City of Toronto RFP #3405-13-3197

Internet Voting for Persons with Disabilities - Security Assessment of Vendor Proposals

FINAL REPORT

Jeremy Clark (*Concordia University*)
Aleksander Essex (*Western University*)

February 14th, 2014

1. Introduction/Summary

From a security design perspective, internet voting is a particularly challenging problem and carries the greatest number of risks of any ballot casting method. Online voting introduces a number of unique potential threats to the voting process: voters must submit secret ballots using a computing device potentially infected with malware or spyware, over a hostile network, for storage on an internet-facing server.

From the perspective of an individual or group who — for any number of reasons — may seek to disrupt, surveil, or otherwise alter the outcome of an election, an online election can be attacked without requiring a physical presence, and can originate from any part of the world, opening the door to the potential of interference from a much wider array of adversaries. Moreover, an internet voting system constitutes a single point of failure. The exposure risk to an attacker, therefore, is considerably lower than in a conventional election.

As a result of these many risks, any internet voting system adopted by the City must be rigorously evaluated. In our opinion, an internet voting system meeting the requirements in RFP 3405-13-3197 would provide the City with a reasonable set of security assurances.

Our Recommendations

Recommendation regarding the use of internet voting: Of the proposals evaluated in the context of the RFP process, it is our opinion that no proposal provides adequate protection against the risks inherent in internet voting. It is our recommendation, therefore, that the City *not* proceed with internet voting in the upcoming municipal election. If the City, contrary to this recommendation, remains committed to the use of internet voting, we advise that the system be limited to voters with disabilities, and not offered to the electorate at large.

Recommendation regarding vendor proposals: From our participation in the City's evaluation, it is our finding that Scytl's solution offers a number of desirable security properties not offered by the competing systems. If the City proceeds with internet voting in the upcoming election, Scytl, in our view, represents the best option of those we observed and evaluated.

Organization

The rest of this report is organized as follows. In Section 2, we provide a high-level summary per vendor of our impressions of their demonstration and proposed solution. In the following sections we enumerate a number of important aspects of internet voting security, outlining particular observations or concerns we had about each respective vendor.

2. Individual Vendor Summaries

Of the three systems we observed and evaluated, we conclude that, on balance, Scytl's system offers the strongest set of security properties. We did not reach a consensus on which system we feel is stronger between Dominion and Everyone Counts, but we do agree that both have significant deficits relative to Scytl's. We reiterate that while Scytl is the best option should the city remain committed to moving forward with internet voting, our ultimate conclusion from evaluating all the three systems is that the City should not adopt internet voting in its upcoming 2014 municipal election.

We compare the vendors in the chronological order in which they presented: Scytl, Dominion, and Everyone Counts.

Scytl

The main advantage of Scytl's voting system is the use of client-side encryption for the voter's ballot selection. This encryption, in principle, enables the voter to meaningfully verify their ballot was collected as cast through a receipt, and offers protection of the ballot right up to a mixing stage. Scytl's HTTPS configuration requires improvement, and no specific encryption algorithms nor key lengths were documented.

Dominion

While the Dominion voting system makes extensive use of encryption, it leaves multiple gaps during the vote collection and tallying phases where the ballots are exposed to insider, and potentially outsider disclosure and undetectable modification. While their configuration of HTTPS is the best of the three vendors, the method they use to encode voter selections appears to undermine the confidentiality provided by HTTPS.

Everyone Counts

Like the Dominion voting system, Everyone Count's system also leaves multiple gaps during the vote collection and tallying phases where the ballots are exposed to insider, and potentially outsider, disclosure and undetectable modification. We are the least confident about their storage of voter PINs, as their documentation claims they are hashed with the weak MD5 algorithm and gives no indication of iterative hashing. They also provide the least security for email delivery of PINs to the voters. We also note they attended their demo day unprepared to answer questions security related statements made in their RFP.

3. Human Factors Security

Human factors security considers with how voters can be exploited into (knowingly or unknowingly) exposing their login credentials or ballot preferences. For their part, each of the vendors acknowledge that mitigating threats of this kind are external to their threat model. Nevertheless, understanding and accounting for these threats must be a matter of consideration for this City if its proceeds with internet voting.

Vote selling and coercion

Ballot secrecy is a mission critical property of most public elections. To that end, conventional (in-person) elections have strict procedures to ensure voters are not able to, voluntarily or otherwise, disclose their voting intentions to another person. Such measures include elections officials only issuing a single paper ballot at a time, signing each paper ballot, directing the voter how to fold the ballot after marked, forbidding cameras in the polling place, and so on.

Clearly, in a remote voting scenario, none of these measures are available. It becomes possible, therefore, for voters to disclose their PINs to others for profit (vote selling). Similarly, it becomes possible in this setting for another person, e.g., a friend or family member, to observe a voter making ballot selections. This opens the door to undue influence, which importantly, can cause the voter to modify their voting behaviour, possibly even without consciously realizing it.

Phishing

Phishing is the practice of sending unsolicited emails in which the sender impersonates an authority figure. Emails of this kind attempt to misdirect recipients into unwittingly revealing their login credentials. A typical target is online banking passwords.

A basic phishing attack in an online voting scenario would be as follows: voters receive an email reminding them to vote and are provided with a helpful link to the “election website.” The link would direct the voter instead to a phishing website made up to look identical to real website, which would proceed to harvest the voter’s login credentials.

Preventing phishing attacks can be difficult: URL spoofing in an HTML-enabled email client is trivial. Spoofing the voting website’s URL in the voter’s web browser is more challenging, but can be accomplished by an IP spoofing attack, or by registering a URL that bears superficial similarity, e.g., `scyt1.com`, `scytl.loginsite.com`, etc. A more sophisticated phishing site could employ a (valid but otherwise irrelevant) SSL certificate to make the “lock” icon appear.

Mitigating phishing attacks would require voters to know—and be looking for—the *correct* election website URL to appear in their browser. As a recent example illustrating how this may be challenging for election officials to ensure, consider the 2011 Canadian federal election voter suppression (i.e., the Robocall) scandal in which voters received phone calls directing them to bogus polling locations. Note that, unlike a voter suppression attack that prevents ballots being cast for a target candidate, a successful phishing attack would be able to casts votes *for* the opposing candidate, having in essence *double* the impact on skewing the outcome.

In particular to this election, we would suggest the City examine how/if screen readers report URLs and SSL connection information to voters.

Other social engineering attacks

Other social engineering attacks can be employed to trick voters into undertaking actions that could install malware or spyware on their computers. Examples include sending users compromised media (e.g., important PDFs, cat/baby pictures, etc), or otherwise directing users to websites able to exploit unknown or unpatched vulnerabilities to deliver malicious payloads.

4. Communication Channel Security (HTTPS)

An important component of any internet voting system is the confidentiality and integrity of communications between the voter and the voting server, which occurs over the public internet. All three vendors use the well-established HTTPS (HTTP over SSL/TLS) protocol to provide security in this regard. HTTPS, however, is a protocol with many subtle security properties, and careful consideration must be given to how it is configured and used.

Extent of HTTPS reliance

For all three vendors, HTTPS is a mission-critical component. A failure in HTTPS would result in voter selections being observable on the internet and even subject to undetectable modification.

For Dominion and Everyone Counts, ballots selections are transmitted over HTTPS with no further protection. In the case of Scytl, the voter client encrypts the ballot before submitting it over HTTPS to the voting server. Ostensibly, this makes the additional encryption provided by HTTPS redundant (but security-in-depth is typically advisable).
Despite this extra layer of encryption, however, Scytl's architecture is essentially as reliant on HTTPS as the other vendors: The Javascript code for performing ballot encryption is itself is transmitted to the voter over HTTPS, and modifications to it (such as removing its encryption functionality entirely) would circumvent its added protection.

In addition to the submission of ballots, all three vendors rely on HTTPS to protect the voter PINs/passwords during login (as well as during the initial transmission of credentials to the voters in the case they are not sent through the postal system).

Server configuration

Because each vendor's solution relies on HTTPS as a core security mechanism, a properly configured server is vital. To assist in evaluating server configurations, we used the Qualys SSL Labs server test tool on the day of each respective vendor's demonstration. Links are provided for reference only (the server configuration may change as demos are removed, as has already happened at the time of writing this report).

Best practices for HTTPS include supporting the newest version, TLS 1.2, of the protocol which resolves certain vulnerabilities in earlier versions (such as 1.0), prioritizing the use of at least 128-bit encryption, and disallowing anything less than 128-bits. Servers should mitigate certain attacks that emerge with the use of TLS compression (e.g., the CRIME attack), TLS renegotiation, CBC-mode block ciphers in 1.0 and earlier (e.g., the BEAST attack), and the RC4 stream cipher (we cannot rule out the applicability of these attacks, although we note they do require specific assumptions that may not apply to specific voting systems). Finally, it is advisable to use a key exchange algorithm that provides *forward secrecy* (i.e., DHE or ECDHE) which will ensure that a future compromise of the election server's HTTPS key (potentially years after the election) will not reveal how voters voted.

Of the three vendors, Dominion had the best SSL/TLS server configuration:

- **Scytl's grade: B** (source: SSL Labs Grade for demo.scytl.com). Scytl's server gets a mediocre score for its TLS configuration. Issues include: not supporting TLS 1.1 or 1.2; allowing the use of weak key lengths (via 56-bit DES); and for not explicitly preferring stronger connections over DES. Scytl's server also supports TLS compression, a known vulnerability. While the server supports cipher modes that provide forward secrecy, it does not explicitly prefer them. Scytl's use of client-side encryption may moot this point if the client-side encryption uses a semantically secure (randomized) encryption algorithm (though we were not able to confirm it).

- **Dominion's grade: A** (source: SSL Labs Grade for intvoting.com). Dominion's server gets a strong score for its TLS configuration. It supports TLS 1.2 and only offers encryption with at least 128 bits. It is not perfect as it prefers RC4, which mitigates BEAST, but has its own vulnerabilities (see the Traffic Analysis section below), and it does not offer forward secrecy at all.

- **Everyone Counts' grade: B** (source: SSL Labs Grade for elect.everyonecounts.com). Everyone Counts' server gets a mediocre score for its TLS configuration. It support TLS 1.2, prefers to use strong 256-bit encryption, but does support weak (56 bit) DES encryption. It does not explicitly mitigate BEAST server-side. It offers and prefers forward secrecy.

We note again that the grading scheme is not our own, and it is designed for general purpose HTTPS configurations, however we believe it is a useful benchmark to establishing the degree to which the vendors comply with best practices.

Traffic Analysis

It is important to understand the limitations of HTTPS. As one relevant example, HTTPS does not hide the length of the message being communicated, as measured in bytes (for RC4) or blocks (for block ciphers). By preferring to use RC4, the traffic leaks finer grained length information than a block cipher would.

Dominion transmits voter selections by candidate name in a JSON object, e.g., `"ChoiceName": "Meghan Agosta"`. Because Dominion's election server prefers the use of RC4, when a voter makes a selection, the length of the corresponding encrypted value (which is transmitted over the public internet) will be proportional to the length of the chosen candidate's name. This may result in an eavesdropper being able to **tell how a voter has voted** based **only on the length of encrypted packets** sent to the election server.

Potential errors in voter understanding of HTTPS

HTTPS is often characterized as a *secure tunnel*. The tunnel originates at the voter's computer. With correct configuration, messages in the tunnel are confidential and cannot be modified. The remaining issue is to establish where the tunnel ends. If it does not end at the voting server, all the protections of the tunnel are thwarted. Establishing where the tunnel ends is accomplished with a *digital certificate* of the server's identity.

In Section 3, we discussed several ways human factors intersect with security, and here we discuss ones specific to HTTPS. Studies reveal that, unfortunately, the average computer users do not possess the expertise needed to correctly use certificates to identify whether their ballots are being transmitted security to the vote server, especially when the protocol is being explicitly attacked.

An attacker may intercept HTTPS traffic through a variety of mechanisms that will cause the user to see a warning (e.g., using an untrusted certificate authority, mismatched domain, etc), however the user can choose to proceed anyway by "clicking through" the warning. Many users are accustomed to dismissing errors/warnings without fully understanding the consequences of doing so, and for a voting system, this could open the voter to disclosing their ballot preferences to the attacker and allowing the attacker to modify them in way they cannot detect.

An attacker may also downgrade the HTTPS connection to an HTTP connection if the voter does not explicitly initiate an HTTPS connection (i.e., types "demo.scytl.com" instead of "https://demo.scytl.com"). It then becomes incumbent on the voter to recognize that the lock icon associated with HTTPS is not visible (the attacker could also place a lock icon somewhere on the page to confuse the voter) and they should not submit their password, PIN, or vote in such a scenario. For average computer users, this may be asking too much.

Certain technical measures, like HSTS headers and certificate pins in browsers like Google Chrome, can help prevent the HTTP downgrade attack, but currently no vendor employs them.

5. Voting Client Security

The use of internet voting places an undue burden on the voters for maintaining the security of their computers. Many common tasks that users do can lead to infection. As just a few examples, malicious software can be installed surreptitiously by visiting a malicious website, opening a malicious PDF document, or downloading a malicious email attachment.

Once installed, malware can interfere with the voting process by stealing the voter's PIN/Password, logging the voter's selections, and/or modifying the voter's selections in a way that cannot be detected by the voter. Malware need not be custom-designed for the election. Many infected computers provide remote access, and a black market exists for renting access to these infected computers. An adversary could specifically purchase access to infected computers located in specific geographic location (as determined by the IP address of the computer).

A wide range of technological capabilities must be considered. On one end, an individual with moderate skill could employ freely available, pre-packaged hacking tools (e.g., the Kali Linux exploitation framework) to conduct a variety attacks (e.g., sending out phishing emails, cloning websites, certain classes of DNS spoofing, malware delivery, credential harvesting, etc). On the other end of the capabilities spectrum there are state-level actors. Although their various capabilities are not widely known, recent disclosures about the NSA's PRISM mass surveillance program and ANT Catalogue of hardware, software and firmware exploits suggest that it is within the realm of possibility for a state-level actor to have a **devastating effect** on an online election.

Importantly, none of the demonstrated vendor solutions attempt to address the security of the voting client.

6. Election Website Security

Credential issuance and management

In each vendor system, voters must use a PIN as authentication for casting a ballot. These PINs may be distributed via mail, but the vendors offer the option for email delivery.
We note, however, email is not very secure and does not generally offer encryption. Any system that intends to distribute PINs through email must account for this.

As a first step, systems like Scytl and Dominion do not send the PINs directly in the email, but instead send a link to a website the voter can use to retrieve their PIN. Website access is encrypted with HTTPS, and access can also be controlled in other ways (e.g., one-time access, IP geolocation, etc). Sending a link to retrieve a PIN in an email, however, is effectively no different than sending the PIN itself. That is, unless access to the PIN is protected by a second factor not found in the email, such as a password set at registration time. Both Scytl and Dominion offer this.

Limiting voter login attempts

Limiting login attempts is an an important security feature for internet facing login sites to prevent brute-force attempts to access voter accounts. For example, freely available software tools (e.g., Burp Suite Intruder) can conduct automated, parallelized login attempts and are relatively simple to execute.

Despite login attempt lockouts being a relatively important functionality, we were unable to test it for any of the demo websites. Scytl stated 3 was their default login attempt lockout threshold, but that the threshold was set to unlimited for the demo. Dominion stated their attempt lockout threshold was configurable, but was apparently not working during the demo.

PIN lengths

Each vendor claimed PIN lengths were configurable. In order for PIN lengths to be secure, they must be long enough to prevent guessing. Even with login attempt lockout periods and strong PIN hashing, short PINs are still guessable. For example, a 4 numeric digit PIN is discovered, on average, after 500 guesses. Instead of an attacker using an automated tool to issue 500 login attempts to the same account (thus triggering a lock out), the attacker could instead use an automated tool using the *same* password to attempt to log in to 500 different voter accounts (and would succeed in logging in to one of them on average). Although IP based filtering on the election server may mitigate this scenario, we recommend selecting PINs from as large a set of combinations as is feasible within the constraints of the usability requirements.

7. Election Server Security

Single point of failure

Unlike a traditional paper-ballot election, an online election has a single point of failure: the election server. Compromising this server could result in a catastrophic failure including breach of the voter PIN/password database, or modification of the ballot database.

While each of the vendors claim to conduct penetration testing of their servers, it is important to note that no server is completely secure, and breaches involving seemingly security conscious companies occur on a regular basis (see e.g., last month's breaches of Bell's password database, Target's credit card database, etc).

Vulnerable election admin software

Note a breach of the ballot database can be mitigated by a cryptographic end-to-end design. In that regard, Scytl in particular is on the right path. That said, we wish to point out a that Scytl directs voting administrators to download and install vulnerable software. Specifically, Scytl's election backoffice requires administrators to download and install a number of software tools, one of which is the Java Runtime Environment (JRE). Instead of directing administrators to download a current release of JRE from Oracle's website, Scytl's RFP instructs them to download a cached copy bundled in their Pnyx.eVoting Backoffice Software (see Section 6.2.3.2 of Scytl's supporting document).

This version, Java 6 Runtime Environment (Update 43), was known for at least 6 months prior to Scytl's demonstration to be vulnerable to a zero-day exploit that would "allow remote attackers to affect confidentiality, integrity, and availability. Given JRE's history of exploits, we discourage its use in an election setting. If, however, it cannot be avoided, we strongly advise the City to use the most **current** release available from Oracle, and update as available.

Voter PIN/password databases

For each vendor, voter PINs are the primary access control mechanism for voting; a valid PIN corresponds to the ability to cast a ballot. A data breach involving the voter PIN database (even one correctly hashed) should, therefore, be regarded as a catastrophic failure in an online election.

Because the occurrence of a breach may not be immediately detected, PIN/password hashing is a vital security mechanism used to "slow down" an attacker's efforts to crack passwords. Scytl and Dominion both claim to use salted password hashing with key stretching. Key stretching is important to slow down an attackers ability to guess relatively low entropy PINs.

Everyone Counts, by contrast, claimed to use salted MD5 hashing (this was stated in the RFP, although in the demo they claimed to have switched to SHA1), and does not, as far as we were able to clarify, employ key stretching.

Freely available password cracking utilities such as hashcat claim to be able to achieve rates of 2^34 MD5 hashes per second on a single GPU. To put that into context, an adversary attacking an MD5 password hashing scheme that utilized salts (but not key stretching) could crack 1 million 4-digit PINs in *one second*. Note that even utilizing key-stretching to limit guess to 10 per second (as claimed by Dominion), one 4-digit PIN could be cracked in under 2 minutes.

The best practice for managing passwords would be to, instead of maintaining a hashed password database, instead use a keyed password hashing/encryption scheme in the context of a hardware security module on a dedicated authentication server.

Ballot databases

Given the importance of the ballot database within the voter server (see Single point of failure above), it is important to consider how access to the ballots is prevented while the data is at rest, as well as who has access to their encryption.

In the Scytl model, ballots are encrypted client-side before they enter the database with a key controlled by the election officials. Thus the database only stores ciphertexts up until the ballots are mixed. The mixing process exposes the ballots, although the documentation provided by Scytl is not clear on exactly how. It appears to be the case that at least one of the following are true: (1) the ballots are decrypted, (2) the ballots are reordered but not rereandomized to unlink them from the ballots entering the mix, or (3) there is no assurance that the ballots output by the mix are the same set as the ballots collected by the voter server (note, a fully end-to-end design would avoid all of these scenarios).

In both the Dominion and Everyone Counts systems, ballots are transmitted over HTTPS and decrypted immediately by the voter server with the server's key. At this point, the ballots selections are exposed to the entity controlling the server and are vulnerable to undetectable modification. The ballots may also be exposed during mixing and prior to tallying, the documentation is not sufficient to make a determination, however it would be effectively to the same entities.

8. Cryptographic End-to-End Verifiability

In an ideal system, ballots would be encrypted by the voter client under a key shared by distinct members of the election authority, and they would not be decrypted until the final tally is computed. Further, between receipt of the votes and the final tally, the ballots would undergo anonymization so no specific voter's ballot can be traced through to the tally.

With such an end-to-end cryptographic layer, the ballot database in the voting server will only ever included encrypted information and can be made transparent to enable voters to independently verify that their cast ballot reaches the final tally and is unmodified. Such as system is said to be end-to-end verifiable. The strongest feature of end-to-end verifiability is that the voting server cannot modify ballots without detection, providing both voters and the election authority assurances that the voting server was not compromised, did not malfunction, and was not tampered with by insiders.

Use of ballot receipts

The first important component of a verifiable system (whether the verifiability is end-to-end or not) is issuing a receipt that enables the voter to check the proper inclusion of their vote in the voter database. However unless if the voter client encrypts, or otherwise obfuscates their ballot selections, such a receipt is must either (1) reveal the voter's selections or (2) not meaningfully verify the ballot's inclusion (and only be an unverifiable assertion by the voter server to have received the ballot).

In the case of Scytl, a meaningful receipt is provided to the voter. In the case of Dominion and Everyone Counts, a receipt value is issued however it does not enable the voter to independently verify their selections are included and unmodified. The receipt is effectively a confirmation from the server asserting it received the ballot.

Use of ballot mixing

As voters login prior to submitting their ballots, the voting server technically has the capability to link submitted ballots to the unique voter identities used to login. It is thus important that ballots are encrypted by the voter client prior to submission (see Ballot databases above). This is necessary but not sufficient for preserving ballot secrecy. Since ballots must be decrypted at some stage, it is important that no entity can trace any received ballot to the corresponding ballot that is decrypted during tallying.

A cryptographic mixer/shuffler enables all of the following (1) the order of the ballots are shuffled without being fully decrypted, (2) the shuffled ballots have different (unlinkable) ciphertexts than the ballots prior to shuffling, and (3) a mathematical proof is produced that the output ballot set encrypts the same identical information as the input set (just in a different order).

Scytl employs a mixer that achieves (1) and (2) but not (3). The vendor commented, during the demonstration, that they have the capabilities to do (3) but the cost would be prohibitive; as assertion that is contrary to our understanding of how (3) can be accomplished.

Dominion and Everyone Counts both perform a mixing step. We first note that voter selections do exist in an unencrypted state between the termination of the HTTPS connection and their inclusion in the voter database, so mixing at best only protects ballot secrecy from insiders who missed that initial window of opportunity. Neither vendor provides sufficient detail to establish the properties achieved. Neither claim (3) and the exact encryption algorithms they claim to use do not permit them to do more than one of (1) or (2).

Future for End-to-End Verifiability

We are not aware of any vendor that provides a COTS internet voting system with true end-to-end verifiability. The property itself has been extensively studied and written about in the academic literature, has been discussed by NIST in the United States, and is the subject of a recently launched project by the Overseas Vote Foundation, also in the United States. We are optimistic about the role it may play in the future and consider it a necessary component of a trustworthy election.

APPENDIX

Remarks on Vendor Demonstrations

Demo Number 1

Requirement: 3.26, 3.27, 3.39

(a) **Hacking**. We evaluate what is stated by the vendor, but have not been granted permission to independently verify any claims or to attempt to penetrate the system. We do not see any measurable difference between the vendors in this regard.

(b) **Insider Tampering**. In general, we consider three broad categories of insiders. Each server is hosted in some environment, where some insiders will have physical access to the server. Each server is configured with to grant administrative privileges to one or more users who can determine the access control policy. Finally, this administrator can assign permissions to less privileged users. We refer to these three as the host, administrator, and user respectively.

All three vendors protect against administrator and user abuse through access control and careful logging of events. However the host is generally not subject to the access control policies and can tamper with logs.

(c) **Viruses or Malware**. It is our belief that no reliable mechanism can establish if a voter’s computer is unsuitable for voting, due to malware, misconfiguration, etc. There are solutions that mitigate the threat of malware through an approach called code voting (used in Norway by Scytl and Takoma Park, MD by Remotegrity) however no proposal here is based on this or any approach to mitigating malware.

(d) **DoS**. All vendors use a similar approach of virtual hosting and load balancing. We do not believe DoS attacks can ever be eliminated, but this is a sensible approach to managing them.

(e) **Phishing**. This is a serious threat for internet voting. Attacks where an email is sent to voters that is ostensibly a reminder to vote, but links to a site other than the official voting page could conceivably trick many average users. Such a site could collect PINs and cast a ballot with the real voting server, potentially even returning valid receipt to the voter. No vendors describe a mitigation strategy for phishing attacks.

(f) **Undue Influence**. Without the protection of a polling booth, anyone can watch you vote or ask for your EID/PIN to cast a ballot on your behalf. This may arise in the context of vote selling or coercion. While mitigation techniques for this problem exist in the academic literature, they are not yet practical enough for use in real-world elections. No vendor described a mitigation

(g) **Device-Launched Attacks**. Voting servers by definition must accept voter-submitted data, and therefore must insure the data is sanitized prior to using it. A malicious formed file extension allowed a code-injection attack that completely compromised a trial internet election in DC. We do not see any obvious attack with the data being submitted by voters in all three vendor systems, but cannot rule out such an attack.

Scytl

◉ Needs Improvement

Scytl provides some level of protection against a server-side adversary. Because the decryption key is generated on separate, offline computers, and the voter's ballot is encrypted on the client side, a server-side adversary cannot decrypt ballots. Because the receipt is cryptographically linked to the voter's selections, a server-side adversary cannot undetectably modify a voter's selections after the ballot is case up to the mixing stage.

Scytl, however, only partially mitigates the threat of a server-side adversary since (a) Javascript used to encrypt the ballot on the client-side is obtained from the server, and (b) the receipt hash is generated server-side. A simple attack resulting from (a) would be for a server-side adversary to "turn-off" ballot encryption in the Javascript. If the system was based on an open standard, voters could obtain the code from a trusted source or even write it themselves, however it is not the case. An even simpler attack, resulting from (b), would simply be for a malicious server, upon receipt of a genuine ballot, to replace it with a spurious ballot, and immediately issue the voter a (valid) receipt for the spurious ballot. This second attack is partially mitigated by the use of voter-specific digital signatures, unless a server-side adversary was able to attack that component as well (which seems at least possible).

Dominion

◉ Weak

Dominion provides little protection against a server-side adversary. In fact a malicious server could sit at the end of the SSL/TLS tunnel, learn how every voter cast their ballot, and undetectably modify each ballot.

Everyone Counts

◉ Weak

Everyone Counts provides little protection against a server-side adversary. In fact a malicious server could sit at the end of the SSL/TLS tunnel, learn how every voter cast their ballot, and undetectably modify each ballot.

Demo Number 20 - Voter Registration

Requirement: 2.3

Some security concerns with self-registration are: ineligible voters will add themselves to the roster, eligible votes will add themselves multiple times, and there is potentially an issue with a flood of fake entries that will have to be manually processed.

All three vendors purport to only allow registration if the information strictly matches to voter registration lists, and allow three have controls in place to ensure duplicate registrations are disallowed. We see little difference between the vendors from a security perspective on this issue.

Scytl

◉ Satisfactory

Dominion

◉ Satisfactory

Everyone Counts

◉ Satisfactory

Demo Number 21 - CAPTCHAs

Requirement: 2.81

CAPTCHAs are designed to be hard to solve in automated fashion but easy to solve by a human. They protect against automated submission of information or access requests. Specific variants of CAPTCHAs historically have had a short shelf-life, with researchers using techniques from vision and artificial intelligence to solve them. Additionally, CAPTCHAs can be replayed from a site that an attacker is interested in breaking the CAPTCHAs on to other high traffic sites they control, forcing the users of this site to solve the CAPTCHA. Finally, solving CAPTCHAs can be outsourced to third world countries for little expense.

For these reasons, we do not place a high level security on CAPTCHAs, however we also do not object to their use if there are no other drawbacks (e.g., in the area of usability and accessibility).

Note: the CAPTCHA service may potentially be hosted outside of Canada. While this data is not personal or sensitive, the solution to the CAPTCHA may be transmitted outside Canadian borders (see Req 1.14).

Scytl

🟢 Satisfactory

reCAPTCHA with audio and visual interfaces. reCAPTCHA is a widely used CAPTCHA service provided by the original designers of CAPTCHA technology.

Dominion

🟡 Needs Improvement

Appeared to use Infragistics with audio and visual interfaces. They claimed their CAPTCHA scored best on usability tests, but in our opinion would be easier for CAPTCHA crackers to solve than ReCaptcha.

Everyone Counts

🔴 Insufficient Detail

Did not demonstrate but claimed to use a “logic-based” CAPTCHAs. It is not possible to assess the security of this approach.

Demo Number 22 - Use of Strong Passwords (Voters)

Requirement: 3.31

The only system to demonstrate a user-chosen password somewhere in the registration process was Scytl. Passwords were restricted to 8-20 characters with a small-case letter, upper-case letter and a digit required. This policy is better than no policy, however it can be improved. There is no security related reason to have an upper-bound (20) on the number of characters, and sometimes restricting passwords length can be an indication of an insecure password storage design (though we don’t have cause to suspect that here).

Scytl’s password system does not appear to check passwords against a blacklist of common/leaked passwords. For example, we registered with “Password1” which is considered “weak” by Google’s password meter and “blacklisted” by Apple’s. None of the vendors appear to offer a blacklist as a configuration option, nor the use of a robust open source meter such as `zxcvbn`.

Note: in the case of each vendor it was a PIN, not a password, that is used as the credential to authenticate voters at voting time. In the demos each vendor used a (weak) 4- or 5-digit PIN although each vendor noted the PIN length should be longer in a live election.

Scytl

◉ Needs Improvement

Remove length restriction (20 characters) and include a blacklist of top most common passwords (e.g., *Password1*).

Dominion

◉ Insufficient Detail

Not demonstrated.

Everyone Counts

◉ Insufficient Detail

Not demonstrated.

Demo Number 24 - Security of Voter Password/PIN Databases

Requirement: 3.32

Protecting the voters' passwords/PINs is **critical** to internet voting applications. In the event a hacker was able to gain access to the password database, additional protections are necessary. Recovering (cracking) a PIN/password from a stolen password database would in essence allow an attacker to steal votes.

Servers should never store passwords in the clear. Firstly, there is no technical need to do; passwords can be authenticated without cleartext storage. Secondly, and more importantly, password lists, especially as voter credentials, are an extremely high-value target for hackers. Current best practice not only requires hashing passwords with a unique salt (to prevent the use of a precomputed "rainbow" table) but also to make use of iterative "stretching" techniques to slow down that ability of an offline adversary to brute force a stolen password database.The use of memory-intensive functions, instead of a hash, can further mitigate offline attacks.

Note: if the system is configured to allow PIN recovery, PINs will not be stored securely by design and can be exported. This appears to a configurable option in each system and our analysis here assumes PINs are not configured to be stored in plaintext.

Note: Each vendor demonstrated a PIN export feature from the administration interface. This 'feature' would require a separate analysis and clarification from vendors.

Scytl

◉ Satisfactory

In a response to a clarification question, they state passwords are stored using `PBKDF2` which is salted and iterative.

Dominion

◉ Satisfactory

Passwords are stored salted and iteratively hashed. The tool `bcrypt` was specifically stated as being used, and configured to rate-limit password attempts to 10 per second (for some unspecified benchmark system). This demonstrates an awareness of the importance of this issue.

Everyone Counts

◉ Weak

Passwords stored as salted `MD5` hashes per their document, although (Director of Engineering) Roy Grossberg claimed (via speaker phone) `SHA256` is now used. There was no indication after follow-up to suggest they used iterative hashing, or understood why it is important. As such their system would not adequately protect against brute-force attacks of common passwords/PINs.

Demo Number 25 - Communication of PIN to voter

Requirement: 2.9

All three systems offer a number of channels to communicate EIDs and PINs to the user. We focus on the security of each channel and concentrate on the communication of the PIN only (the EID is not considered secret information).

Of course, no communication channel is ever completely secure. Postal mail requires physical access to the letters, and is, therefore, harder to perpetrate large-scale attacks against relative to its electronic counterparts.

Email has few privacy guarantees and can be subject to widespread interception (especially by state level actors). SMS (text messages) are protected only by weak encryption. Since all three vendors offer essentially the same suite of channels, there is not much to distinguish them.

Scytl and Dominion claim to offer a more secure version of email than Everyone Counts. With Everyone Counts, PINs are sent directly in the email. With Scytl and Dominion, users are given links to a controlled website that may require a preconfigured password to retrieve the PIN and/or may be a single use link. This requires a modestly more sophisticated adversary to intercept the PIN.

Scytl

◉ Satisfactory

Options include postal mail, SMS, and email (with a unique link to a secure login for PIN recovery).

Dominion

◉ Satisfactory

Options include postal mail, phone, and email (with a unique link to a secure login for PIN recovery).

Everyone Counts

◉ Needs Improvement

Options include postal mail, phone, SMS, and email (with PIN in the content of the email).

Demo Number 26 - Recovery of a Forgotten PIN

Requirement: 2.24

All three systems propose that during registration, voters provide answers to one or more security questions (e.g., mother's maiden name) to be used as a shared secret between the voter and the election officials. This secret will be repeated to regenerate a PIN.

Security questions are known to be weak against impersonation attacks, when the adversary knows information about the victim, or the answer to the security question can be easily guessed with publicly available data. For example, an attacker once gained access to (then US Vice Presidential candidate) Sarah Palin's Yahoo! webmail account by guessing the answers to her security questions, despite not having any personal information about her.

For this reason, we strongly recommend that, at minimum, voters be able to choose their own security questions. No vendor mentioned this option or demonstrated the ability to do so. We rank each vendor, therefore, under the assumption that this is not currently possible.

Finally we note that the option to *recover* a PIN necessitates the PIN is stored in the clear in some part of the system. It is much more secure to only offer the ability to *reset* a PIN and to further ensure that PINs are only stored in a secure manner (refer to Demos 24 and 40).

Scytl

◉ Needs Improvement

Dominion

◉ Needs Improvement

Everyone Counts

◉ Needs Improvement

Demo Number 27 - Password Retrieval

Requirement: 2.41

In addition to a password, the voter often has other credentials that allow registration, obtaining a PIN, etc. Each vendor proposes using this other credentials, and failing that, to effectively reveal all the information that was used in the registration process to grant them the ability to create the PIN. This process however may be accelerated by use of a call centre. There is little to distinguish the vendors on this point, and we see no immediate issues with any proposed procedures.

Scytl

◉ Satisfactory

Dominion

◉ Satisfactory

Everyone Counts

◉ Satisfactory

Demo Number 33 - VPN and Dual-factor Authentication

Requirement: 1.19

To protect the administrative interface from exploratory hacking and illicitly obtained administrative credentials, the RFC suggests the use of a VPN and/or two-factor authentication.

A password-protected VPN offers little security over using a password-protected site served over SSL/TLS. On the other hand, dual-factor authentication can enhance the security of the site significantly.

Scytl

◉ Needs Improvement
Offers a VPN but no mention of dual-factor authentication.

Dominion

◉ Satisfactory
Offers a VPN and demonstrated the use of YubiKeys, a secure token, as a second factor. Also mentioned that access to the server can be restricted by IP.

Everyone Counts

◉ Needs Improvement
Offers a VPN but no mention of dual-factor authentication.

Demo Number 34 - System Prior to Opening of Election

Requirements: 1.8, 2.15

All vendors allow voter credentials to be tested prior to the election, while the inability to vote until the election opens is enforced by the administrative server. We see no difference between the vendors, and the security impact of this requirement is modest (it is more of a functional requirement).

Scytl

◉ Satisfactory

Dominion

◉ Satisfactory

Everyone Counts

◉ Satisfactory

Demo Number 40 - Voter Contact Centre and Voter Information

Requirement: 2.45

It is very important that both private voter information and voting credentials are not exposed or obtainable by the personnel working in the Voter Contact Centre. In the best case, credentials cannot be recovered (only reset) and so there are no credentials stored in the clear at all.

For the information that must be stored, each vendor's system allows a custom role, in a role-based access control (RBAC) system, to be created for staff at the contact centre. Access to voter details can be restricted at various levels of granularity. In addition, all actions taken by users are logged.

As mentioned in our notes on Demo 26, the ability to recover a PIN implies it is stored in the system somewhere in the clear, making the option of resetting a PIN a better approach.

Scytl

◉ Satisfactory

A passing mention was made of the option of resetting a PIN (not just recovery).

Dominion

◉ Needs Improvement/Clarification

No mention was made of the option to reset a PIN instead of recovery.

Everyone Counts

◉ Satisfactory

An explicit mention was made of the option of resetting a PIN (not just recovery).

Demo Number 43 - Voter Choice: Modification & Submission

Requirement: 2.17

From a privacy perspective it is important to consider both when and how ballot selections are communicate to the server.

In terms of *when*, we are talking about the point in the protocol when the ballot selections are communicated to the server, i.e., are the selections being sent to the server before or after the ballot has been cast?

It terms of *how*, it is important to understand how candidate selections are encoded. An important point to make here is that SSL/TLS encryption does not hide the *length* of the message being communicated (in particular when stream ciphers, such as `RC4`, are used). The length of the encoding of each ballot selection, therefore, should be uniform in order to ensure the length of the SSL/TLS packets do not reveal which candidate was selected.

Scytl

◉ Satisfactory

Ballot is not submitted to the server until it is finalized. Therefore the server does not learn anything about the number of times a voter modified their ballot, or what selections were made prior to modification. Further, since ballots are encrypted client-side, encrypted ballots have uniform length and the server learns nothing about the voters selection upon receiving the ballot over the SSL/TLS connection.

Dominion

◉ Weak

Ballot choices are submitted prior to finalization (during a phase called ballot validation) and are decrypted by the server. Thus the server is capable of learning exactly how the voter is modifying their ballot. Further, ballot choices are encoded into a JSON object by candidate name (e.g., `"ChoiceName": "Meghan Agosta"`). It may be possible then to distinguish ballots case for candidates with long names from candidates with short ones---*even under SSL/TLS encryption*. The ability to distinguish message lengths is even more pronounced when using a stream cipher like `RC4`, which happens to be part of the ciphersuite that is most preferred by their voting server's configuration (`intvoting.com`).

Everyone Counts

◉ Needs Improvement

Ballots selections are submitted prior to finalization, which means the server is capable of learning the modifications made to the ballot. Ballot choices are encoded as integers (e.g., ticket_choice=&q1=1). While the integers are not fixed length and it may be technically possible to distinguish a choice 0-9 from a choice 10-99, though this carries a lower risk of exploitation than encoding explicit candidate names.

Demo Number 46 - Overvote Prevention

Requirement: 2.18

All three systems claim they exclude overvoted ballots from count by marking them as spoiled ballots (under the condition that overvotes are permitted to occur). This was not demonstrated, though we have no particular reason to doubt it.

Scytl

◉ Satisfactory

Dominion

◉ Satisfactory

Everyone Counts

◉ Satisfactory

Demo Number 48 - Authorized Ballots

Requirement: 2.21

Prior to receiving a ballot, the voter must log into the system with their voting credential. Upon a successful login, the voter is either provided a cookie or a signing key. The finalized ballot is either submitted with the cookie or submitted along with a signature on the ballot. The server tracks valid cookies and valid signature verification keys it has issued.

We do not see an effective security difference between the approaches of each vendor. Although Scytl's client architecture digitally signs ballots, since the server supplies the signing key to the voter, submitted ballots could in theory be modified as readily as ballots submitted over a cookie-protected channel (as used by the other vendors).

Scytl

◉ Satisfactory

The voter logs in and is issued a signing key, which is used to sign the final ballot.

Dominion

◉ Satisfactory

The voter logs in and is issued a secure cookie, which persists for the ballot submission process (and afterward: see Demo 51 below).

Everyone Counts

◉ Satisfactory

The voter logs in and is issued a secure cookie, which persists for the ballot submission process and is then erased.

Demo Number 49 - End-to-End Verification

Requirements: 2.23, 2.24, 4.16

Not only does no vendor system provide end-to-end verifiability, no system provides end-to-end integrity—there are points in each system's process where election data is vulnerable to undetectable modification.

A mixing algorithm consists of two or more independent (non colluding) parties each executing the following steps:

- **Shuffle**. The input ciphertexts are randomly shuffled.
- **Rerandomization**. The value of each ciphertext is altered—in a random-looking way—such that the contents (i.e., the plaintext inside of the encryption) is not modified.

A verifiable mix would allow each party to prove they performed the steps correctly, without revealing the linking between inputs and outputs.

Scytl

◉ Needs Improvement

The ballot is encrypted client-side, ensuring it is not immediately vulnerable to modification upon receipt by the voting server. This, however, assumes that the voter received the correct javascript code used for the specialized encryption functionality, which, notably, is supplied by the voting server to the voter's computer over an SSL/TLS connection.

At the end of the election, ballots are shuffled to unlink voters from their ballots. The lack of detail in their description, however, makes it impossible to determine if it truly anonymizes the ballots and/or if it exposes the ballots to potential modification. From their description it sounds like they are performing rerandomization (in the form of layered decryption), although by a single entity who would know the association between inputs and outputs.

During the demo, it was noted that zero-knowledge proofs could be employed to make the mixing step verifiable. They explained the necessary software was developed for use in the Norway election but was not included in their RFP, but could be offered, though at an extraordinary cost. We note the shuffle algorithm for Norway is published in the academic literature, and it does not provide end-to-end verification (it provides a weaker property called "verification by proxy").

Dominion

◉ Weak

A ballot is transmitted to the election server over an encrypted SSL/TLS connection. Between the point at which the server receives the ballot, to the point at which the server encrypts the ballot (however brief, the ballot exists in an unencrypted state. During the tally the ballots are shuffled. This process, however, does not appear to employ rerandomization (AES symmetric block-cipher encryption), meaning anyone observing the inputs and outputs (not just the mixer) would be able to follow the association between voter and decrypted selections.

Everyone Counts

◉ Weak

With the exception that Everyone Counts appears to be using a key encapsulation mechanism (`RSA` and `3DES` as opposed to `AES)` for ballot encryption, the same assessment as Dominion (above) applies here.

Demo Number 50 - Receipts

Requirement: 2.22

The use of a voter receipt is essential to assuring voters that their ballot was received and was included in the final tally unmodified. Receipts should be constructed to provide a *meaningful* confirmation to the voter, while, at the same time not revealing or leaking information about the voter's selections to anyone.

The use of receipts was criticized by one vendor, who found through their experience that the receipt confused the voter. We believe that receipts can be useful if their purpose is adequately explained. In conveying the concept to voters, we have found it helpful to refer to receipts instead as *confirmation codes* since this is a concept many voters will be familiar with from other online transactions (e.g., banking, booking hotel rooms, etc).

Scytl

◉ Satisfactory

Scytl's receipt appears to be a hash of the voter's encrypted ballot and signature, which would ensures the encrypted ballot could not be undetectably modified between the time the vote was cast until at least time time the ballots are shuffled. Note this integrity assurance is lost after ballots are shuffled without a verifiable mixing process. The encryption of the ballots ensures the receipt does not reveal how the voter voted.

Dominion

◉ Weak

Dominion's receipt does not appear to be tied to the ballot at all, and thus seems to be an arbitrarily generated code to indicate the voter was "scratched-off" by the voting server. This interpretation, communicated during the demo, was somewhat contradicted by Dominion's response to a clarifying question, which suggested the receipt value may have some cryptographic meaning. Under either interpretation, however, the receipt value does not provide assurance to voters that their ballot was not modified.

Everyone Counts

◉ Weak

The voter receipt issued by Everyone Counts, as explained during their demo, has some form of cryptographic link to voter data, but were explicit that it does not contain any link to a voter's selections. This means that any modification to the ballot would not change the receipt value, which, in turn, means the receipt cannot be relied upon as a mechanism to detect modifications to cast ballots.

Demo Number 51 - Unlinkability

Requirement: 2.25

We discuss shuffling/mixing algorithms in Demo 49 above.

Scytl

◉ Needs Improvement

In this system, the unlinking of a voter identity from their ballots is done at the mixing stage. As mentioned previously, we do know the full details of the mixing protocol. Based on their description, and in addition to shuffling, it seems some kind of rerandomization step is performed involving layered decryption. The entire process, however, seems to be conducted by a single party, allowing them to track the association between inputs and outputs.

Dominion

◉ Inadequate Information

In this system, the voting server interacts with the voter to receive her credentials followed by her ballot. While these are encrypted with SSL/TLS, the encryption ends at the voting server (who then reencrypts the ballots for storage). This means that a malicious voting server (or anyone capable of decrypted the SSL/TLS session) can learn how every voter voted. A shuffling step is claimed to provide unlinkability between stored votes and the votes going into the final decryption step, however it is not adequately described to assess, and no rereandomization seems to be employed.

Everyone Counts

◉ Inadequate Information

In this system, the voting server interacts with the voter to receive her credentials followed by her ballot. While these are encrypted with SSL/TLS, the encryption ends at the voting server (who then reencrypts the ballots for storage). This means that a malicious voting server (or anyone capable of decrypted the SSL/TLS session) can learn how every voter voted. A shuffling step is claimed to provide unlinkability between stored votes and the votes going into the final decryption step, however it is not adequately described to assess, and no rereandomization seems to be employed.

Demo Number 52 - Disputes over being Struck Off

Requirement: 2.29

In all cases, a voter who is incorrectly struck off must convince an election official that this is the case, and certain election officials will have the necessary privileges to allow the voter to cast a(nother) ballot. The risk here is that a voter may convince the election official, even though they have voted previously. We see no notable difference between the systems with respect to this issue.

Scytl

◉ Satisfactory

Dominion

◉ Satisfactory

Everyone Counts

◉ Satisfactory

Demo Number 53 - Last Vote Counts

Requirement: 2.27

In scenarios where a voting system must accept multiple votes from the same user and only count the last one, the system must have a mechanism to determine which cast ballots belong to which voters (so that it may invalidate any previously cast ballots). This is a difficult property to achieve simultaneously with protecting voter anonymity (see Demo 51 above).

Arguably the only way to do this reliably is in the context of voter-encrypted ballots, where the voter’s identity can safely be attached to the stored ballots and allowed to persist until after the election is closed (since it is encrypted with a key unknown to the voting server).

Scytl

◉ Satisfactory

Given the ballots are encrypted by the clients, the system can maintain a record of which voters submitted which encrypted ballots, without any violation of ballot secrecy. With this record, it can implement a policy that allows multiple votes but keeps only the last one submitted.

Dominion

◉ Weak

Does not allow multiple submissions.

Everyone Counts

◉ Needs Improvement

Outlines a mechanism to allow multiple submissions, however there is not enough detail to ensure this mechanism maintains unlinkability between voters and their cast ballots.

Demo Number 54 - Multiple Sessions

Requirements: 2.30, 2.31

We do not see a security issue with allowing multiple simultaneous sessions, so long as only one ballot is allowed to be cast. This latter property is enabled through a centralized electronic voter list, where voters are struck off upon casting a ballot.

Scytl

◉ Satisfactory

Allows multiple sessions but only a single ballot to be cast.

Dominion

◉ Satisfactory

Allows multiple sessions but only a single ballot to be cast.

Everyone Counts

◉ Satisfactory

Prevents multiple sessions through cookies.

Demo Number 55 - Eliminating Rogue Ballots

Requirement: 3.29

See Demo Number 48 above.

Demo Number 58 - Random Generation of PINs

Requirement: 2.56

The PINs used by voters to vote must be generated independently and randomly so that an adversary cannot guess their value and cast ballots on behalf of voters. As previously noted, random generation of PINs is meaningless if the length of the PIN does not prevent exhaustive search of all PINs. In addition, the login screen should be locked down to prevent repeated attempts at guessing a PIN. This in itself is necessary but not sufficient: a botnet could be employed to rotate through IP address, and it cannot provide protection in the case that the list of hashed PINs is leaked.

Each vendor demonstrated either a 4 or 5 digit numeric PIN. Although PIN length must be weighed against accessibility requirements, we note that short PINs are vulnerable to both online and offline attacks. In the former case, automated login cracker tools can brute-force a 4- or 5-digit PIN, *especially* if no CAPTCHA or login attempt limits are employed (as was the case in the Everyone Counts demo). In the latter case, a 4- or 5-digit PIN in a leaked password database are trivially recovered. In the case of a salted, non-iteratively hashed password database (such as the one described by Everyone Counts) storing 4-digit PINs, we estimate a single computer employing an offline brute-force attack would be able to crack over a million PINs **per second**.

Scytl

◉ Seems Satisfactory but not Verified

Vendor acknowledges that care is taken to generate PINs randomly but provides no details as to how.

Dominion

◉ Seems Satisfactory but not Verified

Vendor acknowledges that care is taken to generate PINs randomly but provides no details as to how (claims an open source tool is used).

Everyone Counts

◉ Seems Satisfactory but not Verified

Vendor acknowledges that care is taken to generate PINs randomly but provides no details as to how.

Demo Number 71 - Password Aging

Requirement: 3.33

We do not see any major difference between the vendors on the provision of password aging, although we do echo one vendor’s comment that in the timeframe of a typical election, aging does seem particularly important.

Scytl

◉ Satisfactory

Aging is configurable.

Dominion

◉ Satisfactory

Aging is configurable (but not on CEVL).

Everyone Counts

◉ Satisfactory

Aging is configurable.

Demo Number 72 - Data at Rest / Cryptographic Standards

Requirement: 3.42 (we include 3.43 as well)

All vendors encrypt data at rest. All claim to use standardized cryptography. Using standardized cryptography is necessary but not sufficient: implementations of cryptography may have their own vulnerabilities. It is thus best to use well-tested cryptographic libraries to ensure no side-channel information is leaked during cryptographic operations.

One of the most important considerations is the strength of the cryptographic primitives that are used. The United States National Institute of Standards and Technology (NIST) currently requires 112-bit equivalent security (this effectively means encryption should be at least 112-bits, hash functions at least 224, and RSA at least 2048). It should be noted, however, that election data may be required to remain confidential longer than other types of data, and we do not have good guidelines for cryptographic strength for long-term security.

Note: properly implemented, standardized cryptographic software may not be sufficient to guarantee confidentiality against a state-level actor (see e.g., the `Dual_EC_DRBG` controversy).

Scytl

◉ Needs Improvement

The RFP lists several cryptographic and security standards complies with and mentions use of the well-tested `OpenSSL` library, but does not provide any detail as to what specific algorithms are used, nor is there any assurance that the key lengths provide adequate security (for example, they do not state compliance with NIST SP 800-131A). The client-side Javascript that performs ballot encryption appears to be obfuscated.

Dominion

◉ Satisfactory

Listed algorithms include `RSA` 2048, `AES` 256, and `SHA` 256 which are acceptable in 2014 (as per NIST `SP 800-131A`).

Everyone Counts

◉ Satisfactory

Listed algorithms include `3DES` (168), `RSA` (2048), `MD5` (128), and `SHA256` (256). With the exception of MD5, all of these are acceptable for use in 2014. MD5 is used for password storage, which may still be satisfactory despite known attacks on MD5's collision resistance. That said, transitioning to SHA256 is recommended.

Demo Number 73 - Lock Out

Requirement: 3.34

Enabling the lock out of a user after many login attacks (perhaps indicative of an online password guessing attack) should be considered carefully. Such a policy could could enable an adversary to lock out a specific voter, denying or frustrating their attempt to cast their ballot. For this reason, we consider it satisfactory to *not* provide this option.

Scytl

◉ Satisfactory

Lock outs are configurable.

Dominion

◉ Satisfactory

Lock outs are not permitted by the Dominion Voting System but are present (and not seemingly configurable) in CEVL.

Everyone Counts

◉ Satisfactory

Lock outs are configurable.

Demo Number 85 - Encrypted Transmission of the Voter List

Requirement: 3.9

All vendors claim the ability to receive an encrypted transmission of the voter list.

Scytl

◉ Needs Improvement

Does not mention Certificate Authorities (CAs) or static IP addresses.

Dominion

◉ Needs Improvement

Mentioned that the file is downloaded via FTPS but then uploaded via HTTPS. It is not clear who the midpoint is in this process. The FTPS stream must be decrypted before being reencrypted for the HTTPS stream, so the plaintext will be temporarily exposed. Since, however, this does not occur while “in transit,” it does not obviously violate the requirement. They mentioned CAs but not static IP addresses.

Everyone Counts

◉ Needs Improvement

Mentions static IP addresses but not CAs.