Analyse und Schwachstellen

Christoph Bieringer, Simon Schneider

27.10.2022

Nachdem die Lösung der Vorgängergruppe nun genauer beschrieben wurde, wird sie im Hinblick auf ihre Eignung für Online-Wahlen untersucht. Hierzu werden zunächst einige Anforderungen an E-Voting-Systeme genannt und beschrieben. Anschließend wird untersucht, inwiefern das System diesen Anforderungen gerecht wird. Danach werden noch verschiedene andere Schwachstellen (sowohl technischer als auch methodischer Natur) des aktuellen Systems aufgelistet. Eine abschließende Zusammenfassung bildet den letzten Abschnitt.

Inhalt

Anforderungen an ein Wahlsystem

Ein Wahlsystem sollte im Idealfall eine große Menge an Anforderungen erfüllen. Die hier genannten sind eine Auswahl (leicht umgeordnet) aus (Neumann et al. 2016) und dienen nur als Richtlinien. Sie sollten keinesfalls als endgültig oder für ein Produktivsystem ausreichend angesehen werden.

1. Eligibility: Das Wahlsystem stellt sicher, dass nur die Stimmen von Wahlberechtigten gezählt werden.
2. Eligibility Verifiability: Es ist für Beobachter verifizierbar, dass die Anforderung „Eligibility“ eingehalten wurde.
3. Robustness: Das System gibt ein Wahlergebnis zurück.
4. Vote Integrity: Jede Stimme fließt korrekt in das Wahlergebnis ein.
5. Individual Verifiability: Es ist für den einzelnen Wähler verifizierbar, dass seine Stimme wie gewünscht abgegeben und gespeichert wurde.
6. Universal Verifiability: Es ist verifizierbar, dass alle abgegebenen Stimmen korrekt ausgezählt wurden.
7. Vote Secrecy: Das Wahlsystem liefert nicht mehr Informationen über die Absichten eines Wählers als das Wahlergebnis selbst[1]. Das Wahlgeheimnis wird also gewahrt.
8. Receipt-freeness: Das Wahlsystem liefert dem Wähler keinen Beleg (engl. Receipt), mit dem er einem Dritten beweisen könnte, für einen bestimmten Kandidaten gestimmt zu haben.

    > Detail: Es ist für den manipulationsfreien Ablauf einer Wahl von großer Wichtigkeit, dass auch die Wähler selbst ihr eigenes Wahlgeheimnis nicht verraten können, da dies Stimmenkauf[2] und/oder Zwang[3] ermöglicht.

9. Coercion-resistance: Das Wahlsystem stellt Mechanismen bereit, um Wähler vor Zwang zu schützen.
10. Accountability: Falls eine Verifikation fehlschlägt, kann die verantwortliche Partei identifiziert und zur Verantwortung gezogen werden.

Darüber hinaus existieren noch weitere Anforderungen an ein Wahlsystem.

11. Accessibility: Das System sollte für möglichst viele Wähler einfach und ohne Hilfe benutzbar sein.

Untersuchung des aktuellen Systems

Es folgt eine Untersuchung des aktuellen Systems hinsichtlich der gerade beschriebenen Anforderungen. Aus Zeitgründen konnten leider nicht alle Anforderungen detailliert auf Angriffsmöglichkeiten untersucht werden, sodass die Diskussion meist auf einer relativ oberflächlichen Ebene bleibt.

Um dennoch einen Eindruck von den Schwächen des Systems zu erhalten, wurde die Anforderung „Vote Secrecy“ gewissermaßen als Stichprobe etwas genauer untersucht. Für sie konnten einige konkrete Angriffsmöglichkeiten gefunden werden, mit denen ein böswilliger Akteur das

---

[1] Beispiel: Wenn ein Kandidat in einem Wahlkreis alle abgegebenen Stimmen erhält, dann lässt sich daraus schließen, dass jeder einzelne Wähler für ihn gestimmt hat. Kein noch so sicheres Wahlsystem kann an dieser Tatsache etwas ändern (es sei denn, das genaue Ergebnis wird nicht veröffentlicht, sondern nur der Name des Gewinners).

[2] „Wenn du beweisen kannst, dass du für mich gestimmt hast, dann [hier Belohnung einfügen].“

[3] „Wenn du nicht beweisen kannst, dass du für mich gestimmt hast, dann [hier Drohung einfügen].“

Wahlgeheimnis brechen könnte. Es ist gut möglich, dass ein kompetenter und gut ausgestatteter Angreifer auch zu den anderen Anforderungen Angriffe finden könnte.

Eligibility & Eligibility Verifiability

Da das aktuelle System keine Vorgaben oder Vorschläge für den Ablauf vor der Wahl abgibt, kann über diesen Teil der Anforderungen nur wenig gesagt werden. Ähnlich wie bei aktuellen (Brief)Wahlsystemen muss der zuständigen Behörde vertraut werden, nur Wahlberechtigten das Wählen zu erlauben. Im Fall des untersuchten Systems geschieht dies durch den Versand von Authentifizierungsinformationen einerseits und dem Eintragen dieser Informationen in die voters-Tabelle andererseits.

Die Wahlserver überprüfen vor der Stimmabgabe, oder der Wähler tatsächlich wahlberechtigt ist (d.h. ob ID und Authentifizierungscode in der voters-Tabelle stehen) und noch nicht gewählt hat. Solange dies verlässlich durchgeführt wird und die Behörde sich korrekt verhalten hat, können tatsächlich nur Wahlberechtigte wählen.

Ein kompromittierter Wahlserver dagegen könnte verschiedene Formen des Wahlbetrugs ermöglichen, etwa indem er bestimmte IDs mehrfach wählen lässt oder eine Stimmabgabe von nicht Wahlberechtigten zulässt. Solange dies nicht zu unsinnigen Ergebnissen[4] führt, wäre dies sehr schwer nachzuweisen. In der Blockchain werden nämlich nur Hashes gespeichert, von denen keinerlei Rückschluss auf die Identität gezogen werden kann.

Abhilfe könnte es hier schaffen, die voters-Tabelle (mitsamt der HAS_VOTED-Spalte) zu veröffentlichen. Auf diese Weise könnten Beobachter zumindest verifizieren, dass nur so viele Stimmen auf der Blockchain abgegeben wurden, wie es auch in der voters-Tabelle eingetragene, aktive Wähler (d.h. solche, die gewählt haben) gibt.

Robustness

Bei diesem Punkt haben papierbasierte Systeme aufgrund ihrer geringeren Anforderungen an die Infrastruktur naturgemäß einen deutlichen Vorteil[5]. Ein Cyberangriff, der flächendeckend das Internet lahmlegt, wäre bspw. fatal für das untersuchte System, aber nicht notwendigerweise für herkömmliche Wahlsysteme.

Aber selbst kleinere Angriffe könnten die ordnungsgemäße Durchführung der Wahl verhindern. In der aktuellen Form des Systems stellt bspw. das Register einen Single Point of Failure dar. Ein DDoS-Angriff auf es könnte dazu genutzt werden, entweder die Abgabe von legitimen Stimmen oder aber das Auslesen der Blockchain für die Auszählung zu verhindern und somit die Wahl zu stören.

Das aktuelle System hat keinerlei Deployment-Konzept und somit auch keine Vorkehrungen bspw. zur DDoS-Mitigation festgelegt. Ein solches Konzept zu erstellen wäre zumindest ein erster Schritt hin zu mehr Robustheit und eine mögliche zukünftige Erweiterung.

Vote Integrity

Durch die Blockchain ist die Integrität der gespeicherten Stimmen zunächst einmal garantiert, da jede nachträgliche Änderung einem Block die Integritätshashes der späteren Blöcke ungültig werden lassen würde. Dies ist aber für Beobachter nur schwer nachvollziehbar, da das Register in der aktuellen Version nur die Transaktionen auf der Blockchain (d.h. die [Hash, verschlüsselte Stimme]-Paare) veröffentlicht, nicht aber die gesamte Blockchain inkl. Hashes.

---

[4] Bspw. mehr abgegebene Stimmen, als es überhaupt Wahlberechtigte gibt.

[5] Wähler können zum nächsten Wahllokal laufen oder fahren, Wahlzettel lassen sich im Extremfall auch noch mit Kerzenlicht, Papier und Stift auszählen.

Wenn die gesamte Blockchain veröffentlicht werden würde, könnten sich auch externe Beobachter durch Überprüfen der Hashwerte von der Integrität der gespeicherten Stimmen überzeugen und so das Vertrauen in das System steigern. Diese Erweiterung wäre also ein lohnendes Unterfangen.

Individual Verifiability

Diese Anforderung erfüllt das aktuelle System schon relativ gut. Über den entsprechenden API-Endpoint des Registers kann sich ein Wähler jederzeit und beliebig oft davon überzeugen, dass seine Stimme korrekt verarbeitet und gespeichert wurde. Die Funktion ist zudem schon in das Frontend integriert, sodass hier ein niederschwelliges Angebot besteht.

Universal Verifiability

Die Auszählfunktion der Wahlserver steht prinzipiell jedem offen (und ist auch in das Frontend integriert). Allerdings muss der Wähler hierbei dem Wahlserver vertrauen können. Dieser könnte nämlich auch einfach falsche Wahlergebnisse zurückliefern. Solange nämlich die Wähler die auf der Blockchain gespeicherten Stimmen nicht selbst entschlüsseln und einsehen können, bestehen nur geringe Chancen, eine solche Manipulation des Wahlergebnisses zu erkennen.

Würde nämlich der Wahlserver bspw. für eine bestimmte Partei weniger Stimmen angeben, als sie tatsächlich bekommen hat, so ist dies für einen einzelnen Wähler nicht nachvollziehbar, da er ja nur seine eigene Stimme verifizieren kann. Um ein Ergebnis von n Stimmen für eine Partei als falsch zu entlarven, müssten im aktuellen System mindestens n+1 Wähler dieser Partei ihre Wahlentscheidung öffentlich machen (oder zumindest einer vertrauenswürdigen Stelle mitteilen) und verifizieren, um zu zeigen, dass der Wahlserver gelogen hat. Dieses Verfahren könnte besonders bei größeren Wahlen nur schwer organisier- und umsetzbar sein.

Vote Secrecy

Das aktuelle Design enthält bereits mehrere Features, die das Wahlgeheimnis schützen sollen. Die verschlüsselte Speicherung der Stimmen (mit Wählerschlüsseln verschlüsselt) und Wählerschlüssel (mit einem Administratorschlüssel verschlüsselt) ist eine davon. Das Mix-Netzwerk aus Wahlservern, mit dem Stimmen schwerer zurückverfolgt werden können, ist eine weitere. Auch die Verwendung von persönlichen Hashes, die sich nicht mehr auf IDs zurückführen lassen, gehört zu dieser Kategorie.

Alle diese Maßnahmen können aber im aktuellen System das Wahlgeheimnis nicht garantieren. Im Folgenden werden, wie eingangs erwähnt, mehrere Angriffe auf das System skizziert, die das Potenzial hätten, das Wahlgeheimnis zumindest teilweise (oder zumindest mit einer gewissen Wahrscheinlichkeit) zu brechen.

Angriff 1: Dieser Angriff basiert auf der permanenten Verfügbarkeit der Auszählfunktion. Der Angreifer überwacht den Verkehr zwischen dem Computer des Wählers und dem Wahlserver (es ist nicht einmal erforderlich, den Verkehr mitlesen zu können – die Kommunikationsmetadaten reichen aus)[6]. Parallel fragt er in kurzen Abständen die aktuelle Auszählung ab (und speichert deren Ergebnis). Sobald er zum Zeitpunkt t eine Kommunikation zwischen Wähler und Wahlserver beobachtet, vergleicht er die letzte Auszählung vor t mit der ersten Auszählung nach t. Falls nur eine der Parteien in der späteren Abstimmung mehr Stimmen als in der früheren hat, so kann der Angreifer sich sicher sein, dass der Wähler für diese Partei gestimmt hat. Je weniger Wähler es gibt, desto wahrscheinlicher, dass dieser Fall eintritt, weshalb dieser Angriff für kleinere Wahlen effektiver ist als für sehr große (bei denen einzelne Stimmen gewissermaßen in der Masse an Stimmen

[6] Über die Längen von Anfrage und Antwort könnte der Angreifer evtl. auch zwischen tatsächlichen Stimmabgaben einerseits und Auszählanfragen andererseits unterscheiden, um sich ganz sicher sein zu können, tatsächlich eine Stimmabgabe beobachtet zu haben. Diese Möglichkeit müsste aber noch genauer untersucht werden.

untergehen können). Durch kürzere Intervalle zwischen den Auszählanfragen (evtl. von mehreren Geräten aus, um weniger aufzufallen) kann der Angreifer zusätzlich die Erfolgschancen erhöhen (kürzere Intervalle -> höhere Chance, dass in einem Intervall nur wenige Stimmen abgegeben werden).

> Anmerkung: Man beachte, dass es für diesen Angriff nicht nötig ist, eine der Systemkomponenten zu kompromittieren. Ein (rein passiver) Zugriff auf die Kommunikation des Wählers sowie die öffentlich verfügbare Auszählfunktion reichen aus.

Angriff 2: Die Entry-Nodes des Mix-Netzwerks erhalten die Stimmen der Wähler im Klartext. Ein kompromittierter Wahlserver könnte also eine Zuordnung zwischen (bspw.) IP-Adressen und Stimmen anlegen, um die das Wahlgeheimnis einzelner Wähler zu brechen[7].

Angriff 3: Das Mix-Netzwerk bietet keine vollständige Verschleierung der Herkunft von Nachrichten. Insbesondere wenn in einem bestimmten Zeitraum nur wenige Wähler abstimmen, könnte ein Angreifer, der sowohl den Verkehr zu den Entry-Nodes hin als auch die auf der Blockchain eintreffenden Stimmen beobachten kann, eine Zuordnung zwischen Wählern und Stimmen, wie sie an das Register gesendet werden (d.h. persönliche Hashes und verschlüsselte Stimmen), vornehmen. Ähnlich wie Angriff 1 funktioniert dies umso besser, je weniger Stimmen abgegeben werden (bzw. je größer die Intervalle zwischen diesen sind).

Angriff 4: Ein kompromittierter oder einfach nur unsicher implementierter Wahlserver kann schwache oder im Voraus bekannte Wählerschlüssel erzeugen. Jeder, der über das entsprechende Wissen verfügt, kann dann Stimmen entschlüsseln, ohne Zugriff auf den Administratorschlüssel zu haben.

Angriff 5: Die Auswahl der PINs durch den Wähler birgt die Gefahr, dass Wähler unzureichende oder leicht zu erratende PINs (etwa 12345 o.ä.) verwenden. Ein Angreifer, der über die persönlichen IDs verfügt (bspw. ein Insider aus der Registrierungsbehörde), könnte eine Liste solch schwacher PINs ausprobieren, um eine Zuordnung zwischen IDs und persönlichen Hashes herstellen. In Kombination mit anderen Angriffen (etwa Angriff 4) könnte dies ausreichen, um eine Zuordnung zwischen Personen und Stimmen zu erreichen (und damit das Wahlgeheimnis zu brechen).

Angriff 6: Beim Verifizieren einer Stimme entschlüsselt das Register die Stimme des Wählers und leitet diese im Klartext an den Wähler weiter. Ein kompromittiertes Register kann also, ähnlich wie die Wahlserver in Angriff 2, eine Zuordnung zwischen Personen und Stimmen herstellen.

> Angriff 6a: Wenn das Register über den privaten Benutzerschlüssel verfügt, so kann es auch einen Teil der in der Blockchain gespeicherten Stimmen durch selbst gewählte und (mit den entsprechenden Wählerschlüsseln) verschlüsselte Stimmen ersetzen, bevor diese zur Auszählung an einen Wahlserver weitergeleitet werden. Da beim aktuellen System die Integritätshashes der Blockchain nicht mitübertragen werden und die gespeicherten Stimmen nicht gesondert authentifiziert werden, kann auch ein ehrlicher Wahlserver diese Form der Manipulation nur schwer erkennen.

Angriff 7: Ein kompromittierter Wahlserver könnte den Administratorschlüssel an dritte weitergeben, die mit diesem dann zunächst Wählerschlüssel und anschließend Stimmen entschlüsseln können. Wenn nun noch eine Zuordnung zwischen Wählern und persönlichen Hashes bekannt ist (etwas wie

[7] Auch wenn sich mehrere Wähler (z.Bsp. durch Carrier-Grade-NAT) eine IP-Adresse teilen, könnte der Angreifer mit anderen Methoden (etwa Browser-Fingerprinting) auf die Identität einzelner Wähler schließen.

in Angriff 3 und 5), lässt sich auch auf diese Weise eine Zuordnung von Personen und Stimmen herstellen.

Receipt-Freeness & Coercion Resistance

In diesem Punkt haben Briefwahl- und E-Voting-Systeme gegenüber „klassischen“ Systemen mit Wahlkabine und –urne einen deutlichen Nachteil.

Trotz anderweitiger Behauptungen der Vorgängergruppe ist nämlich in einem korrekt ausgestatteten Wahllokal die Vertraulichkeit der Wahl sowohl für Dritte als auch für den Wähler selbst nur schwer zu brechen. Durch

1. Das Ausfüllen des Stimmzettels in einer Kabine
2. Das Falten des Stimmzettels
3. Das Einwerfen des Zettels in eine (geschlossene und idealerweise blickdichte) Urne

ist es vergleichsweise schwer, einem Dritten einen Einblick in den eigenen Stimmzettel zu gewähren (insbesondere so, dass dies keiner der anderen Anwesenden mitbekommt).

Wenn der Wähler dagegen zu Hause oder an einem anderen Ort wählt, lässt sich nicht ausschließen, dass er den Wahlvorgang unbemerkt in irgendeiner Art dokumentiert (etwa durch ein Foto), um so einen Beleg für seine Wahl zu erhalten[8].

Das untersuchte System macht es noch einmal deutlich einfacher, einen Beleg für die eigene Wahl mit anderen zu teilen. Hierfür muss ein Wähler einfach die QR-Codes für seinen persönlichen Hash und seinen privaten Schlüssel, die er vom Wahlserver erhält, weitergeben. Der Empfänger der Codes kann dann (über die Verifikationsfunktion des Registers) selbst überprüfen, wie der Wähler abgestimmt hat. Es ist schwer zu sagen, wie diese Schwäche im aktuellen System behoben werden kann, ohne gleichzeitig die Verifizierbarkeit einzuschränken.

Accountability

Das aktuelle System hält keine Mechanismen bereit, mit denen die Verantwortlichen für fehlgeschlagene Verifikationen ausfindig gemacht werden können. Eine falsch in der Blockchain gespeicherte Stimme bspw. könnte sowohl die Schuld eines Wahlservers oder eines Registers sein, ohne das für einen externen Auditor eine Chance besteht, den wirklich Schuldigen zu finden.

Darüber hinaus ist auch nicht einmal festgelegt, wie das System auf verschiedene Fehler (bspw. eine fehlgeschlagene Verifikation) reagieren soll.

Accessibility

Das aktuelle Frontend wurde noch nicht detailliert auf Accessibility untersucht.

Es erscheint aber plausibel, dass in diesem Bereich einige Schwächen bestehen. So werden bspw. an verschiedenen Stellen nicht die semantisch korrekten HTML-Tags verwendet. Dies erschwert die Benutzung der Website für Sehbehinderte, die einen Screenreader verwenden.

[8] Fotos in einer Wahlkabine zu machen, ist dagegen 1.) unbemerkt deutlich schwieriger als zu Hause und 2.) an vielen Orten verboten und wird tlw. streng bestraft (vgl. bspw. https://en.wikipedia.org/wiki/Ballot_selfie).

Weitere Schwachstellen des aktuellen Systems

Technische Schwachstellen & Fehler

Neben den weiter oben genannten Angriffsmöglichkeiten und der mangelnden Erfüllung verschiedener Anforderungen besitzt das aktuelle System noch verschiedene weitere Mängel technischer Natur. Einige von ihnen werden hier kurz vorgestellt.

Mangelnde Threadsicherheit beim Zugriff auf die Datenbank

Der Vorgängergruppe war offenbar nicht klar, dass Flask-Applikationen von sich aus multithreaded sind. Der Flask-Server (bzw. in einem Produktivsystem der WSGI-Server) startet im Betrieb für jede Anfrage einen eigenen Thread. Es ist daher von elementarer Wichtigkeit, dass evtl. verwendete globale Objekte threadsicher sind (oder durch die Verwendung von Mutexes o.ä. threadsicher verwendet werden). Dies ist bei dem Connection-Objekt der verwendeten mariadb-Bibliothek, welches als globale Variable existiert, nicht der Fall. Bei mehreren Anfragen kurz hintereinander könnte es passieren, dass mehrere Threads zeitlich verschränkt auf das Objekt zugreifen. Dies kann zu sporadischen und schwer diagnostizierbaren Fehlern führen[9].

Hier besteht aber eine einfache Lösung. Die Funktionen aus auth_service.py, die auf die Datenbank zugreifen, können so umgeschrieben werden, dass sie jeweils ein eigenes Connection-Objekt erzeugen und nach der Benutzung wieder vernichten. Auf diese Weise greift jeder Thread auf sein eigenes Connection-Objekt zu.

Mangelnde Threadsicherheit beim Übergeben der neu erzeugten Wählerschlüssel

Ein ähnliches Problem befindet sich in encryption_service.py. Wann immer der Wahlserver einen Wählerschlüssel erzeugt, so wird der entsprechende private Schlüssel an die globale Variable private_key_list (ein Array) angehängt. Wenn nun mehrere zeitlich eng beieinander liegende Stimmabgaben den Wahlserver erreichen, könnte es analog zum vorigen Punkt dazu kommen, dass mehrere Threads zeitlich verschränkt auf private_key_list zugreifen. Dies kann u.U. dazu führen, dass einzelne Schlüssel vor dem Auslesen überschrieben oder doppelt verwendet werden. Auch dies könnte zu sporadischen Fehlern führen.

Eine einfach Lösung wäre es, die in der Funktion submit_vote_to_blockchain erzeugten Schlüssel als ein Rückgabewert der Funktion an den Wahlserver-Code zu übergeben, anstatt hierfür eine globale Variable zu verwenden. Da submit_vote_to_blockchain ohnehin schon mehrere Rückgabewerte hat, würde dies nur kleinere Änderungen am entsprechenden Code nach sich ziehen.

Datenbank-Passwort offen im Programmcode

Um ein möglichst großes Vertrauen in das System herzustellen, sollte ein möglichst großer Teil des Programmcodes als Open-Source-Code verfügbar sein. Dann darf aber keinesfalls das bisher verwendete Datenbank-Passwort offen in auth_service.py stehen, da ein Angreifer mit seiner Hilfe schwerwiegende, nicht einfach zu entdeckende Manipulationen an der Datenbank vornehmen könnte (er könnte bspw. seinen has_voted-Eintrag zurücksetzen, um mehrmals abzustimmen).

Auch hier ist die Lösung denkbar einfach: Das Passwort kann in eine separate Datei ausgelagert werden, die außer dem Passwort selbst keine Programmfunktionalität enthält und (im Gegensatz zum restlichen Code) nicht veröffentlicht wird. Die öffentliche Einsehbarkeit des Programmcodes bleibt damit gewahrt.

[9] Diese Erfahrung konnte einer der Autoren bei seiner Projektarbeit machen.

Beschränkungen der PIN-Auswahl

Die persönlichen PINs, die die User bei der Wahl angeben müssen (und für die Berechnung des persönlichen Hashes benutzt werden), müssen in der aktuellen Form aus genau sechs Ziffern bestehen, um vom Frontend akzeptiert zu werden.

Diese Beschränkung erscheint wenig sinnvoll. Sie reduziert den Keyspace für PINs auf eine Million Möglichkeiten, ohne dass hierfür eine technische Notwendigkeit besteht (die verwendete argon2-Implementierung arbeitet ohnehin mit Strings beliebiger Länge, nicht mit Zahlen). Gleichzeitig besteht die Gefahr, dass Nutzer leicht zu erratende PINs (etwa eine verkürzte Version ihres Geburtsdatums) als PIN verwenden.

Als Alternative könnten für die persönliche PIN passwortartige Regeln verwendet werden (z. Bsp. Mindestens 8 Zeichen, Groß- und Kleinbuchstaben, Sonderzeichen etc.)[10]. Dies würde die Menge an validen PINs deutlich erhöhen und gleichzeitig die Verwendung offensichtlicher PINs (etwa die schon erwähnten Geburtstage oder „123456“) zumindest etwas einschränken.

Methodische Schwachstellen

Neben den schon genannten technischen Mängeln und Angriffsmöglichkeiten existieren in dem aktuellen System noch verschiedene methodische Mängel, von denen im Folgenden einige genauer beschrieben werden. Auch diese Liste sollte keinesfalls als vollständig interpretiert werden, sondern nur einen ersten Eindruck von den Problemen geben, die vor dem Produktiveinsatz des Systems noch zu lösen wäre. Für eine deutlich umfangreichere Liste an Fragen, die beim Design eines E-Voting-Systems beantwortet werden sollte, sei der Leser an (Park et al. 2020) verwiesen.

Mangel einer vollständigen Untersuchung

Dieses Dokument kann, wie bereits erwähnt, keine vollständige Untersuchung des aktuellen Systems bieten. Eine solche (bspw. analog zu (SCRT SA 2022)) wäre aber vor einem evtl. Produktiveinsatz notwendig, um weitere Sicherheitslücken (zusätzlich zu den hier beschriebenen) zu finden und diese, falls nötig, zu schließen. Diese Untersuchung sollte von Experten durchgeführt und einem Peer Review unterzogen werden, um Korrektheit und Vollständigkeit der Analyse möglichst wahrscheinlich zu machen.

Selbst eine solche Untersuchung wäre aber keine absolute Garantie für die Sicherheit des Systems[11]. Dies zeigen auch die von der Vorgängergruppe genannten Beispiele, etwa die „Voatz“-App des gleichnamigen Start-Up-Unternehmens, bei denen in professionellen Systemen trotz genauer Untersuchung noch Schwachstellen gefunden wurden[12]. Auch bei dem Schweizer E-Voting-System, von der Vorgängergruppe als positives Beispiel herangezogen, wurden erst 2020, also mehrere Jahre nach der Einführung, Lücken im Verifikationsprozess gefunden (vgl. Haines et al. 2020).

Mangel eines Systems zur Klärung von Disputen

Der aktuelle Zustand des Systems stellt, wie weiter oben schon erwähnt, keine Regeln zur Klärung von Disputen bereit. Es ist bspw. nicht klar, an wen sich ein Wähler wenden sollte, wenn seine Authentifizierungsdaten nicht funktionieren oder seine Stimme bei der Verifikation nicht korrekt ausgelesen wurde.

---

[10] Auch wenn es sich dann bei den PINs streng genommen nicht mehr um Personal Identification *Numbers* handeln würde.

[11] Um es mit Bruce Schneier zu sagen: „Cryptography is harder than it looks.“

[12] Die erste fundierte Untersuchung der Schwachstellen in „Voatz“ findet sich in (Specter et al. 2020). Voatz Inc. stritt die Existenz der Schwachstellen zunächst ab, ein späterer Security Audit durch Trail of Bits Inc. (verfügbar unter https://blog.trailofbits.com/2020/03/13/our-full-report-on-the-voatz-mobile-voting-platform/) bestätigte aber deren Existenz.

Der Entwurf eines solchen Systems dürfte verschiedene Schwierigkeiten mit sich bringen. Insbesondere muss beachtet werden, dass ein böswilliger Akteur mittels dieser Schlichtungsstelle versuchen könnte, die Wahl zu verzögern (bspw. über vorsätzlich falsche Anfragen), das öffentliche Vertrauen in die Wahl zu verringern oder auf andere Weise den Stakeholdern der Wahl Schaden zufügen könnte. Die Schlichtungsstelle muss also gleichzeitig legitime Beschwerden zufriedenstellend behandeln können, ohne dabei missbraucht werden zu können. Mangels Expertise auf Seiten der Autoren konnte dieses Problem im Rahmen dieser Arbeit nicht weiter verfolgt werden.

Vertrauen auf kryptographische Primitive

Das aktuelle System vertraut darauf, dass verschiedene kryptographische Primitive sicher sind. An erster Stelle sind hier natürlich ECDH, AES, argon2 und SHA256 zu nennen. Allerdings benötigt das System für den Produktiveinsatz auch HTTPS, um die Kommunikation zwischen Wählern und Wahlserver/Register zu verschlüsseln (da sonst die Stimmen im Klartext übertragen würden).

Dieses Vertrauen sollte bei einer Anwendung mit der Bedeutung eines Wahlsystems kritisch hinterfragt werden. Der Ausgang von Wahlen kann nämlich (je nachdem, wer oder was gewählt wird) enorme persönliche, wirtschaftliche oder auch geopolitische Konsequenzen haben, nicht nur für die Wähler und Kandidaten, sondern auch für Dritte. Hieraus folgt, dass Angriffe auf das Wahlsystem von überaus motivierten Angreifern, die tlw. über sehr große Ressourcen verfügen, zu erwarten sind. Für die Sicherheit des Wahlsystems ist es also von großer Bedeutung, jeden denkbaren Angriff (auch sehr aufwändige) zu untersuchen. Natürlich kann dies in der aktuellen Arbeit allein aus Gründen der mangelnden Fachkompetenz der Autoren nicht zufriedenstellend durchgeführt werden. Dennoch kann im Folgenden (und im nächsten Abschnitt) auf einige grundlegende Probleme zumindest hingewiesen werden.

Zunächst einmal wäre die Verwendung von HTTPS und damit auch SSL/TLS zu nennen. Seit der Einführung von TLS wurden in diesem Protokoll mehrere Sicherheitslücken gefunden (vergleiche (Internet Engineering Task Force 2015) für eine Auswahl der bis 2015 gefundenen Schwachstellen). Auch wenn diese zum größten Teil rasch behoben werden konnten, ist nicht auszuschließen, dass in Zukunft noch weitere Schwachstellen gefunden werden. Ein Angreifer könnte eine solche Schwachstelle u.U. sogar selbst entdecken[13] und versuchen, sie möglichst lange geheim zu halten. Dann könnte er sie auf verschiedene Weisen ausnutzen, bspw. um in die verschlüsselte Kommunikation zwischen Wähler und Wahlserver einzugreifen.

Darüber hinaus verlässt sich die aktuelle Lösung auf externe Implementierungen von kryptographischen Primitiven und Funktionen. Dies ist zunächst einmal richtig, da laienhaft selbst implementierte Kryptographie ein großes Risiko darstellt. Andererseits kann auch die Verwendung der besten Bibliothek keine hundertprozentige Sicherheit garantieren. Es ist durchaus denkbar, dass auch in einer der verwendeten Implementierungen in Zukunft eine Schwachstelle gefunden wird. Historische Präzedenzfälle sind bspw. der Heartbleed-Angriff auf die weit verbreitete OpenSSL-Bibliothek. Auch hier gilt wieder dieselbe Überlegung wie im letzten Abschnitt – ein fähiger Angreifer könnte eine solche Schwachstelle sogar selbst als Erster finden und geheim halten.

Zudem wäre anzumerken, dass ein kryptographisches Primitiv oder Protokoll nicht unbedingt sicher sein muss, nur weil in der offenen Literatur keine realistischen Angriffe bekannt sind. Es wäre durchaus denkbar, dass bspw. feindliche Geheimdienste insgeheim über Erkenntnisse verfügen, die ihnen das Brechen der verwendeten Kryptographie ermöglichen. Selbst theoretische oder mathematische Durchbrüche, die einen Paradigmenwechsel in der Kryptographie einläuten würden,

[13] Wie weiter oben schon erwähnt, könnte es sich bei dem Angreifer um einen sehr gut ausgestatteten Akteur handeln – etwa einen Nachrichtendienst mit eigenen Kryptographen.

könnten der Öffentlichkeit lange Zeit verborgen bleiben. Hierfür existieren durchaus historische Präzedenzfälle. So entwickelte der für das britische GCHQ arbeitende Mathematiker Clifford Cocks ein dem RSA-Verfahren ähnliches, asymmetrisches Kryptosystem, fünf Jahre, bevor Rivest, Shamir und Adleman ihre eigene Arbeit veröffentlichten[14].

Vertrauen auf kryptographische Infrastruktur

Das weiter oben beschriebene Vertrauen auf HTTPS (und damit SSL/TLS) ist noch aus einem weiteren Grund problematisch. Für die Authentifizierung werden in dieser Protokollfamilie sog. Zertifikate verwendet, die von speziellen Zertifizierungsstellen (engl. Certificate Authorities, kurz CAs) digital signiert wurden. Eine Schwachstelle in diesem Authentifizierungsprozess kann es einem Angreifer ermöglichen, sich für einen Wahlserver/das Register auszugeben und somit einen Man-in-the-middle-Angriff auf die Kommunikation eines Wählers zu starten.

Ein großes Problem hierbei sind die CAs selbst. Ungenügende Sicherheitsvorkehrungen, menschliches Versagen oder Korruption könnten es dem Angreifer ermöglichen, ein gefälschtes Zertifikat zu erhalten. Hierfür existieren einige Präzedenzfälle, bspw. der DigiNotar-Hack 2011, bei dem Unbekannte (nachdem sie die IT-Infrastruktur einer CA kompromittiert hatten) mit gefälschten Zertifikaten u.a. einen Man-in-the-middle-Angriff auf Google-Dienste versuchten. Für einen detaillierteren Bericht dieses Zwischenfalls vergleiche bspw. (Hoogstraaten 2012), insb. Kapitel 11 („Lessons Learned“).

Auswirkungen von E-Voting auf Wahlbeteiligung und Partizipation

Neben den rein technischen Gesichtspunkten, die bisher behandelt wurden, gibt es natürlich auch nichttechnische Fragen zum Thema E-Voting. Aufgrund der mangelnden Fachkenntnisse der Autoren können diese hier nur selektiv und oberflächlich betrachtet werden. Eine ernsthaftere Untersuchung sollte sie aber zweifelsohne genauer zu beantworten versuchen, da sie insbesondere für politische Wahlen (im Gegensatz zu bspw. Gremiums- oder Vorstandswahlen) von großer Relevanz sind.

Die Vorgängergruppe äußerte in ihrer Dokumentation mehrfach die Hoffnung, dass ein E-Voting-System eine einfachere Wahlmöglichkeit schaffen und damit die Wahlbeteiligung vergrößern würde. Dies ist aber – auch mit Blick auf einige von der Vorgängergruppe selbst genannte Beispiele – nicht eindeutig zu belegen. So fanden bspw. (Germann und Serdült 2017) und (Ehin et al. 2022) keine signifikante Veränderung der Wahlbeteiligung nach der Einführung von E-Voting-Systemen in der Schweiz und in Estland.

Um in diesem Bereich fundiertere Aussagen treffen zu können, scheint eine genauere Untersuchung notwendig.

Allgemeine Vertrauenswürdigkeit des Wahlsystems/Risiko der Politisierung

Neben den Auswirkungen von E-Voting auf die Wahlbeteiligung muss noch ein weiterer nicht-technischer Punkt angesprochen werden. Verschiedene Ereignisse in der jüngsten Vergangenheit (bspw. im Rahmen der US-Präsidentschaftswahlen) haben gezeigt, dass Politiker u.U. durchaus dazu bereit sind, einzelne Wahlformen (bspw. die Briefwahl) und auch Wahlergebnisse allgemein anzuzweifeln und zu diskreditieren, wenn sie sich davon politische Erfolge versprechen. Dies kann unter Umständen das Vertrauen in die Demokratie schädigen und schwere Ausschreitungen verursachen (bspw. der Sturm auf das Kapitol 2021).

Vor diesem Hintergrund sollten sich die für eine Wahl Verantwortlichen überlegen, wie ratsam es ist, eine zusätzliche und noch wenig erprobte Form der Stimmabgabe einzuführen. Die

[14] Die Existenz von Cocks‘ Arbeit blieb tatsächlich sogar 24 Jahre lang geheim. Vergleiche (Ellis 1999) für eine detaillierte Beschreibung (auch des geschichtlichen Hintergrunds).

hochtechnologische Natur des E-Votings ist hierbei ein zusätzliches Hindernis, da ja der überwiegende Teil der Wahlberechtigten i.d.R.[15] nicht über die nötigen Fachkenntnisse verfügt, um sich persönlich von der Sicherheit des Wahlsystems zu überzeugen (auch wenn es sich um Open-Source-Software handelt). Dies dürfte die ohnehin vorhandene Skepsis gegenüber Änderungen am Wahlsystem (immerhin einer der Grundpfeiler jeder Demokratie) im Fall einer Einführung von E-Voting weiter verstärken[16].

Aus diesen Gründen könnte E-Voting sehr anfällig gegenüber politischen Diskreditierungsversuchen sein. Jeglicher Gewinn für die Demokratie (bspw. durch einfachere Teilnahme an der Wahl) könnte durch den hieraus folgenden Verlust von Vertrauen in den Wahlprozess wieder zunichte gemacht werden.

Zusammenfassung und Fazit

Das Wahlsystem in seiner aktuellen Form verfügt durchaus über interessante Ideen und Ansätze. Wie in diesem Dokument aber gezeigt werden konnte, liefert schon eine oberflächliche Analyse zahlreiche Angriffsmöglichkeiten und technische Probleme. Selbst wenn diese alle gelöst werden, bestehen noch weitere, schwerwiegende methodische Mängel, wie im letzten Abschnitt gezeigt werden konnte. Insbesondere ist der versprochene Nutzen bei weitem nicht so eindeutig wie von der Vorgängergruppe dargestellt, während es gleichzeitig mehrere schwere Risiken (sowohl technischer/kryptographischer als auch politischer Natur) gibt, die bisher ignoriert wurden.

Unter diesen Umständen muss von einer Verwendung des aktuellen Systems, selbst nachdem alle in dieser Arbeit beschriebenen Mängel beseitigt wurden, **dringend** abgeraten werden.

Trotz dieser Mängel zeigt das System, wie bereits erwähnt, interessante Ideen und Ansätze, nicht zuletzt die Verwendung einer Blockchain. Weitere Arbeiten hätten ausreichend Stoff, um die hier begonnene Analyse fortführen und eigene Erweiterungen/Verbesserungen durchführen zu können. Selbst wenn auf diese Weise sehr lange erst einmal kein einsatzfähiges System entsteht, könnten die dabei gewonnenen Erkenntnisse wertvoll und nützlich sein.

[15] Die Wahlen der Deutschen Gesellschaft für Informatik, die das POLYAS-E-Voting-System verwenden, bilden hier eine Ausnahme.

[16] Falls hieran Zweifel bestehen ist der interessierte Leser dazu eingeladen, eine fachfremde und nicht sehr technikaffine Person davon zu überzeugen, dass 1.) asymmetrische Kryptographie funktioniert und 2.) so sicher ist, dass man ihr den wichtigsten politischen Prozess eines Landes anvertrauen kann.

Quellen

Ehin, P., Solvak, M., Willemson, J., & Vinkel, P. (2022). Internet voting in Estonia 2005-2019: Evidence from eleven elections. *Gov. Inf. Q.*, 39, 101718.

Ellis, J.H. (1999). The History of Non-Secret Encryption. *Cryptologia, 23*, 267-273.

Germann, M., & Serdült, U. (2017). "Internet voting and turnout: Evidence from Switzerland." *Electoral Studies 47*, 1-12.

Haines, T., Lewis, S.J., Pereira, O., & Teague, V. (2020). How not to prove your election outcome. *2020 IEEE Symposium on Security and Privacy (SP)*, 644-660.

Hoogstraaten, H. (2012). Black Tulip Report of the investigation into the DigiNotar Certificate Authority breach. Online verfügbar unter https://www.researchgate.net/publication/269333601_Black_Tulip_Report_of_the_investigation_into_the_DigiNotar_Certificate_Authority_breach/link/5486fcf80cf268d28f06fa61/download (Letzter Zugriff 10.11.2022)

Internet Engineering Task Force (Hsg.) (2015). RFC 7457. Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)

Neumann, S., Volkamer, M., Budurushi, J., & Prandini, M. (2016). SecIVo: a quantitative security evaluation framework for internet voting schemes. *Annals of Telecommunications*, *71*(7), 337-352.

Park, S., Specter, M., Narula, N., & Rivest, R. (2020). "Going from bad to worse: from Internet voting to blockchain voting." *J. Cybersecur.* 7 (2021): n. pag.

SCRT SA (Hsg.) (2022). Examination of the Swiss Internet voting system. Onine verfügbar unter https://www.newsd.admin.ch/newsd/message/attachments/71144.pdf (letzter Zugriff 6.11.2022)

Specter, M., Koppel, J., & Weitzner, D. (2020). The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections. *USENIX Security Symposium.*