# E2EV Comparative Analysis

The following is a preliminary comparative discussion of existing E2E systems. These systems are not competitive, per se, as they are either not commercially available or supported in manner similar to what we envision with Open E2E, or targeted to actual governmental elections, however, they are systems of interest.

This section presents a broad-brush comparative analysis of E2EV voting systems including a preliminary assessment of 3 apparently “true” remote end-to-end verifiable (E2EV) voting systems. As no structure, set of questions or criteria for such an analysis was previously defined, it can only be deemed **preliminary, descriptive and analytic;** it is not prescriptive and does not offer any judgments or recommendations as to better or worse voting systems.

**Major Findings**

The analysis of system features found that only ***3 systems could satisfy all the E2EV*** properties. This conclusion is contestable, however, as some scientists would consider each of these systems to have actually embraced tradeoffs that could render no system a complete E2EV system.

A number of additional systems are under development currently that hold promise and should be watched. We term these ***“Horizon”*** systems. They are not covered in this document.

**No E2EV system has solved all security issues,** and it could be that none is likely to solve them all until the current public Internet is re-architected and re-engineered for security.

**Definition of E2EV Voting Systems**

For this analysis, it was agreed that an end-to-end verifiable (E2EV) voting system requires these properties:

(1) **End-To-End Integrity**, specifically meaning that the voter’s vote selections are:

1. Recorded as intended
2. Cast as recorded
3. Counted as cast

(2) ***Verifiability*** of all integrity properties, by the voter (and a smaller set of properties verifiable by the public, to preserve voter privacy in vote choices); and achieves

(3) ***Software Independence.***

“Software independence” is a property that Ron Rivest and John Wack first identified as a possible cure for software vulnerabilities that could cause inaccuracies in vote totals [11, 12]. Phillip Stark and David Wagner have further explained the property:

A voting system is strongly software-independent, if an undetected error or change to its software cannot produce an undetectable change in the outcome, and we can find the correct outcome without re-running the election. Strong software-independence does not mean the voting system has no software; rather, it means that even if its software has a flaw that causes it to give the wrong outcome, the overall system still produces “breadcrumbs” (an audit trail) from which we can find the true outcome, despite any flaw in the software.

Systems that produce voter-verifiable paper records (VVPRs) [for instance, voter marked paper ballots] as an audit trail are strongly software-independent, provided the integrity of that audit trail is maintained, because the audit trail can be used to determine the true outcome.

**Disclaimer:** this report by no means represents a definitive, peer-reviewed body of research. Rather it is a starting point. There may be errors or inaccuracies.

**True E2EV Voting Systems**

Demonstrate the following properties: (1) End-To-End ***Integrity, specifically meaning that the voter’s vote selections are:***

1. Recorded as intended
2. Cast as recorded
3. Counted as cast

(2) ***Verifiability*** of all integrity properties, by the voter (and a smaller set of properties verifiable by the public, to preserve voter privacy in vote choices); and achieves

(3) ***Software Independence.***

**System 1 - Remotegrity**

***Introduction***

During the mid-2000s, the remote voting system **Remotegrity** was proposed and developed by Filip Zagorski, joined by a team of other distinguished cryptographers that included Poorvi Vora, Richard T. Carback, David Chaum, Jeremy Clark & Aleksander Essex. [R.1]. Remotegrity is an end-to-end verifiable (E2EV) absentee voting system, which has been architected to work in concert with an in-person/precinct voting system called ***Scantegrity*** (a paper-based system).

Remotegrity adapts the “code-voting1” approach and features found in the “mother-system” Scantegrity. Using this approach, Remotegrity seeks to protect voter privacy, as well as provide resiliency against any malware-induced software modifications, election official or intruder manipulations of vote tallies, and other potential injuries to election integrity.

Remotegrity was deployed in Takoma Park, Maryland, in 2011, where election officials used it in a trial of both Scantegrity & Remotegrity systems. In addition, the trial included ***ranked-choice voting*** *[R.1]*methods. In this report we have reviewed the version of Remotegrity used in the 2011 Takoma Park local election [R.2]. This analysis is based on the publicly available research papers and interviews with the individual architects and developers.

*Notes:*

[R.1] “Ranked-choice voting” (also called preferential or “instant run-off voting” (IRV) requires voters to rank their choices among the available candidates. While IRV election can be structured with different rules (for instance, at least two distinct types of IRV structures can be created, including e.g., directing voters to “rank your top three choices of candidates by placing a ‘1’ indicating your top choice, and ‘3’ your third choice among the field of 8 candidates,” or “Rank each of the 8 candidates in the order of your preference, giving a ‘1’ to your top choice candidate and an ‘8’ to your least preferred candidate.” The virtues and evils of IRV are well beyond the scope of this Report. Suffice it to say that IRV presents substantial voter education hurdles, and that the test-running of Remotegrity in concert with IRV eliminates the ability to draw any sound conclusions about the usability or deficits in either innovation.

[R.2] Code voting helps in achieving privacy by replacing all the elements on a ballot by codes, which are cryptographically generated.

**1. Core Architecture & Operation**

1.1 Basic Architecture & Design

Remotegrity uses both the online component as well as the paper component from ***Scantegrity.*** Paper ballots are printed along with an additional feature called the ‘Authorization card’. The Authorization card contains codes which are hidden under a scratch off and which are used in casting the vote (‘Auth Code’) and finalizing the casted vote (‘Lock Code’).

These two printed components are sent to the registered voters via any postal service and the online component includes voters accessing the voting site, casting the vote, checking their vote and finally viewing the results on a publicly accessible Bulletin Board (which is another website).

Remotegrity is developed in Java. The databases (for example: MySQL) can be hosted on a cloud infrastructure. The system requires a separate offline server that is not connected to the Internet to check the validity of the voters’ submitted vote & authorization codes. This server is dedicated to maintaining all the cryptographic keys and validation signatures. Remotegrity developers architected the system for high security and data integrity and tested it for security gaps prior to its deployment in Takoma Park.

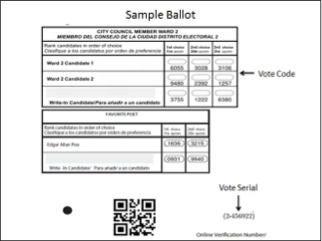
Remotegrity is designed to ensure that voters will receive unique codes for the same candidate [R.3]. Remotegrity is designed so that an election computer receiving vote codes is able to check whether a code corresponds to a valid choice on the ballot, without knowing to which vote it corresponds. This maintains vote privacy while preventing malware on the voter’s computer from changing the vote. The vote codes are cryptographically calculated using the keys the election officials generate via automated processes.

1.2 Voting Process

Printing Ballots: Before ballots are printed the unique codes for every candidate are calculated along with the Auth and Lock codes for each voter. The cryptographic values and their relation to candidates and voters are generated and stored on an offline validation server.

The ballot paper and the authorization card also contain a ‘Vote Serial’ and ‘Ack Code’ respectively. These are used to validate that the codes entered came from a particular ballot and an authorization card. The ballots and the authorization cards are printed and mailed to all the voters. A sample ballot and the sample authorization card are shown in Fig 1 and Fig 2 below.

Fig. 1 Sample Ballot





Note:

[R.3] The system utilizes distributed key generators & pseudo-random number generators for generating the codes that are printed on the ballots and authorization cards.

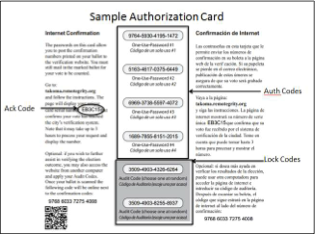




Fig. 2 Sample Authorization Card

Ballot casting: The voters will use the ‘Vote Code’ as shown in Fig. 1 of their preferred candidate and enter it on the voting portal. Upon entering their choice, the voter is asked to enter an ‘Auth Code’, which is under a scratch off on the authorization card. This ‘Auth Code’ is a one-time password (single-use password) used to validate the voter.

There are 4 Auth codes given to a voter under a scratch off, which are helpful in case is the voter needs to attempt to vote again (because their computer performs a denial of service attack or does not respond correctly to instructions) or in case there is a dispute over the values entered.

Once the values are entered, the voter is asked to wait for a few hours and revisit the election web site, allowing the election system and officials to verify the validity of the codes entered by the voter. Upon verification the election system signs the code entries and displays it back to the user. All communication with the offline computer is performed over an “air gap”; that is, a human transfers the data to and from the offline computer at regular intervals (of about 3-4 hours for the Takoma Park election). This prevents any malware from online computers from signing false data. Then the voter is shown his choice along with the Vote Serial and the Ack Code.

The Ack Code is printed on the authorization card (shown in Fig 2) and is never entered by the voter, and hence, if it is correct, it could not have been generated by any entity other than the election computer. The presence of a correct Ack Code on the election website hence communicates to the voter that the entered vote codes were valid and not manipulated by the computer used by the voter for voting.

If the voter is satisfied, he/she has to scratch off one of the ‘Lock codes’ and enter it to use the value to finalize/freeze the vote. Once the ‘Lock Code’ is used the vote cannot be changed. Fig 3 shows Vote codes being entered. Fig 4 shows Auth Code being entered. Fig 5 shows the verification back to the voter along with the Auth and Ack Code-EB3C15

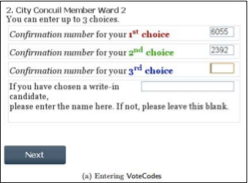


Fig. 3 Vote Codes entered – 6055 & 2392

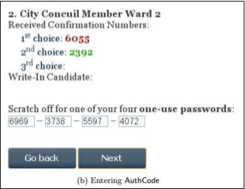


Fig. 4 Auth Code entered - 6969-3738-5597-4072



Fig. 5 User shown his/her choice along with Auth & Ack Code on a public Bulletin Board

Results: Once the final tally is done the results are displayed on a publicly accessible bulletin board. For a particular voter, this will show his candidate’s ‘Vote Code’, the ‘Vote Serial’ of the ballot paper, the ‘Auth Code’ used, the ‘Lock Code’ used and the ‘Ack code’ of the authorization card. Fig 6 shows the final vote as seen by the user post the completion of the tally. Here the voter is shown the Lock code as well as the Vote Serial along with other codes

**2. Security**

2.1 Security Overview

Remotegrity utilizes the “code-voting” approach and features found in the Scantegrity voting system. It thereby seeks to protect voter privacy, provides resiliency against any malware-induced software modifications, election official or intruder manipulations of vote tallies, and other potential injuries to election integrity. It uses distributed key generators and pseudo random generators are used to build a (threshold) shared secret amongst the election officials, which is then used to generate the codes.

2.2 Malware Resiliency

In the voting process the values entered by the voter are the Vote, ‘Auth & Lock codes’. The values, which are never entered by the voter, are Vote Serial and the ‘Ack Code’ (which are present on the ballot and authorization card respectively, but only displayed electronically once the vote is casted).

Considering a case where the malware makes changes to the entered Vote code during ballot casting, this will be rejected by the election official while the initial verification is performed by the system, since this vote code will not generate a relation with the corresponding relation to the Vote serial for this particular voter. Also, since the vote codes are generated randomly there is low probability that the malware will be able to guess a correct alternative code for a particular candidate.



Fig. 6 Final vote confirming the Lock code & Vote Serial (2-456922)

Considering the malware is able to guess a valid code for a candidate, it should also know a valid Auth & Lock Code for confirming the choice. Since the Auth & Lock Codes are generated randomly and sufficient in length, a malware with high probability will not be able to guess the correct alternative Auth & Lock Code (which would be still under a scratch off). Any chances of a malware changing the Auth & Lock Code will also be rejected since this code will not generate the relation with the Ack Code for that particular authorization card for the voter.

2.3 Malfeasance by Election Official and Dispute Resolution

The use of Auth codes and the Lock codes under a scratch off makes sure that there is no malicious activity by an election authority. As we mentioned earlier an election system/computer has to sign every verified entry before posting it on bulletin board, this makes sure that the election official is responsible for the results shown to the voter. Another feature of Remotegrity requires the use of a new Auth code every time a signed entry is displayed, thus if an election official alters the values on the bulletin board after the voter submits the vote means that they (election authority or computer) would require a new Auth code. Similarly, if they attempted to lock-in a vote that the voter did not approve, they would need a Lock code which was not scratched off by the voter. This makes it easier to detect any alterations made by an election official and point out the discrepancy. These properties of the system also allow in maintaining vote integrity and helps in resolving any disputes.

**3. Voter Privacy**

Remotegrity utilizes code voting. This is a feature in which the identities of the voter and the candidates are replaced by cryptographically generated codes and the relation/binding between them is retained for verification purpose. In Remotegrity, the candidates are replaced by ‘Vote Code’ present on the printed ballot and the validity of the voter is verified by the use of the one-time password known as the ‘Auth Code’.

Every voter gets a generated ‘Vote Serial’ and ‘Ack code,’ which allows the voter to verify that the choice of the candidate entered by him on the voting system was recorded correctly. Since the final results are displayed publicly as codes entered by the voter, only the voter knows his vote/choice and does not make any sense to anyone else. This feature of having a publicly verifiable election of Remotegrity and the back end of Scantegrity makes this system universally verifiable.

Details on ***Auditability, Trust structure*** and other aspects can be found in the System detail table. **4. Infrastructure required**

By the election official –

1. Officials who will generate the master secrets for generating the codes
2. Hardware for printing with the capability of having the Auth and Lock codes under a scratch off  

3. A web server infrastructure in case standalone hosting is done or a cloud infrastructure can be used

By the voter – 1. A computer or a smart phone with an Internet connection.

**5. Shortcomings**

We have seen so far that the Remotegrity voting system which was used in Takoma Park is an end to end verifiable absentee voting system which is capable of ensuring the integrity of the votes, results, privacy of the voters, resiliency to changes made by malware and an election official, dispute resolution. However, this system is not designed to be resistant towards voter-coercion. Moreover, like any other web facing infrastructure this system by itself is not capable of coping up with any kind of denial of service attack. This risk can however be reduced if the system is deployed on a cloud infrastructure which provides very high percentage of up time. Also, in case of a denial of service attack, if voters are not able to access the internet; this system can fall back to a regular mail-absentee ballot system. We also did not have any data on the usability aspects of this voting system.



**RIES - Netherlands**

***Introduction***

RIES, the Rijinland Internet Election System was developed by the Hoogheemraadschap van Riginland, one of Netherland’s regional water management authorities. RIES is patented by Piet Maclaine Point and Riginland Water Board [RN.1]. The basic design idea is derived from the master thesis [RN.6] of Maclaine Point’s student, Herman Robers. It was first introduced as a solution to the low turnout rate of Water Board Elections in 2004. About 72,000 online votes were cast, out of 2.2 million eligible voters.

In November 2006, RIES was used by Dutch voters reside outside of Netherland, to participate in the Lower House Parliamentary Elections. Around 20,000 voters voted via Internet, which accounts for 91% of the total eligible voters [RN.4]. The source code of RIES was published on June 2008 [RN.2].

One of the main distinguishing features of RIES is that it enables voter to verify --after the election is closed --that their own votes have been counted correctly, and that the result of the tally corresponds to the cast votes.

Nedap ES3B, the most widely adopted voting system in Netherland, was subjected to some major hacks in fall 2006 that resulted in broad news coverage and discrediting of internet voting. The hacking of Netpad led to the failure of full adoption of RIES for the 2008 parliament election. In addition, a study of RIES’ published source code in 2008 [RN.2] reveals serious security holes that make the system vulnerable to Cross-Site Scripting, SQL injection and predictable token generation. Arguably, the system accords insider election administrator too much power and also does not have any protections from voter coercion in a family situation.

**1. Core Architecture & Operation**

1.1. Basic Architecture &Design

The main voting system is written in Java and Javascript. Cryptographic mechanisms such as DES, 3DES, DESmac, MDC-2, RSA and SHA-1 are deployed through the voting cycle. The discussion in this report is based on RIES 2008. Due to the fact that the majority of the resource on RIES is written in Dutch, the report is only using the available English documents for analysis.

1.2. Voting Process:

In general, the voting process consists of the following steps:

* Before the voting, the administrative agency will use crypto-hardware to generate a personal key for each voter. These keys are printed on ballots and distributed by mail. Furthermore, it will generate ballot collections [RN1] for each voter, combine all the ballot collections to a pre-election reference table and then publish the table on the Internet.
* During the voting, voter will use the RIES web interface to enter its personal key from the ballots, and then select the candidate. When successful, the browser will return a technical vote on screen which serves as a receipt. Furthermore, voter should destroy his ballot with secret key and store the technical vote for future verification. All the technical votes are stored on the network server SURFnet.
* After the election is closed, SURFnet will hand over all technical votes to the administrative agency. The administrative agency computes a hash of every technical vote, validate it using the pre-election reference table and then compute the voting outcome. Finally, the voting office will publish the total outcome.

**2. Security & Trust**

2.1. Security Overview

As further illustrated in system detail tables, RIES were subject to several security related testing before deployment and evaluation. According to the official website, those testing results were positive. Without access to those reports, however, these analysts cannot judge the adequacy of the testing.

In terms of trust structure, a significant amount of trust is placed on election administrators. For voters vote via Internet, although verifiability is achieved through the voting cycle, they still need to trust the administrative party/vendor to handle their personal secret key securely.

For voters who vote via postal ballots, they need to trust the Postal Votes Processing Bureau to convert their mail votes to technical votes correctly because they can’t validate what has been done to their votes. This problem, combining with the large amount of hacking targeting RIES, lead to the failure of full adoption of RIES for the 2008 parliament election.

In addition, a study of RIES’ published source code in 2008 [RN.2] reveals serious security holes that make the system vulnerable to Cross-Site Scripting, SQL injection and predictable token generation. Arguably, the system accords insider election administrators too much power and it also does not have any protections from voter coercion in a family situation.

2.2. Malware Resiliency

This research did not find enough information to conclude whether or not RIES is resistant to DDoS and Malware attacks at Server side. Since the RIES system assumes that voter’s PC is secure (often a flawed assumption), we can conclude that it is not resistant to Malware or ‘Man-in-the-middle-attack’ at Client side.

2.3. Malfeasance by Election Official and Dispute Resolution  When some disputes arise, an umpire can check and recalculate various steps in the whole process and pass his/her own judgment. But this only works for a limited type of disputes. [RN.3]

**3. Vote Privacy**

Voter privacy has been a significant area of concern for RIES voting system. First, vote secrecy is highly depending on the way the personal keys are handled. Although the administrative party/vendor that generate the personal keys are required to destroy these keys post-election, threats like insider activities or malware attacks on the server jeopardize vote secrecy. In addition, the vote server can link the originating IP address to the vote that is cast. The lack of anonymous channel creates another risk for voter privacy. [RN.3]

**4. Auditability.**

Each individual voter can verify, with his stored technical vote, whether his actual vote has been correctly casted. This is achieved by comparing the hash value on his technical vote with the hash value in the pre- election reference table.

The tally verification can be done by anyone interested. Theoretically, interested party can download all technical votes from the network server provider SURFnet, compute the hash value for each vote, and then compare the results with the pot-election reference table. However, this does not enable anyone to verify that the votes are truly as intended.

**5. Testing and Deployments**

According to the official website www.openries.nl, a number of independent organizations have evaluated the RIES voting system before its deployment:

“Various prominent institutions have tested and positively evaluated RIES: TNO Human Factos from Soesterberg tested usability of the voting interface; A team of specialists from Peter Landrocks Cryptomathic (in Aarhus, Denmark) tested the cryptographic principles; Madison Gurka from Eindhoven tested the server and network setup and security; A team under supervision of Bart Jacobs (Radboud University Nijmegen) did external penetration tests.” (Originally written in Dutch, translation cited from [RN.2])

It appears that scientists as well as independent third parties have looked into various aspects of the design and security of RIES, both before and after the deployment. However, most of the testing reports and scientific works are published within Netherland therefore are written in Dutch. This certainly creates an obstacle for the project team to access and interpret those documents.

**6. Usability**

As mentioned in point 5, there were usability and accessibility tests conducted on RIES voting system, but the project team could not find one addressed to the international audience. Based on the available resources, we know that RIES can accommodate both Internet voting and the traditional postal ballots voting. It is accessible to the disabled community in a sense that people can vote at home via Internet using their own accessibility technologies.

**7. Infrastructure**

Available on request in system detail table.

**8. Shortcomings**

Most of the testing reports and scientific works are published within Netherland therefore are written in Dutch. This certainly creates an obstacle for the project team to access and interpret those findings.

Before the 2008 Word Board elections, the ministry hired Fox-IT to perform the formal approval of RIES-2008. The company had found very serious problems with the underlying cryptography.6 [RN.2, p3]

This problem, combining with the large amount of hacking targeting RIES, lead to the failure of full adoption of RIES for the 2008 parliament election. In addition, a study of RIES’ published source code in 2008 [RN.2] reveals serious security holes that make the system vulnerable to Cross-Site Scripting, SQL injection and predictable token generation.

Another testing report [RN.1] on RIES also reveals shortcomings in the following areas: 1) the procedure of voter self-check is quite complicated 2) the two- channel (mail and internet) voting makes system less transparent 3) too much power is given to the election administrator and SURFnet 4) issues with reference table modification due to ballot revoke 5) possibilities of collision hashes 6) voter coercion such as family voting





6 The report is in Dutch. Gedrojc, B., Hueck, M., Hoogstraten, H., Koek, M., Resink, S.: Rapportage Fox-IT - Advisering toelaatbaarheid internetstemvoorziening waterschappen (2008), http://www.verkeerenwaterstaat.nl/Images/20081302%20Bijlage%201%20rapport\_tcm195-228336.pdf

**Helios**

***Introduction***

Helios was developed by Ben Adida. During his time at Harvard he decided to create an open audit voting system that would reside on the Internet. Building upon the ideas of pioneers such as Josh Benaloh, Ben Adida was able to develop an open source platform for an End-to-end Verifiable Voting system.

**1. Core Architecture & Operation**

1.1.Basic Architecture & Design

The Helios system is setup with a website, a python infrastructure, a couple servers, and a built-in encryption system for the votes. The election administrators can from the website, setup an private or public election, invite other users to join the election, run the election, and post the results. The system allows for registration through Google, or Facebook and an alternative login system is in the works. All the non-sensitive data is housed on a server owned by Ben Adida and all the private or sensitive data is held on the voter’s computer.

1.2. Voting Process

In a private election, the administrator inputs the email addresses of the voters who will be participating and the system emails the voters their randomly generated login information and the link to the Internet based election.

Once on the site, the voter follows the on screen prompts and clicks tabs that correspond with their election choices, followed by clicking the next button.

At the end of the prompts is an option to check if their ballot has been encrypted properly and see if their votes have changed. Following this option allows the voter to verify that their vote was accurate but also destroys the ballot and prompts the voter to vote again. On their second try they can skip the encryption check and click finish. This will send an email to the voter saying that their choices have been casted and that the results will be shown at the end of the election. At any time, a voter can follow their old link and vote; the new vote will replace the old vote.





Fig.1 Initial page where the voter can initiate the voting process



Fig. 2 This is the actual voting page where the voter makes his/her selections



Fig. 3 Page that allows the voter to confirm and encrypt the ballot



Fig. 4 This is the encryption page where the system encrypts the ballot.



Fig. 5 Confirmation page that the ballot is ready and the voter is issued their own unique signature



Fig. 6 Ballot audit page, which allows the voter to validate the encryption on their ballot





Fig. 7 Final page where the voter puts in his/her credential to submit the vote.

**2. Security**

2.1. Security Overview

When it comes to security Helios does a decent job but is still plague by various security problems. Orion from the University of Washington highlighted a number of weaknesses in his security review blog. “It is possible, just after the voter casts his/her ballot, for a corrupt router to intercept the ballot en route to the Helios server and send the user a fake Helios server success code, causing the “voting booth” to immediately display a false success message and clear the ballot from memory.”[H.5] I

n this scenario if the user fails to realize that his/her vote has been erased then their vote would end up not counting which would help the adversary manipulate the election. “As it currently exists, if the election administrator allows Helios to administrate the election (as it seems they suggest doing), it is possible for a corrupt Helios server to create new, fake voters and cast ballots on their behalf without easily being discovered.”[H.5]. This would allow for a corrupt server to vote for the desired winner and completely debunk the voting process.

Also, validation is only carried out by the users, so in the off chance that no one audits the election, the corrupt servers could in fact manipulate with being detected. “As currently implemented, the election administrator (who has the power to add voters and freeze the election) is authenticated through Google Accounts. Any vulnerability in the login (weak password, easily guessed security questions, etc.) could allow an attacker to end the election prematurely or add additional voters (potentially multiple accounts for the same voter).”[H.5]

Due to the reliance on the Google account to actually administer the election, if the administrators account was hijacked then an attacker could do a number of things to hack the system in their favor. Helios is also susceptible to every possible Internet attack in their various forms. Helios does not have advanced or cutting edge means to defend itself against any one Internet attack.

This means that any system crashing or infrastructure hacking attacks available can be utilized against Helios. Its’ only real defense are the general defenses seen in most if not all up to date websites. Helios is patched regularly and is mount on a secure platform in Heroku but outside of this, fulfills the bare requirements of being considered secure.

2.2. Trust Structure

“Helios takes an interesting approach: there is only one trustee, the Helios server itself. Privacy is guaranteed only if you trust Helios. Integrity, of course, does not depend on trusting Helios: the election results can be fully audited even if all administrators – in this case the single Helios sever – is corrupt.”[H.1]

Helios was constructed in such a way that the only thing a user would have to trust is themselves and the server. Also, because it has an open audit system, they could always double-check the results to make sure that the server hasn't been compromised. The assumption that comes with these elements is that they have not been hacked and they are in fact safe. In a situation where malware has overtaken either of these items of trust, then the system would be undermined. This isn't a problem unique to Helios, but it is a problem none-the-less.

When it comes to the ballots everything is handled and encrypted by the system itself so at no part of the process does the LEO or Election Official touch or handle the ballots. The only things that they have control over is the keys that decrypt the votes which is only enabled when the election is over. The decrypt keys themselves allow for the votes to be posted, the officials get to see the results the same time the participants see them.

2.3. Dispute Resolution

Dispute resolution is handled by the creator and administrator Ben Adida. He has up to this point only had one dispute to resolve and that was handled quickly and everything was straightened out.

**3. Auditability**

“A web-based open-audit voting system. Using a modern web browser, anyone can set up an election, invite voters to cast a secret ballot, compute a tally, and generate a validity proof for the entire process.

Helios is deliberately simpler than most complete cryptographic voting protocols in order to focus on the central property of public audit-ability.”[H.1]

The entire basis of Helios is the ability to audit the election after the polls are closed. If the election is public, then anyone with Internet access can check the encrypted votes and verify that the elections were legitimate.

The problem with Helios is that though users can audit the system, the process by which one would tends to be complicated to those who are not computer savvy. As mentioned in the usability portion, studies have shown that it is hard for common users to comprehend and properly utilize the auditing tools. This means that the auditing capability is there but the barriers of use are too high for the users.

***Remotegrity***

[R1] Filip Zagorski et al. Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System - https://eprint.iacr.org/2013/214.pdf

[R2] A. Essex, J. Clark, U. Hengartner, and C. Adams. Eperio: Mitigating technical complexity in cryptographic election verification - https://eprint.iacr.org/2012/178.pdf

[R3] David Chaum et al. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy, - https://www.usenix.org/legacy/events/sec10/tech/full\_papers/Carback.pdf

[R4] Remotegrity FAQ -http://www.scantegrity.org/wiki/index.php/Remotegrity\_Frequently\_Asked\_Questions#What\_can\_go\_wr ong\_with\_Remotegrity.2C\_and\_how\_will\_you\_protect\_against\_it.3F

[R5] Cryptographic Voting Debuts, MITnews, http://web.mit.edu/newsoffice/2009/rivest-voting.html

[R6] Remotegrity Poster - http://zagorski.im.pwr.wroc.pl/papers/Remotegrity-poster.pdf

[R7] Takoma Park Public Bulleting Board - http://takoma.remotegrity.org/BulletinBoardFinal.php

[R8] B. Adida. Helios: web-based open-audit voting. In USENIX Security Symposium-http://static.usenix.org/event/sec08/tech/full\_papers/adida/adida.pdf

***RIES***

[RN.1] Engelbert Hubbers, Bart Jacobs and Wolter Pieters.RIES - Internet Voting in Action. Security of Systems, Nijmegen Institute for Computing and Information Sciences. Radboud University Nijmegen

[RN.2] Rop Gonggrijp, Willem-Jan Hengeveld, Eelco Hotting, Sebastian Schmidt, and Frederik Weidemann. RIES - Rijnland Internet Election System:A Cursory Study of Published Source Code. E- Voting and Identity. Second International Conference. VOTE-ID 2009. Luxembourg. 7-8 Sep. 2009, pp 157-171

[RN.3] Engelbert Hubbers, Bart Jacobs Berry Schoenmakers Henk van Tilborg Benne de Weger. Description and Analysis of the RIES Internet Voting System. Institute for the Protection of Systems and Information (EiPSI). Faculty of Mathematics and Computer Science Eindhoven University of Technology. version 1.0. 24 Jun. 2008 [RN.4] Query results from Competence Center for Electronic Voting and Participation, http://db.evoting.cc/index.php?page=database&sub=query\_quick, last accessed April 9 2014

[RN.5] Hugo Jonker, Melanie Volkame. Compliance of RIES to the Proposed e-Voting Protection Profile, E-Voting and Identity-First International Conference. VOTE-ID 2007. Bochum. Germany. 4-5

Oct. 2007. pp 50-61

[RN.6] Herman Robers.Electronic elections employing DES smartcards. Master's thesis. Delft University of Technology. Dec.1998.

[RN.7] Pont Piet Maclaine .Rijnland Internet Election System (RIES) facts and features sheet. Workshop on UOCAVA Remote Voting Systems – Position Papers. 2010. version 2.0. 28-Jul. 2010

***Helios***

[H.1] Adida, Ben. "Helios: Web-based Open-Audit Voting." USENIX Security (2008): n. pag. Web- http://static.usenix.org/event/sec08/tech/full\_papers/adida/adida.pdf

[H.2] Adida, Ben. "Helios: A Deeper Look." Telephone interview. Transcript.

[H.3] Karayumak, Fatih, Maina M. Olembo, Michaela Kauer, and Melanie Volkamer. "Usability Analysis of Helios  - An  Open  Source  Verifiable  Remote  Electronic  Voting  System."  CASED:TechnischeUniversität  Darmstadt  (n.d.):  n.  pag.  Web.   <http://www.usenix.org/event/evtwote11/tech/final\_files/Karayumak7-27-11.pdf>.

[H.4] Weber, Janna-Lynn, and Urs Hengartner. "Usability Study of the Open Audit Voting System Helios." JannaWeber.com. Janna-Lynn Weber, Sept. 2009. Web. <http://www.jannaweber.com/wp- content/uploads/2009/09/858Helios.pdf>.

[H.5] University of Washington, Orion. "Security Review: Helios Online Voting." UW Computer Security Research and Course Blog. University of Washington, 13 Mar. 2009. Web. <https://cubist.cs.washington.edu/Security/2009/03/13/security-review-helios-online-voting/>.

[H.6] SecVote, Dagsthul. "Usable Verifiable Remote Electronic Voting Case Study HELIOS." SecVote. SecVote, 18 July 2012. Web. <http://secvote2010.uni.lu/slides/mvolkamer-usability.pdf>.