I would be grateful if the City Clerk's office could include this study in the package which will go out to council for the may 28 council meeting

Knights and Knaves run elections: Internet Voting and Undetectable Election Fraud
https://openprivacy.ca/assets/knightsandknaves.pdf

This paper looks at Switzerland's e-voting situation. It's an interesting approach. Switzerland is seeking to maintain public control of their elections and run them themselves, but they hired a vendor, Scytl, to create the software for them.

What happened next was a huge scandal. And Scytl is now declaring bankruptcy, although itis not clear if the bankruptcy is related to the scandal.

Please let me know today if you are able to include this in the package.
many thanks
Lin Grist

Knights and Knaves Run Elections: Internet Voting and Undetectable Electoral Fraud

Chris Culnane | University of Melbourne
Aleksander Essex | Western University
Sarah Jamie Lewis | Open Privacy Research Society
Olivier Pereira | Université Catholique de Louvain
Vanessa Teague | University of Melbourne

The cryptographic weaknesses recently discovered in SwissPost's e-voting system are described in this article. We explain how they relate to security problems in other Internet voting systems and discuss the necessary principles for building trustworthy elections.

On the island of knights and knaves, everyone looks identical; however, knights always tell the truth, and knaves always lie.[24] When you arrive on the island you ask the first man you see whether he is a knight or a knave. "I'm a knight, of course!" he says. What can you infer?

He may be a knight who tells the truth about being a knight. But we give this puzzle to children so they can see that there's another solution: he might be a knave, lying about his own untrustworthiness.

Digital Object Identifier 10.1109/MSEC.2019.2915398
Date of publication: 9 July 2019

A secure computer that runs the right voting software may look exactly like a compromised computer running malicious or buggy software. If you ask it whether it recorded the vote you asked for, it will say yes. If you ask it to recount the votes, it will perfectly repeat the first count performed. This tells you nothing about whether it was honest in the first place.

The risk of foreign attacks on trusted voting systems and accounts has been widely documented, most recently in the Mueller report.[21] Russian agents

> *targeted individuals and entities involved in the administration of [U.S.] elections. Victims included U.S. state and local entities, such as state boards of elections, secretaries of state, and county governments as well as individuals who worked for those entities. [They] also targeted private technology firms responsible for manufacturing and administering election-related software and hardware …*

But breaking in from the outside isn't the only way to compromise a system. In July 2018, the Maryland State Board of Elections informed citizens that one of their contractors, ByteGrid LLC, was primarily controlled by a Russian oligarch. ByteGrid "hosts the statewide voter registration, candidacy, and election management system, the online voter registration system, online ballot delivery system, and unofficial election night results website," they announced.[28]

So how can we derive evidence of an accurate election outcome when the election is conducted by computer when we cannot determine who controls it or what it is doing?

Yet Another Fundamental Cryptographic Weakness in an E-Voting System

In early 2019, several of the authors (Lewis, Pereira, and Teague) examined the source code for the Swiss-Post e-voting system. The system, provided by Scytl and intended for use in Swiss elections in October 2019, was in the process of certification for use by up to 100% of Swiss voters in the cantons that chose to use it.[29] The source code circulated reasonably freely online, likely

 Copublished by the IEEE Computer and Reliability Societies

because of Swiss Federal Chancellery Ordinance 161.116,[34] which mandates open public comment on the source code.

We discovered three different ways in which a compromised computer could manipulate votes while pretending to provide a proof that no manipulation occurred.[15–17] One was a cryptographic trapdoor, which allowed a cheating authority to provide a perfectly verifying proof that it had shuffled the votes correctly, even if the votes had been manipulated. This would leave no way for anyone to detect the fraud, because the tampered proof would not only pass the prescribed verification process, it would actually be perfectly indistinguishable from a truthful proof.

In light of our findings, and because at least one of them also affected an earlier version of the same voting system (which includes fewer security features), SwissPost decided to not offer this e-voting system for elections in May of 2019.[30] Switzerland's Federal Chancellery approved this decision and announced its intention to review their licensing and certification procedures for e-voting systems.[31]

When our first finding was disclosed, Australia's New South Wales (NSW) Electoral Commission, which had purchased an e-voting system from the same supplier, unexpectedly declared that their system was affected by the same error. At the time, the system was being used for their state election; however, the commission said it would be fixed prior to decryption time two weeks later.[32]

When we identified a second problem in the same components of the Swiss system 12 days later, the NSW Electoral Commission insisted that it did not affect them.[33] Because their code is secret, there is no way that this can be verified.

Although numerous serious security problems have been found in e-voting systems before,[13,23,26,27] this was the first discovery of a cryptographic flaw in a verification mechanism in a system already being used in government elections. This is significant because verification potentially allows a way out of the inscrutability of computers and perhaps a way forward for securing electronic elections.

To explain the importance of this failure, we need to explain what verifiability is, and what it is not, and how to tell whether you can trust an election outcome without trusting the computers, administrators, or vendors.

Evidence-Based Elections

One approach for accomplishing this is to have voters produce paper ballots, enforce a strict chain of custody on them, and then use a public manual count or a risk-limiting audit of the paper ballots to verify the computerized count.[18] Currently, this appears to be the only routinely deployed method of incorporating the advantages of computers into electoral processes without sacrificing integrity. It does, however, rely on carefully securing the paper ballots.

For many years, the cryptography research community has investigated an alternative. End-to-end verifiability[5] provides evidence to voters that their vote is accurately recorded and included; it then provides evidence to everyone that every electronic vote was properly tallied.

The main point is simple: imagine you are on the island of knights and knaves. You cannot tell which people or computers behave in a trustworthy fashion, but you insist on deriving meaningful evidence of an accurate election outcome. An end-to-end verifiable election system should provide an opportunity to detect an incorrect election result, regardless of whether all the computers and the people running the election might be knaves.

The properties of end-to-end verifiability are complementary to those of risk-limiting audits, i.e., both require assurances that only eligible voters participate, and both provide public evidence of an accurate tally; however, end-to-end verifiability also provides each voter with a chance to check that his or her own vote was properly included.

There are many open source, publicly owned academic projects for end-to-end verifiability, some of which are used for online voting while others are intended for e-voting in a polling place.[2,6,9,12,14,22,25] All of these were designed for government elections, and some were deployed. As far as we know, none of these have experienced significant problems, yet none remain in use. Many proprietary systems claim to provide strong verifiability properties, but upon further examination they are generally found to be seriously lacking.

So why does this idea sound ideal but is not in use? Why do deployed systems with some of these verifiability properties fail so miserably in practice? In this article, we explain what end-to-end verifiability is, describe the imitations that should be avoided, and examine how to improve the integrity of election systems worldwide.

Trust No One: What End-To-End Verifiability Would Be If We Had It

End-to-end verifiability is typically achieved in three steps:

1. cast-as-intended verification, which must be performed by the individual voter and allows the voter to check that his or her ballot accurately reflects his or her intention
2. recorded-as-cast verification, which can be outsourced but is usually expected to be performed by the voter and allows for checking that his or her vote was recorded unaltered (usually on a public list of votes)
3. universal verifiability, which allows any member of the public

to check that all of the recorded votes have been legitimately submitted by voters and have been correctly entered into the count.

The SwissPost–Scytl system claimed to achieve "complete verifiability," which is an incomplete alternative to end-to-end verifiability, in which at least some electoral authorities must be trusted not to cheat and cannot be identified if they do.[34]

Criticisms and Limitations of End-To-End Verifiability

End-to-end verifiability is a fault-detection mechanism, so the potential for unintended consequences exists.[10] Voters may claim to have detected faults where none occurred, which is especially damaging if third parties cannot distinguish system misbehavior from spurious accusations by voters who want to cast doubt on an election's outcome.

Verification is required at multiple levels, and a verifiable system may be rendered useless without it. Voters or auditors may not bother to verify; if no one or too few people verify, then we cannot express any confidence in the election result. Furthermore, evidence suggests that even when the system correctly detects a fault, voters may incorrectly attribute it to their own actions and fail to report it.[20]

The system verification process must be carefully verified as well. Protocol errors, implementation vulnerabilities, or hidden trapdoors in end-to-end verification software could be exploited to produce a valid-looking proof of a false election result, similar to that which was demonstrated in the SwissPost system (described previously) or the Helios system.[4,8]

Nevertheless, end-to-end verifiability has the potential to offer a kind of transparency that is more robust than that offered by traditional paper-based systems and absent in many electronic systems. This method has been run on computers used in government elections in a polling place, where it provides a complementary evidence trail that enhances (although, arguably, does not replace) a risk-limiting audit of paper votes.[2] Perhaps a solution to this problem is to combine some of the advantages of end-to-end verifiability with a risk-limiting audit of paper evidence.

In the following section, we explain why the SwissPost–Scytl e-voting system did not achieve its intended verifiability properties.

The SwissPost–Scytl System and Why Its Results Are Not Verifiable

Trapdoor Commitments: When Is a Proof Not a Proof?

In the SwissPost–Scytl system, each voter submits his or her encrypted vote to an election server. These votes are then reencrypted and shuffled by a series of mixers to protect individual voter privacy. Each mixer that shuffles votes is supposed to prove that the set of input votes it received corresponds exactly to the differently encrypted votes it outputs. This is intended to provide an electronic equivalent of shaking a publicly observable ballot box. It must secure both the privacy of each voter's choice and the overall integrity of the votes.

Proofs of shuffle are among the most complex cryptographic protocols used and notoriously difficult to design and implement correctly. In this case, Scytl decided to make use of a proof of shuffle proposed by Bayer and Groth.[3] This proof makes use of various cryptographic primitives and depends on their security. Should any of these primitives fail, then the proof of shuffle loses its security. This is exactly what occurred here; the Bayer–Groth proof of shuffle relied on a cryptographic commitment scheme, which was incorrectly implemented.

A cryptographic commitment is a digital equivalent of putting a written number into an envelope. First, the number cannot be changed once it is in the envelope, i.e., the commitment is binding; and second, nobody can read the number until the envelope is opened, i.e., the commitment is hiding. The commitment scheme is a critical part of the shuffle proof because the mixer commits to the permutation it will use to rearrange the votes, proves that it is a true permutation, and then proves that it has applied it properly. If the binding property of the commitment scheme is broken, the mixer can apply a function other than a permutation to the votes, and hence, add, drop, or change them.

In the SwissPost–Scytl system, the chosen commitment scheme offered a very specific feature—a trapdoor. In their protocol, this trapdoor is computed privately by the mixers when they produce the keys used in the commitment. Should a mixer decide to make use of this trapdoor, it would be able to break the binding property of every commitment. As a result, it becomes possible to manipulate votes while also producing what passes for a valid shuffle proof. This is similar to the electronic equivalent of shaking the ballot box in full view of observers, while somehow managing to substitute ballots.

While we were discussing this issue with SwissPost, two other teams of researchers independently discovered and reported it (Haenni[11] and Haines).

How hard is it to cheat? In this article, we have presented two cheating examples, which are available for testing by anyone with access to the Scytl–SwissPost code (https://people.eng.unimelb.edu.au/vjteague/SwissVote). The first example requires knowing the randomness used to generate the vote ciphertexts that will be manipulated. There are several ways this could be achieved.

For example, an attacker could compromise the clients used for voting. Weak randomness generation would allow the attack to be performed without explicit collusion.

The second cheating example does not require any extra information at all, although it does rely on the election parameters to have been set up in a particular way.

In both cases, a mixer controlled by the voting system operator must cheat. Such a cheat could have several sources: for instance, it may happen because the mixing server was hacked by a third party, because a corrupted server manager is a victim of blackmail, or simply because the operator is willing to cheat to support a specific candidate. Such potential issues are precisely the reason why a verifiable voting system is required in the first place: the trustworthiness of an election result should not depend on the security of a specific server or on the reliability of the voting system operators. And in this case, the claimed authenticity of the system may actually work against its security; because the system is expected to be correct, the operational security of the mix servers may be confirmed by a proof of shuffle passing verification, and further investigations may be overlooked.

The Weak Fiat–Shamir Transform, and the Implications for Decryption and Voter Verification

The second part of proving a proper election outcome—given a set of received votes—is to prove that they have been properly decrypted. Suppose there is an authority (human or machine) who knows the decryption key for all the votes. This authority could prove that it had properly decoded by publishing its private decryption key, which would certainly allow everyone to check that the translation was correct. Unfortunately, it would also allow everyone to decrypt individuals' input votes, thereby determining how particular people voted.

Alternatively, we could simply ask the authority to decrypt and trust it to do so correctly; however, this would call into question the integrity of the process because the authority could produce votes that were different from the true interpretation of the votes it had received.

The SwissPost–Scytl system, like many cryptographic voting systems, instead provided a zero-knowledge proof of correct decryption. It aimed to prove that the votes were correctly deciphered, but unfortunately, it suffered from a known error,[4], i.e., the construction of the zero-knowledge proof allowed a cheating authority to construct an apparently valid decryption proof, which passed verification, but turned a valid input vote into nonsense that could not be counted.[17] This is the electronic equivalent of leaving the ballot box in plain sight all day, but somehow substituting nonsense votes into the poll when it's time to display the votes on the counting table.

Proof failure in deployed systems. Finally, we showed that the same error in the Fiat–Shamir heuristic was also present in the voting step[15] (and in other places as well, with an unknown impact).

The Swiss e-voting system uses a code-return system, i.e., voters receive a paper mailout with random "yes" and "no" codes for each voting option (candidate). When their vote is cast, voters expect to receive the yes code for the candidate they chose and the no code for all the rest. We showed that the weakness in the zero-knowledge proof implementation applied here, too, thus allowing a cheating voting client to send a nonsense vote while ensuring that the voter received exactly the return codes he or she was expecting. In this way, an apparently successful vote verification would hide the submission of an invalid vote.

After submitting this issue, we were informed that this error was present in voting systems that had previously been used in Swiss elections in the belief that the code-return verification mechanism was sound. Although exploiting the problem was detectable in principle, by checking for invalid votes appearing at decryption time, it was not an explicit part of the verification process (formal verification would have passed even if the votes had been changed in this way). Furthermore, although we could not find an undetectable way to exploit this weakness, there is no reason to be confident that no such opportunity exists.

At this point, given news of a serious problem in a system that had already been established, SwissPost decided to put the previously used system in standby mode. It was not used in the Swiss elections of May 2019.

Stepping back: other broken proofs, unused code, and quality. Although the failures of shuffle proof and decryption proof compromised the security of the SwissPost–Scytl system, these failures alone do not fully capture the extent of the issues with it.

We also documented that the source code included the implementation of an OR proof (a proof of logical disjunction) construct that also contained a critical defect (a missing verification step), rendering it insecure. The SwissPost system did not require an OR proof, and conversations with Scytl revealed that it was not the only part of the code that was unused.

Considering that the ostensible purpose of making the code available was to allow third-party auditing, it is concerning that the code as provided was significantly bloated with unnecessary (not to mention broken) constructs. Source code review is inherently a difficult task, especially when the code itself was never designed to be easily audited, where important cryptographic verification is spread across multiple

files or packages, even without the addition of unnecessary (and unused) functionality.

Taken as a whole, it is possible to draw two distinct, but complementary, conclusions from the SwissPost–Scytl system. The first is one of a system so lacking in basic quality controls that a keen eye anywhere would unearth critical bugs. The second is one of skilled researchers making educated guesses regarding where the critical flaws are most likely to be and finding them. Neither conclusion by itself tells the full story, but both combined paint an accurate picture of real-world voting software that contained election-stealing vulnerabilities and was simply not fit for use. And sadly, this story is not unique.

Security and Verification Problems in Other E-Voting Systems

2019 Voter Verification in NSW

Although the shuffle proof used in NSW in 2019 seems to have been very similar to SwissPost's, its cast-as-intended verification mechanism was completely different.

Each voter casts a vote using his or her web browser. At the end of the voting session, the browser would send an encrypted version of the vote the voter entered and then print a QR code on the screen. If the voter didn't trust the software in his or her web browser to cast the vote correctly, he or she could download a closed source app from the same company onto his or her smartphone, hold its camera up to the QR code, and ask the app what vote the browser code had sent.

> *Suppose the company's second piece of software tells you that its first piece of software sent the vote that you asked for. What can you infer?*

We hope it is clear that this verification mechanism does not add any evidence. If the software provider is a knight, then the software sends the right vote the first time; however, if it is a knave, it sends the wrong vote and then lies about what vote it sent. In neither case does the voter receive any information by asking it. Nor does the voter have any way to prove if it did misbehave. Even innocent programming or configuration errors, such as switching the names or positions of two candidates, could be repeated in both programs and cause the verification step to produce what the voter expected regardless of whether the true vote was different.

Alberta: Eligibility Unverifiability

The current governing party in the province of Alberta, Canada, held its leadership vote in 2017 using an online voting system. The election has since become the subject of a criminal investigation into allegations of fraud after it was discovered that some party members were recorded as having voted, despite never having received their login credentials.

New members of the governing party completed a membership application form, which included an email address field. The fraud is alleged to have occurred as follows: at some time between when the membership applications were completed and when the online voting period began, fraudulent email addresses were allegedly substituted into the membership records. When the login PINs were emailed to the new party members during the voting period, they went to the malicious accounts instead.

Media reports described several instances where email addresses were modified or inserted without the voter's knowledge.[1] The domains of several email addresses in question were registered to the same provider in the United States around the time of the election and are linked together by the Subject Alternative Name for the public-key certificate. It was also reported that several members were registered using the same email address as the business of one of the candidates.[19]

Western Australia: Outsourcing Trust

An earlier version of the NSW iVote system ran in Western Australia in 2017.[7] All voter-facing parts of the system were set up behind a TLS proxy. It was not obvious to voters that such a service was being used.

A TLS proxy counters distributed denial-of-service (DDoS) attacks against a server by inserting an authorized man in the middle as a gatekeeper. The TLS proxy can see the decrypted traffic and analyze it for any potential threats. The actual target server does not respond to normal external requests, it accepts communication only from the TLS proxy. However, in the case of the Western Australian election, the voting server in Sydney was visible on the Internet in the normal way, thus completely undermining any DDoS protection until we pointed this out to the authorities. Figure 1 shows the use of a proxy certificate used for elections in Western Australia, where the Western Australian Electoral Commission is one of numerous alternative names that all use the same certificate.

Protection comes at a price: there is a third party that intercepts and inspects voters' communications with the Electoral Commission. The physical analog would be if the Electoral Commission was inundated with junk mail, so it decided to outsource the processing of postal votes to a third-party company by redirecting all of its mail to a warehouse. In that warehouse, the company would open all of the envelopes, decide which ones were genuine, and then forward them on to the Electoral Commission.

At the very least, one would expect scrutineers to be present during the opening to monitor what was being rejected and what was being

sent on; however, in a digital setting, meaningful scrutiny is impossible.

The certificates and key pairs that authenticate connections to the electoral commission are distributed globally. In the case of the Western Australia deployment, there were points of presence serving the certificate in numerous countries, including China, Canada, the United States, and the United Kingdom.

Even if we put aside the risk of a nation-state attack, this use of a TLS proxy presents a number of problems in the context of a voting system.

JavaScript injection. When the TLS proxy first received a connection from a voter, it injected its own obfuscated JavaScript into the response from the Electoral Commission. This JavaScript is normally used to profile the client to assist in DDoS protection. However, the Electoral Commission has no oversight of, or control over, what was contained within that JavaScript. As such, the client effectively ran a modified version of the election system, albeit a version modified hopefully with good intentions. It was deemed possible to construct a malicious, obfuscated JavaScript that extends the profiling functionality to leak the voter's credentials via a cookie, while still maintaining the overall length of the obfuscated JavaScript.[7]

Bridging the separation of roles. The TLS proxy was also used for the registration service, as shown in Figure 2. The iVote system was designed with a separation between the registration server—which inevitably learned the voter's identity—and the voting server, where people voted with a pseudonymized ID they had acquired at registration time.

Unfortunately, the TLS proxy service automatically downloaded persistent cookies to the voter's device at registration time. Thus, voters accessed the registration and voting service from the same browser without clearing their cookies, the TLS proxy would be able to identify it is as the same client as part of the normal operating procedure. This means it could learn the voters' name (at registration time) and later link it with their vote (at voting time).

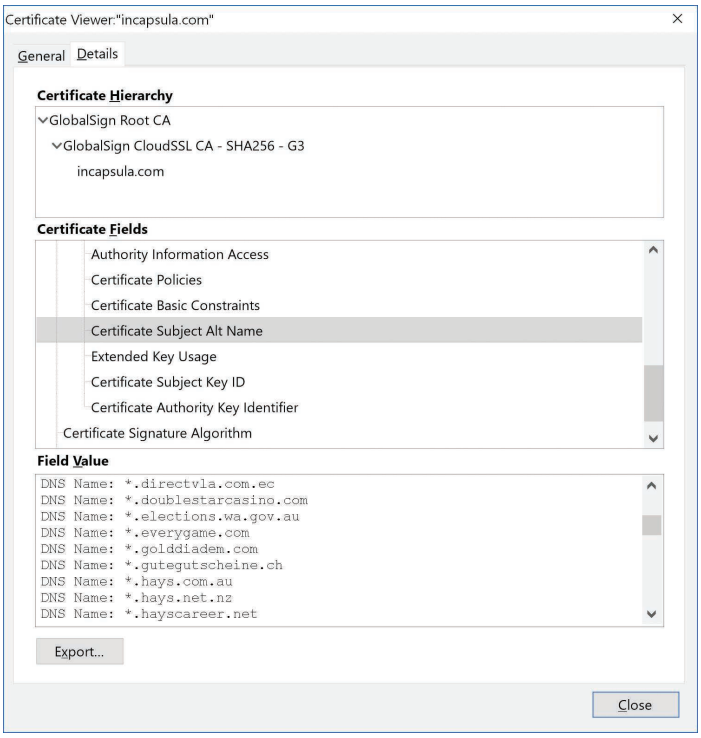

Figure 1. The certificate used to protect the connections between Western Australian Internet voters and the electoral commission. Note that elections.wa.gov.au is one of many domains that rely on this Incapsula certificate.

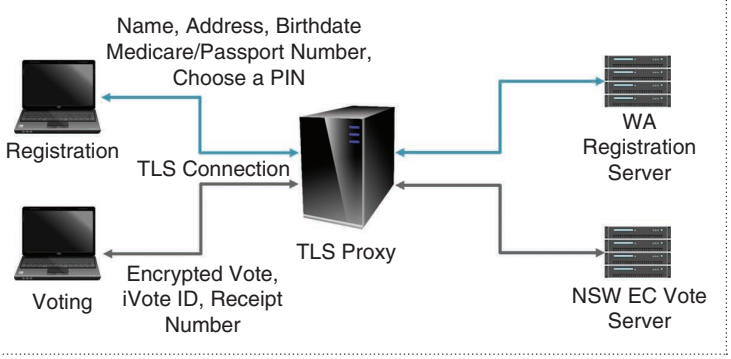

Figure 2. The TLS proxy deployed for iVote in Western Australia. WA: Western Australia; EC: Electoral Commission.

Summary of trust and transparency issues. It is difficult to justify including a foreign entity as a trusted man in the middle in an election system. It is particularly concerning that the use of such a service was not communicated to the voters or the public until after we published our findings.[7] As such, voters were interacting with a system that gave the impression it was communicating directly with the

Electoral Commission, when in fact it was not.

Who Wants Election Verification Anyway?

In the following section, we attempt to explain the structural and incentive problems that have contributed to the numerous technical issues we have observed.

Vendor Transparency, Source Code, Incentives, and Legal Punishments

The errors we detected in the SwissPost system were widely publicized, but the problems in the 2019 NSW system may have been worse, especially because they were identified only after the election began. Both code bases were available only under a nondisclosure agreement, which we did not sign. However, the Swiss law that mandates open access allowed the code to circulate quite freely in practice, so we could examine it without restrictions. NSW law is the opposite; it criminalizes the sharing of source code, making it completely unavailable in practice. Aside from the absurdity of sending Swiss officials to jail (because the e-voting systems have a common source code apparently), this means that decisions are made in NSW without any public feedback about the system. Currently, Switzerland is deciding whether or not to trust this system, and while we are uncertain of their decision, at least they will have more information than their NSW counterparts.

Unlike electronic voting machines, Internet voting systems cannot be examined outside of election time unless the authorities make a specific decision to make them available; attempting to demonstrate manipulation during a real election is (quite rightly) a serious crime. However, if the authorities choose to make no such opportunity available, this gives the real attackers (who are willing to break the law) a huge advantage over security researchers who would otherwise be able to identify problems and fix them.

Neither the updated SwissPost nor the NSW systems are openly available for public scrutiny, which makes it impossible to verify that the flaws we identified have been correctly repaired. And even if they were, there is no reason to think that those are the only errors that expose an election to undetectable fraud or privacy breaches.

Part of the problem is that electronic elections often correspond to outsourcing, i.e., the software is provided by a commercial entity with entirely different incentives from those of an electoral authority. Nobody likes to admit they've made a mistake, but a commercial enterprise makes money by convincing people to trust its systems, which is inconsistent with complete frankness about problems and errors.

Outsourcing reduces costs, but it also means outsourcing the trust that citizens place in their electoral authorities. Compromising on trust, privacy, and security to deploy a voting system just because it is cheaper or more convenient is not an acceptable course of action. To do so is a betrayal of democracy.

Cryptographic protocols for election verification are no different from cryptographic protocols for anything else: a protocol may have errors or weaknesses, but even if it doesn't, the implementation may suffer from errors or mistaken assumptions that can be exploited. In this case, exploitation means that the election may appear to come with a proof of its integrity, when in fact, the proof can be fabricated to conceal a successful manipulation.

Electoral authorities often make poor decisions because they do not recognize that it is their own systems, employees, and suppliers who might be the most easily exploited threat to election security.

It is hard work as well as a short-term reputational risk to offer citizens a genuine way of checking whether the election outcome is correct. It might seem easier, cheaper, and safer for the country's stability to offer an nontransparent system, with code available only under a secrecy agreement, and a reassuring appearance of verification regardless of whether there was error or fraud. However in the long term, this undermines trust in the integrity of elections.

We do not know whether any paperless e-voting system will ever prove itself adequate for government elections; thus far, none has. Certainly no election should be entrusted to a system that has never had meaningful open scrutiny.

We will begin to see improvements in the world's election conduct only when ordinary voters and candidates begin thinking critically about what sort of evidence they demand before they accept an election result. ■

Acknowledgments

We thank Andrew Conway, Matt Green, Peter Ryan, and Hovav Shacham for their help with the code, math, and report.

References

1. D. Anderson, C. Dunn, A. Dempster, B. Labby, and A. Neveu, "Fraudulent emails used to cast votes in UCP leadership race, CBC finds," CBC News, Apr. 10, 2019. [Online]. Available: https://www.cbc.ca/news/canada/calgary/ucp-leadership-voter-fraud-membership-lists-data-1.5091952
2. S. Bell et al., "Star-vote: A secure, transparent, auditable, and reliable voting system," *USENIX J. Election Technol. Syst. (JETS)*, vol. 1, no. 1, pp. 18–37, Aug. 2013.
3. S. Bayer and J. Groth, "Efficient zero-knowledge argument for

correctness of a shuffle," in *Proc. Advances in Cryptology (EUROCRYPT)*, 2012, pp. 263–280.
4. D. Bernhard, O. Pereira, and B. Warinschi, "How not to prove yourself: Pitfalls of the Fiat–Shamir heuristic and applications to Helios," in *Proc. Int. Conf. Theory and Application Cryptology and Information Security*, 2012, pp. 626–643.
5. J. Benaloh, R. Rivest, P. Y. Ryan, P. Stark, V. Teague, and P. Vora, End-to-end verifiability. 2015. [Online]. Available: https://arxiv.org/abs/1504.03778
6. R. Carback et al., "Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy," in *Proc. 19th USENIX Security Symp.*, 2010, pp. 291–306.
7. C. Culnane, M. Eldridge, A. Essex, and V. Teague, "Trust implications of DDOS protection in online elections," in *Proc. Int. Joint Conf. Electronic Voting*, 2017, pp. 127–145.
8. N. Chang-Fong and A. Essex, "The cloudier side of cryptographic end-to-end verifiable voting: A security analysis of Helios," in *Proc. 32nd Annu. Conf. Computer Security Applications*, 2016, pp. 324–335.
9. C. Culnane, P. Y. Ryan, S. Schneider, and V. Teague, "vVote: A verifiable voting system," *ACM Trans. Inform. Syst. Security (TISSEC)*, vol. 18, no. 1, p. 3, 2015.
10. A. Essex, "Detecting the detectable: Unintended consequences of cryptographic election verification," *IEEE Security Privacy*, vol. 15, no. 3, pp. 30–38, 2017.
11. R. Haenni, "Swiss post public intrusion test: Undetectable attack against vote integrity and secrecy," Bern Univ. Appl. Sci. Biel, Switzerland, 2019. [Online]. Available: https://e-voting.bfh.ch/app/download/7833162361/PIT2.pdf
12. R. Haenni, R. E. Koenig, P. Locher, and E. Dubuis, "CHVote system specification," Bern Univ. Appl. Sci. Biel, Switzerland, 2019. [Online]. Available: https://eprint.iacr.org/2017/325.pdf
13. J. A. Halderman and V. Teague, "The New South Wales iVote system: Security failures and verification flaws in a live online election," in *Proc. Int. Conf. E-Voting and Identity*, 2015, pp. 35–53.
14. A. Kiayias, T. Zacharias, and B. Zhang, "Demos-2: Scalable E2E verifiable elections without random oracles," in *Proc. 22nd ACM SIGSAC Conf. Computer and Communications Security*, 2015, pp. 352–363.
15. S. J. Lewis, O. Pereira, and V. Teague, "Addendum to how not to prove your election outcome: The use of non-adaptive zero knowledge proofs in the Scytl-SwissPost Internet voting system, and its implications for cast-as-intended verification," Univ. Melbourne, Parkville, Australia, 2019. [Online]. Available: https://people.eng.unimelb.edu.au/vjteague/HowNotToProveElectionOutcomeAddendum.pdf
16. S. J. Lewis, O. Pereira, and V. Teague, "Ceci n'est pas une preuve: The use of trapdoor commitments in Bayer-Groth proofs and the implications for the verifiability of the Scytl-SwissPost Internet voting system," Univ. Melbourne, Parkville, Australia, 2019. [Online]. Available: https://people.eng.unimelb.edu.au/vjteague/UniversalVerifiabilitySwissPost.pdf
17. S. J. Lewis, O. Pereira, and V. Teague, "How not to prove your election outcome: The use of non-adaptive zero knowledge proofs in the Scytl-SwissPost Internet voting system, and its implications for decryption sound proofness," Univ. Melbourne, Parkville, Australia, 2019. [Online]. Available: https://people.eng.unimelb.edu.au/vjteague/HowNotToProveElectionOutcome.pdf
18. M. Lindeman and P. B. Stark, "A gentle introduction to risk-limiting audits," *IEEE Security Privacy*, vol. 10, no. 5, pp. 42–49, 2012.
19. A. MacVicar, "Alberta NDP calls for special prosecutor to oversee RCMP investigation of UCP leadership race," Global News, May 2, 2019. [Online]. Available: https://globalnews.ca/news/5233913/notley-special-prosecutor-ucp-leadership-race/
20. E. Moher, J. Clark, and A. Essex, "Diffusion of voter responsibility: Potential failings in E2E voter receipt checking," *USENIX J. Election Technol. Syst. (JETS)*, vol. 3, no. 1, pp. 1–17, Dec. 2014.
21. R. S. Mueller, "Report on the investigation into Russian interference in the 2016 Presidential Election," U.S. Dept. of Justice. Washington, D.C., 2019. [Online]. Available: https://www.justice.gov/storage/report.pdf
22. Wombat. [Online]. Accessed on: May 22, 2019. Available: https://wombat.factcenter.org/
23. D. Springall et al., "Security analysis of the Estonian Internet voting system," in *Proc. ACM SIGSAC Conf. Computer and Communications Security*, 2014, pp. 703–715.
24. R. M. Smullyan, *What Is the Name of This Book?* New York: Touchstone Books, 1986.
25. Verificatum. [Online]. Accessed on: May 22, 2019. Available: https://www.verificatum.com.
26. S. Wolchok et al., "Security analysis of India's electronic voting machines," in *Proc. 17th ACM Conf. Computer and Communications Security*, 2010, pp. 1–14.
27. S. Wolchok, E. Wustrow, D. Isabel, and J. A. Halderman, "Attacking the Washington, DC Internet voting system," in *Proc. Int. Conf. Financial Cryptography and Data Security*, 2012, pp. 114–128.
28. https://elections.maryland.gov/press_room/documents/July%2013%20Press%20Statement.pdf
29. Swiss Post, "Swiss Post's e-voting solution: Electronic voting and elections for Switzerland." Accessed on: May 22, 2019. [Online]. Available: https://web.archive.org/web

/20190428114751/https://www.post.ch/en/business/a-z-of-subjects/industry-solutions/swiss-post-e-voting
30. O. Flüeler, "Ballot box not hacked, errors in the source code—Swiss Post temporarily suspends its e-voting system," Swiss Post, Mar. 29, 2019. [Online]. Available: https://www.post.ch/en/about-us/company/media/press-releases/2019/swiss-post-temporarily-suspends-its-e-voting-system
31. R. Lenzin, "Federal Chancellery to review e-voting," The Federal Council, Mar. 29, 2019. [Online]. Available: https://www.admin.ch/gov/en/start/documentation/media-releases.msg-id-74508.html
32. NSW Electoral Commission, "NSW Electoral Commission iVote and Swiss Post e-voting." Accessed on: May 22, 2019. [Online]. Available: https://elections.nsw.gov.au/About-us/Media-centre/News-media-releases/NSW-Electoral-Commission-iVote-and-Swiss-Post-e-vo
33. NSW Electoral Commission, "NSW Electoral Commission iVote and Swiss Post e-voting update." Accessed on: May 22, 2019. [Online]. Available: https://elections.nsw.gov.au/About-us/Media-centre/News-media-releases/NSW-Electoral-Commission-iVote-and-Swiss-Post
34. The Federal Council. (2013). 161.116 Federal Chancellery ordinance on electronic voting (VEleS), article 5.5. Federal Council. Bern, Switzerland. [Online]. Available: https://www.admin.ch/opc/en/classified-compilation/20132343/index.html

Chris Culnane is a lecturer of cybersecurity and privacy at the University of Melbourne, Australia. Contact him at christopher.culnane@unimelb.edu.au.

Aleksander Essex is an associate professor of software engineering at Western University, Canada. Contact him at aessex@uwo.ca.

Sarah Jamie Lewis is the executive director at the Open Privacy Research Society, Canada. Contact her at sarah@openprivacy.ca.

Olivier Pereira is a professor of cryptography at Université catholique de Louvain, Belgium. Contact him at olivier.pereira@uclouvain.be.

Vanessa Teague is an associate professor of cryptography at the University of Melbourne, Australia. Contact her at vjteague@unimelb.edu.au.

Digital Object Identifier 10.1109/MSEC.2019.2922090