Enhancing and Implementing Fully Transparent Internet Voting

Kevin Butterfield*, Huian Li*, Xukai Zou* and Feng Li†
*Department of Computer and Information Science
Purdue University Indianapolis, Indiana 46202, USA
Email: butterfk@iupui.edu
Email: {huili, xkzou}@cs.iupui.edu
†Department of Computer and Information Technology
Purdue University Indianapolis, Indiana 46202, USA
Email: fengli@iupui.edu

***Abstract*—Voting over the internet has been the focus of significant research with the potential to solve many problems. Current implementations typically suffer from a lack of transparency, where the connection between vote casting and result tallying is seen as a black box by voters. A new protocol was recently proposed that allows full transparency, never obfuscating any step of the process, and splits authority between mutually-constraining conflicting parties. Achieving such transparency brings with it challenging issues. In this paper we propose an efficient algorithm for generating unique, anonymous identifiers (voting locations) that is based on the Chinese Remainder Theorem, extend the functionality of an election to allow for races with multiple winners, and introduce a prototype of this voting system implemented as a multiplatform web application.**

***Index Terms*—Internet Voting; E-Voting; Anonymous; Transparent; Secret Sharing; Chinese Remainder Theorem; Mutually Restraining**

I. Introduction

Internet voting has been an active research topic in recent years. One clear advantage is allowing people to vote from anywhere at anytime, as long as they have a computer. Many experimental systems have been proposed; Some [17] require physical settings like voting booths. Some [16], [5], [15], [14], [19] require special hardware such as a tamper resistant randomizer, verification module, or smart cards. Two systems, Helios [2] and Prêt à Voter [9], have been used for university elections [3], [6]. Punchscan/Scantegrity [10] is a security enhancement for traditional optical scan voting systems and has been used for voting experiments. Internet voting has even been used for large scale political elections in several nations including Estonia [21].

As critics are quick to point out, conducting an election via the internet exposes it to a wide variety of attack [20]. They are correct: due to poor security practice [23], vulnerabilities in web browsers [13], or other methods of attack, internet voting does carry significant risk. However, it also brings with it significant benefits. The more convenient voting process has been shown to increase turnout in low-stakes elections (such as city-level officials) [11], its more centralized and automated nature allows the election to be managed by security professionals instead of untrained volunteers, and the ability to make results transparent exposes alterations to the tally and assures voters that their votes is in the tally and was counted correctly.

Meanwhile, traditional paper-ballot elections are extremely prone to error [4]. Elections are conducted with only cursory attention to security, with ballots stored in unattended, unmarked boxes and in the trunks of cars. Electronic ballot readers are no better: they often run proprietary firmware that is vulnerable to attack, and just like the paper ballots they are frequently stored with no concern for security, in places like school gymnasiums or officials' garages. Internet voting carries risks, that much is certain, but the traditional alternatives are just as bad or even worse, and internet voting brings great advantages as well [7].

Unlike most other online applications, internet voting has strict requirements, including verifiability, accountability, receipt-freeness, and coercion-resistance. These requirements pose many challenges, and some of them are even conflicting: for example, an inherent tradeoff exists between anonymity versus verifiability. A perfectly anonymous voting system would perfectly preserve voter privacy, but since their ballots have no connection at all to themselves, voters would have no ability to verify that their vote was counted correctly, eliminating one of the key advantages of internet voting.

While many internet voting systems exist, most obfuscate the transition from casting ballots to tallying results. For example, mixnet-based voting systems [8] such as Helios and Prêt à Voter display the full contents of all ballots, but use a secret permutation to disassociate them from the voters who cast them. They can assure voters their ballot is in the tally only with a mathematical proof. Proofs may satisfy a mathematician or computer scientist, but most elections are not held strictly among such people. Estonia's system also lacks transparency, as pointed out by election observers [21]. Additionally, Helios, Prêt à Voter, and Scantegrity have been found to be exceptionally low in usability [1]. Seeing a web page tell voters their ballot was counted or seeing a long display of math is not as assuring as being able to look directly at their ballot within the tally, know that it is theirs (while no one else does) and see for themselves that its choices are exactly as they intended them.

This is the author's manuscript of the article published in final edited form as:
Butterfield, K., Li, H., Zou, X., & Li, F. (2015). Enhancing and Implementing Fully Transparent Internet Voting. In 2015 24th International Conference on Computer Communication and Networks (ICCCN) (pp. 1–6). http://doi.org/10.1109/ICCCN.2015.7288408

The protocol in the INFOCOM'14 paper [25] is fully transparent: it removes the obfuscation or gap in the transition from casting secret ballots to tallying individual votes. Thus, it offers two types of assurance to voters. One is vote-casting assurance on "secret" ballots. Every voter can identify their ballot as their own and ensure the contents are what they cast, providing individual verification in a way that less transparent protocols cannot. This does not endanger the privacy of any voter, but because nothing about the election's results is available to anyone until all votes have been cast, this allows for anyone to ensure that the ballots are not tampered with after the election results become known. The other assurance relates to vote-tallying: a voter is assured that the vote is viewably counted in the final tally. Anyone can see the contents of all votes and verify for themselves that the tally is accurate, providing universal verification.

Unfortunately, the protocol as it was described at INFOCOM suffers from two main issues. First, the location anonymization scheme that allows individual verifiability may not be efficient. It requires multiple synchronized rounds of activity between all voters, which could slow the election to a halt and annoy the voters. Additionally, the protocol currently could only handle one-winner scenarios, while in many real life elections multiple winners are allowed.

In this paper, we present our contributions to the aforementioned issues. First, we propose an efficient algorithm for generating unique, anonymous voting locations that runs in constant time without need for synchronization among voters, based on the Chinese Remainder Theorem. Second, we extend the functionality of an election to allow for races with multiple winners. Third, we introduce a prototype of this voting system implemented as a multiplatform web application.

The rest of this paper is laid out as follows. In section II we briefly introduce our voting protocol included in INFOCOM'14 [25]. In Section III we discuss the enhancements we have made to the protocol. In Section IV we introduce the implemented prototype we have developed. Section V compares our improved protocol with the original and discusses scalability. We conclude in Section VI.

II. Overview of the Protocol

In this section, we give an overview of the mutually restraining e-voting protocol presented in the INFOCOM'14 paper [25], summarizing the assumptions and stages involved in the protocol.

A. Assumptions

Suppose there are N $(N > 3)$ voters denoted V_i where $1 \leq i \leq N$, M candidates being voted on, and 2 authorities (called Collectors) denoted C_1 and C_2 who have *conflicting* interests that prohibit them from colluding but who cooperate to follow the protocol. The assumption of multiple conflict-of-interest authorities was previously proposed by Moran and Naor [17], and applied to real world scenarios like the multi-party political system in the US. The protocol can support any number of collectors ≥ 2, but we limit the count to 2 in this paper for simplicity. The protocol ensures that neither collector working alone can produce a partial tally (seeing in-progress results before the election is complete) with the information they have, or decrypt any voter's secret ballot.

The majority of voters are assumed to be benign. This assumption is reasonable since the colluding majority can easily control the election result otherwise. Secure unicast channels between a voter and each authority are assumed. Public key based cryptosystems can be applied to provide such channels.

B. Five Stages of the Protocol

1) Initialization: Voter V_i's choice is represented by a bit string of length M, called a *voting vector*, where each bit corresponds to a candidate. V_i chooses exactly one candidate to vote for, sets that bit to 1, and all others to 0. V_i then obtains a unique, secret number L_i called a *voting location*. The concatenation of all voting vectors ordered by voting location produces a tallied voting vector that allows universal verifiability and, since voting locations are known to the voter and only to the voter, individual verifiability. By left-shifting each voting vector by $L_i \times M$ bits, this concatenation may be achieved by summing all voting vectors.

2) Vote Casting: Each voter encrypts their shifted voting vectors to create their *secret ballot* using Simplified-(N, N)-Secret Sharing [24], an additively homomorphic cryptosystem that ensures that the sum of all secret ballots is the same as the sum of all plaintext voting vectors. Generation of secret cryptographic shares is split among the collectors, so no one party ever possesses enough information to decrypt any voter's submission, and no partial tally is possible because the final sum is created only when all voters vote. Secret ballots may be made public on an In-Progress Bulletin Board so that anyone may see encrypted submissions as they come in.

3) In-Process Check and Enforcement: Voters share the bit-wise reverse of their shifted voting vector the same way. Since voters must set exactly one bit to 1 and all others to 0, the product of (plaintext) forwards and reverse equals $2^{N \times M-1}$ regardless of which bit the voter chooses. The collectors may obtain this product without exposing the plaintext using Secure Two Party Multiplication [18], thus ensuring that a malicious voter cannot vote multiple times and cannot deliberately avoid voting.

4) Collection / Tally: The collectors each calculate the sum of all secret ballots to produce the tallied voting vector and analyse it to determine the winner(s) of the election. Additionally, because secret ballots are made public, any third party can perform the same calculation.

5) Verification: As mentioned, any third party can independently tally the results to ensure that they are accurate and, since partial tallies are impossible, any third party may check that submissions at the end are not altered from their original state, ensuring the results have not been tampered with, providing universal verifiability. Additionally, because individual ballots (voting vectors) are ordered by voting location, each voter may find his or her ballot within the tally and verify that

the choice is exactly what they intended, providing individual verifiability.

C. Drawbacks

The original algorithm for location anonymity was highly secure (only direct collusion between collectors could violate voter privacy; just as with voting) but it was inefficient. Voters chose their locations at random, shared them as they would voting choices, and if any voters chose the same number all voters must go again until there is no collision. This requires an unknown number of synchronized rounds of activity between all voters, and is not ideal.

Additionally, since exactly one bit must be cast by each voter, elections that involve multiple votes among a pool of candidates were not supported. In Section III we introduce a new location finding algorithm that runs in a single step with no need for synchronized sessions of activity, as well as a workaround to the multiple voting issue.

III. Protocol Enhancements

In this section, we introduce the new enhancements to the protocol. We have created a new algorithm for generating voting locations, that runs in constant time with no need for synchronized activity between voters. Additionally, we have developed a way to allow voters to vote for multiple distinct candidates without violating the protocol.

A. Enhanced Location Anonymization

Each voter must have a unique voting location. This is defined as a number L_i that is unique among all voters, yet is known only to the voter to whom it belongs to. We thus require a function, $f(A, B, i) = L_i$ that produces unique output L_i for V_i with unique inputs A and B. A and B are sets such that the source of either must not know anything about the contents of the other.

The algorithm that provided this required multiple synchronized sessions of activity from every voter. Here we propose an algorithm that assumes collectors will not collude with each other *or with any voter*, but that runs in constant time with no need for synchronization between voters. Alone the collectors still have no ability to deduce any voter's location.

1) Chinese Remainder Theorem based Algorithm: In an election with N voters, each collector C_j chooses a positive prime number p_j and creates a set U_j of unique numbers such that $|U_j| = N$ and $\forall u \in U_j : p_j > u$ where $j = \{1, 2\}$. Each voter V_i receives p_j and u_{ji} from both collectors where u_{ji} is chosen from U_j; thus each voter receives two primes (all voters get the same primes) and two arbitrary numbers (each collector never sends the same number to two different voters). V_i thus has p_1, p_2, u_1, u_2, and finds the unique L_i using the following equation:

$$a = u_1 p_2 (p_2^{-1} \bmod p_1)$$

$$b = u_2 p_1 (p_1^{-1} \bmod p_2)$$

$$L_i = (a + b) \bmod p_1 p_2$$

This is an application of the Chinese Remainder Theorem (CRT), which states that for pairwise relatively prime positive integers $p_1 \dots p_r$, distinct integers $u_1, \dots u_r$, and a system of congruences:

$$\begin{aligned} x &\equiv u_1 (\bmod p_1) \\ x &\equiv u_2 (\bmod p_2) \\ &\dots \\ x &\equiv u_r (\bmod p_r) \end{aligned}$$

There is a unique solution modulo $K = p_1 \times p_2 \times \dots \times p_r$ Thus, a unique voting location can be obtained by solving for x, and as long as no collector obtains the information generated by the other, all locations found are secret. While CRT requires only *relatively* prime integers p_j, requiring prime numbers ensures that all p_j are relatively prime without any need for one collector to be aware of any other's choice. A proof of CRT is equivalent to proving the function X is a bijection where X is a mapping: $\mathbb{Z}_K \to \mathbb{Z}_{p_1} \times \dots \times \mathbb{Z}_{p_r}$ and defined as $X(x) = (x \bmod p_1, \dots, x \bmod p_r)$. A detailed proof of CRT itself can be found in [22].

This algorithm scales to any number of collectors without issue; as usual, only two collectors are mentioned for the sake of simplicity. If one voter receives the same u_{ji} from both collectors, that does not cause any problems. If voters receive the same p_j from both collectors, the algorithm cannot proceed, and voters must report an error to the collectors, who may simply generate new prime numbers and try again. This is of little concern: if the collectors are generating cryptographically secure prime numbers the likeliehood they choose the same one is all but nonexistent, and even if they do, the first voter to log in will report this error automatically, it will be corrected with very minimal work, and all future voters will not be affected.

2) Location Consolidation: The algorithm produces highly discontinuous results. While guaranteed to be unique, the outputs are by no means guaranteed to be close to each other. This significantly increases the size of the voting vectors because of the empty space, and could confuse voters who expect N locations to be numbered $1, 2 \dots, N$.

An optional step allows for continuous locations to be generated from the discontinuous ones. A voter's Absolute Location is defined as the number generated by the algorithm above, and if all voters treat theirs as the index of a candidate and "vote" on them, doing no location shifting, the resulting tally (called the Location Vector) will allow voters to reduce their Absolute Location to a Relative Location, where the set of all N Relative Locations will be exactly $\mathbb{Z}_N$.

Each voter obtains the Location Vector, finds the bit that corresponds to their Absolute Location, and counts all the 1-bits that precede it. This Relative Location is the number of Absolute Locations smaller than one's own. This is still guaranteed to be unique and anonymous, and adding 1 to all locations prior to display in user interfaces produces the desired $1, 2, \dots, N$. This process is visualized in Fig. 1.

3) Security: The new algorithm breaches voter privacy if a collector colludes with any voter. The old algorithm required

Obtain binary Location Vector:	10100101
Calculate Absolute Location L:	5
Find 2^L within vector (bold):	10**1**00101
Count 1s preceding own:	10**1**00<u>1</u>0<u>1</u>
Relative Location is this count:	2

Fig. 1. Consolidating Locations Example

collectors to collude together, which is easier to secure against. Thus, the new way is appropriate only when collectors can be trusted not to collude with *anyone*, (alone they are still unable to breach any voter's privacy) the voters can be trusted not to collude with collectors, (willingly or unwillingly) or the secret data for location is not made available to anyone who might collude. (Unlike ballots, nothing but the Location Vector must be made public to achieve full transparency)

B. Multiple Winners

A special form of election involves multiple distinct choices from a pool of candidates instead of a single choice. The nature of voting vectors prohibits this: voters must choose exactly 1 bit before encryption. Our solution forms such an election as a set of distinct elections: one for every choice voters are allowed. These sub-elections are called **instances**; during vote tallying, the tallies of each election's instances are aggregated, and that aggregated tally is the one used to determine winners. A "normal" election where voters get one choice may be treated as an election with one instance.

The opposite case, where empty ballots are permitted to be cast, is easily solved with no alteration to the protocol: introduce an "Abstain" candidate to the election.

IV. Implementation

We have developed a prototype application suite that allows for functional demonstration/use of the enhanced protocol, implemented in PHP and Python.

A. Roles

Three distinct groups of people are involved in an election. The **administrator** defines the election itself: what races are being voted on, who the candidates are, whether or not a given race is one of the multi-winner cases described in Section III-B, what the time limits are (if there are any) to each phase of the election, and who the collectors are (in the form of web addresses). This information is stored in the JSON format and distributed to the collectors.

The **collectors** maintain a database that keeps track of voter usernames, the keys they use to encrypt their ballots, and the uploads they submit. With the exception of the keys, this information is publicly viewable on a page discussed in Section IV-C. They run an application that automates all steps of the protocol; human operators are needed only to set up the application and ensure that everything runs as expected.

The **voters** are the people who actually vote for the candidates. They are identified by a unique username, which may be made public, which is secured with a password. At each phase of the election, the voters use a web page to give the needed information to the collectors. Since collectors interact with voters independently of each other, the application obtains and sends secure data to each of the collectors via asynchronous javascript.

B. Phases

Activity is broken into four phases, three of which involve the voters, one of which is optional, with a strictly observational fifth phase at the end.

1) Preliminary: Before the election is opened to voters, the administrator must create the election file as described above and distribute it to the collectors. The collectors, meanwhile, must set up all necessary files on their servers. A web application has been developed that allows the administrator to easily specify necessary information for the election file, and which automatically distributes that file to the collectors. Once obtained, the collectors initialize their databases and open Registration.

2) Registration: Voters choose a username and password to send to the collectors. If the username is available, it is stored in the databases, and that voter is now registered. For private elections, this step could be integrated with a credential check or could be bypassed altogether and voters could be pre-registered by election officials. Whatever security checks are necessary for registration occur at a higher level than the election itself.

The administrator may specify a time when Registration ends, or the collectors may manually force it to end using a menu option. Either way, when Registration closes, the collectors generate a share matrix for the location-vote and, based on the settings of the election, open either Confirmation or Voting.

3) Confirmation: This is an optional phase that serves to consolidate locations as discussed in Section III-A2. Voters see only a prompt for username and password, then a response. However, the application quietly executes the steps to consolidate locations. Confirmation ends when all registered voters confirm or an optional timeout occurs, whichever comes first. Collectors tally and publish the Location Vector.

If this behavior is not needed, Confirmation may be skipped with a setting in the election file. If pre-registration is in use, Confirmation may act as voter registration. For example, if a university wishes to conduct an election among its faculty, a list of qualified voters could be generated and the Confirmation phase could be used to establish voter participation. Qualified but uninterested voters would not "register", and would be disqualified from the election after the timeout occurs.

4) Voting: Once Confirmation is finished, collectors compensate for any unconfirmed voters and open Voting. Alternatively, if Confirmation is not being done, collectors may open Voting immediately after Registration closes with no additional work done. During this phase, voters log in using their usernames and passwords and are presented with the names of the races and radio buttons corresponding to candidates. Behind the scenes the application executes all that is necessary

to obtain voting location and prepare for encryption, which may occur while the voter is making his or her choices. When the voter is finished, a voting vector is generated for each instance of each race and sent to the collectors. When all voters vote or an optional admin-specified timeout occurs, the collectors independently tally and publish the results and the election ends. If collectors do not produce exactly the same tally results, misbehavior has occurred at some level.

C. Transparency

The collectors maintain two web pages for the publishing of data: first a Raw Results page that displays encrypted data in real time during the election, and secondly an Interpreted Results page that displays the tallied results after the election is over. The Raw Results page is opened at the beginning of Registration and acts as a dynamic bulletin board: during Registration, it displays taken usernames as they are registered. During Confirmation, it displays in yes-no format whether or not that voter has confirmed. During Voting, it displays the encrypted ballots (numbers) for each instance of each race as they are cast. A snapshot of this page midway through the Voting phase is pictured in Fig. 2.

The Interpreted Results page is opened at the close of the election when the ballots are tallied and acts as the actual results-viewer. At the top, it displays the winner(s) of each race along with how many votes they obtained. Below, it displays each location and the choices that voter made. Examples of both are shown in Figures 4 and 3. Thus, anyone may verify that the tallied results match perfectly the actual votes cast, and any voter may verify that their location's choices are exactly the choices they made. As the encrypted ballots are visible in the Raw Results page, anyone may run their own tally and verify that the interpreted locations accurately match what was cast. All of these verifications are done by simple observation that anyone, even non-technical people, can perform and understand.

V. Comparison and Discussion

In this section, we give a comparison between our newly implemented prototype and the INFOCOM'14 protocol [25]. Then we briefly discuss the scalability of the protocol and implementation.

TABLE I
Comparison between the INFOCOM'14 protocol and ours

Protocols	This	INFOCOM'14
Location anonymization	1 round	multiple rounds
Allowed winners	multiple	limit to 1

A. Protocol Comparison

Table I lists the major difference of our implementation comparing to the INFOCOM'14 protocol [25]. Since the enhanced location anonymization scheme in our implementation requires only one round of interaction between voters and collectors, thus dramatically decreases the communication and synchronization cost. In addition, our implementation

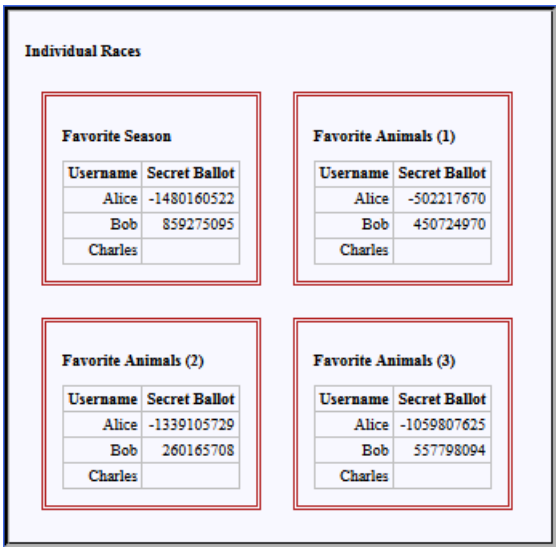

Fig. 2. In-Progress Bulletin Board

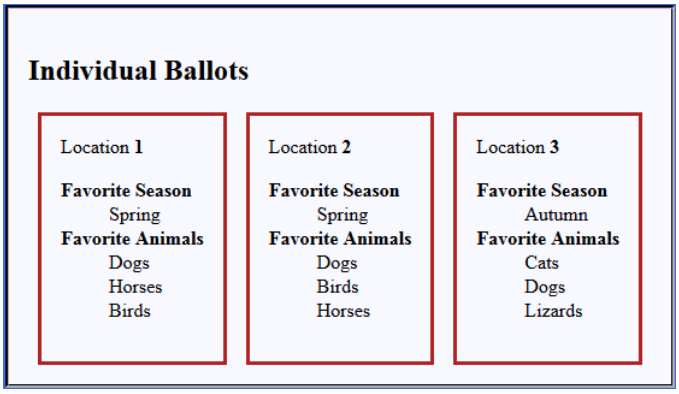

Fig. 3. Full contents of all ballots

Fig. 4. Displaying winners (including ties)

allows multiple winners, so it can be applied to more election scenarios in real life.

B. Scalability

The amount of random numbers that must be generated for (N,N) Secret Sharing grows $O(N^2)$ and the size of the ballots scales $O(N)$ in the worst cases (the voters with the largest locations). For a very large number of voters this is problematic: the wait times resulting in sending enormous ballots could become unacceptable. This is analogous to the traffic involved with conducting a very large traditional election through a single polling place, and has the same solution: a **hierarchical voting structure**.

Rather than grouping all voters together, they would be divided into precincts. Based on a survey by the US EAC on voting booth based elections, the average precinct size for a presidential election is approximately 1100 registered voters as of 2004 [12]. Each precinct would conduct a separate tally and merge the results. This would result in share quantity and ballot sizes that scale with the *precinct size*, which can be bounded at a level that produces acceptable performance in all cases. The same collectors could manage all precincts, virtualizing each one, or multiple pairs of collectors could exist with precincts distributed among them. This latter case introduces additional security: in a worst case scenario only a fraction of the voters' privacy would be exposed.

VI. Conclusion

We have presented our new algorithm, based on the Chinese Remainder Theorem, for generating unique voting locations, as well as our strategy for reconciling universally verifiable voting vectors with multiple voting. We have introduced and explained our prototype web implementation, including previewing its interface and discussing its scalability.

The prototype implementation is exactly that: a prototype. It is fully functional: it accurately demonstrates the protocol and an election may be carried out using it; but it is still in active development and many security vulnerabilities have not yet been addressed. Along with addressing these, we are continuing to refine the web interfaces for better usability.

Once the prototype is complete the system will be available for use by anyone who wishes to run an election. We plan to debut it for our department's faculty tenure and promotion voting process, and it is appropriate for use by any organization that wishes to have fully transparent internet voting, secured by mutually restrained authorities.

References

[1] Claudia Z. Acemyan, Philip Kortum, Michael D. Byrne, and Dan S. Wallach. Usability of voter verifiable, end-to-end voting systems: Baseline data for helios, prêt à voter, and scantegrity ii. In *2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14)*, San Diego, CA, August 2014. USENIX Association.

[2] Ben Adida. Helios: Web-based open-audit voting. In *USENIX Security Symposium*, volume 17, pages 335–348, 2008.

[3] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Security Symposium, 2009.

[4] Davide Balzarotti, Greg Banks, Marco Cova, Viktoria Felmetsger, Richard A. Kemmerer, William Robertson, Fredrik Valeur, and Giovanni Vigna. An experience in testing the security of real-world electronic voting systems. *IEEE Transactions on Software Engineering*, 36(4):453–473, 2010.

[5] O. Baudron, P. A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proc. of 20th ACM Sym. on Principles of distributed computing*, PODC '01, pages 274–283, 2001.

[6] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter YA Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. Using prêta voter in victorian state elections. In *2012 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE'12)*, 2012.

[7] Kevin Butterfield and Xukai Zou. Analysis and implementation of internet based remote voting. In *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on*, pages 714–719. IEEE, 2014.

[8] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. of the ACM*, 24(2):84–90, Feb. 1981.

[9] D. Chaum, P. Ryan, and S. Schneider. A practical voter-verifiable election scheme. In *Proc. of the 10th European Conf. on Research in Comp. Security*, ESORICS'05, pages 118–139, Milan, Italy, 2005.

[10] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *Security & Privacy, IEEE*, 6(3):40–46, 2008.

[11] Sang Ok Choi and Byung Cho Kim. Voter intention to use e-voting technologies: Security, technology acceptance, election type, and political ideology. *Journal of Information Technology and Politics*, 9(4):433–452, 2012.

[12] The U.S. Election Assistance Commission. Polling places 2004 general election: EAC election day survey. 2006.

[13] Saghar Estehghari and Yvo Desmedt. Exploiting the client vulnerabilities in internet e-voting systems: Hacking helios 2.0 as an example. In *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX, 2010.

[14] M. Hirt. Receipt-free k-out-of-l voting based on elgamal encryption. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter A. Ryan, and Josh Benaloh, editors, *Towards Trustworthy Elections*. 2010.

[15] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proc. of Info. Security and Cryptology*, volume 2971, pages 245–258, 2003.

[16] B. Lee and K. Kim. Receipt-free electronic voting through collaboration of voter and honest verifier. In *Proc. of JW-ISC*, pages 101–108, 2000.

[17] T. Moran and M. Naor. Split-ballot voting: Everlasting privacy with distributed trust. *ACM Trans. Inf. Syst. Security*, 13(2):16:1–16:43, 2010.

[18] S. Samet and A. Miri. Privacy preserving ID3 using Gini index over horizontally partitioned data. AICCSA '08, pages 645–651, 2008.

[19] Scytl. White paper: Auditability and voter-verifiability for electronic voting terminals.

[20] Barbara Simons and Douglas W. Jones. Internet voting in the u.s. *Communications of the ACM*, 55(10):68–77, 2012.

[21] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J Alex Halderman. Security analysis of the estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 703–715. ACM, 2014.

[22] Douglas R. Stinson. *Cryptography Theory and Practice*. Discrete Mathematics and its Applications. Chapman and Hall / CRC, third edition, 2006.

[23] Scott Wolchok, Eric Wustrow, Dawn Isabel, and J. Alex Halderman. *Attacking the washington, DC Internet voting system*, pages 114–128. Springer, 2012.

[24] X. Zhao, L. Li, G. Xue, and G. Silva. Efficient anonymous message submission. In *IEEE Int. Conf. on Comp. Comm., INFOCOM'12*, volume 2012, pages 2228–2236, May 2012.

[25] Xukai Zou, Huian Li, Yan Sui, Wei Peng, and Feng Li. Assurable, transparent, and mutual restraining e-voting involving multiple conflicting parties. In *INFOCOM, 2014 Proceedings IEEE*, pages 136–144. IEEE, 2014.