA.2013.040423

Argersinger, P. H. (1985). New perspectives on election fraud in the gilded age. *Political Science Quarterly, 100*(4), 668–669. https://doi.org/10.2307/2151546

Bader, M. (2014). Do new voting technologies prevent fraud? Evidence from Russia. 2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14).

Benson, J. (2009). Voter fraud or voter defrauded highlighting an inconsistent consideration of election fraud. *Harvard Civil Rights-Civil Liberties Law Review, 44*(1), 14–42.

Berlinski, N., Doyle, M., Guess, A. M., Levy, G., Lyons, B., Montgomery, J. M., Nyhan, B., & Reifler, J. (2021). The effects of unsubstantiated claims of voter fraud on confidence in elections. *Journal of Experimental Political Science*, 1–16. https://doi.org/10.1017/xps.2021.18

Cherney, O. T., & Chaikina, Z. V. (2019). Technology of secure remote voting via the internet. *The 21st Century from the Positions of Modern Science: Intellectual, Digital and Innovative Aspects*, 275–281. Springer International Publishing. https://doi.org/10.1007/978-3-030-32015-7_31

Claassen, R. L., Magleby, D. B., Monson, J. Q., & Patterson, K. D. (2013). Voter confidence and the election-day voting experience. *Political Behavior, 35*(2), 215–235. https://doi.org/10.1007/s11109-012-9202-4

Deb, A., Luceri, L., Badaway, A., & Ferrara, E. (2019). Perils and challenges of social media and election manipulation analysis: The 2018 us midterms. *Companion proceedings of the 2019 world wide web conference,* 237-247. https://doi.org/10.1145/3308560.3316486

Effah, J., & Debrah, E. (2018). Biometric technology for voter identification: The experience in Ghana. *The Information Society, 34*(2), 104–113. https://doi.org/10.1080/01972243.2017.1414720

Esser, F., & de Vreese, C. H. (2007). Comparing young voters' political engagement in the United States and Europe. *The American Behavioral Scientist, 50*(9), 1195–1213. https://doi.org/10.1177/0002764207299364

Estehghari, S., & Desmedt, Y. (2010). Exploiting the Client Vulnerabilities in Internet E-voting Systems: Hacking Helios 2.0 as an Example. In *2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 10)*.

Fogarty, B. J., Kimball, D. C., & Kosnik, L.-R. (2022). The media, voter fraud, and the U.S. 2012 elections. *Journal of Elections, Public Opinion and Parties, 32*(1), 46–66. https://doi.org/10.1080/17457289.2019.1711383

Giammo, J. D., & Brox, B. J. (2010). Reducing the costs of participation: *Are states getting a return on early voting?. Political Research Quarterly, 63*(2), 295–303. https://doi.org/10.1177/1065912908327605

Gibson, J. P., Krimmer, R., Teague, V., & Pomares, J. (2016). A review of e-voting: the past, present and future. *Annals of Telecommunications*, *71*(7), 279–286. https://doi.org/10.1007/s12243-016-0525-8

Hardin, R. (2013). Government without trust. *Journal of Trust Research, 3*(1), 32–52. https://doi.org/10.1080/21515581.2013.771502

Kshetri, N., & Voas, J. (2018). Blockchain-enabled e-voting. *IEEE Software, 35*(4), 95–99. https://doi.org/10.1109/ms.2018.2801546

Krimmer, R. (2012). The evolution of e-voting: why voting technology is used and how it affects democracy. *Tallinn University of Technology Doctoral Theses Series I: Social Sciences, 19.*

Levin, I., Cohn, G. A., Ordeshook, P. C., & Alvarez, R. M. (2009). *Detecting Voter Fraud in an Electronic Voting Context: An Analysis of the Unlimited Reelection Vote in Venezuela* (Report No. 83). Caltech/MIT Voting Technology Project. http://hdl.handle.net/1721.1/96615

Meehan, Margaret F. (2004). *Voter apathy: why does it exist and what can be done to overcome it?* (Publication No. 1196) [Master's thesis, Rowan University]. Theses and Dissertations. https://rdw.rowan.edu/etd/1196

Mena, P. (2020). Cleaning up social media: The effect of warning labels on likelihood of sharing false news on Facebook. *Policy & Internet, 12*(2), 165–183. https://doi.org/10.1002/poi3.214

Nosco, T., Ziegler, J., Clark, Z., Marrero, D., Finkler, T., Barbarello, A., & Petullo, W. M. (2020). The industrial age of hacking. *29th USENIX Security Symposium*, 1129–1146.

Singh, K. K. V., & Gupta, H. (2016). A new approach for the security of VPN. *Proceedings of the Second International conference on Information and Communication Technology for Competitive Strategies*, 1–5. https://doi.org/10.1145/2905055.2905219

Tarasov, P., & Tewari, H. (2017). The future of e-voting. *IADIS International Journal on Computer Science & Information Systems*, 12(2).

Udani, A., Kimball, D. C., & Fogarty, B. (2018). How local media coverage of voter fraud influences partisan perceptions in the United States. *State Politics & Policy Quarterly*, *18*(2), 193–210. https://doi.org/10.1177/1532440018766907