IntechOpen *Journals*

AI, Computer Science and Robotics Technology

REVIEW PAPER

Proof of Work and Secure Element in Electronic Voting

Vitaly Zuevsky*

Independent Researcher, Distributed Systems Consultant, Introspec Ltd, London, UK
*Correspondence: E-mail: vitaly.zuevsky@gmail.com

Abstract

Electronic voting consistently fails to supplant conventional paper ballot due to a plethora of security shortcomings. Not only are traditional voting methods mediocre in terms of transparency, audit, and costs, but they also encompass a principal-agent problem, where acting governments have real capability for tampering. Here I propose Proof of Work by user devices to fortify integrity of votes as cast and seed-isolated time-based one-time passwords to force observable polling stations. Coupled with end-to-end verifiability, the measures proposed are intended to improve on the issues mentioned. A state would only issue single-use authorizations to vote, while an untrusted publisher would collect and publish two unrelatable lists; votes and voters. Eventually, anyone could tabulate results. Content distribution networks are shown to be instrumental security providers. Weaknesses of proposed architecture are also discussed.

Keywords: e-voting, i-voting, PoW, proof of work, secure element, TOTP, CDN, edge computing, verifiable entropy, coercion

Citation
Vitaly Zuevsky (2023), Proof of Work and Secure Element in Electronic Voting. *AI, Computer Science and Robotics Technology* 2(1), 1–14.

DOI
https://doi.org/10.5772/acrt.28

Published
21 September 2023

1. Introduction

Democracy is a great achievement of human evolution; its implementation, however, is prone to fraud. Where state authorities manage elections, an inherent conflict of interest must exist as ruling cohorts have vested interest to be re-elected and possess real power to tamper with paper ballots. Perhaps, it is exactly that alleged conflict that gave rise to Capitol Hill riots we observed at Trump's departure in 2021. We could also recall some lengthy and expensive audit cycles following the events [1].

The problem has little prospect of solution in principle as long as the voting process remains on paper. There appears to be a growing number of electronic alternatives, however they seem fall short of convincing expert communities and the public that they are sufficiently fit for purpose [2, 3].

Here, a novel approach is presented, where tabulation of electronic votes is no longer a prerogative of a certain body. Instead, electronic voting records can be

downloaded and tabulated by any interested party. There are a few known problems with open votes. First of all, integrity of votes, that is, the votes remain as cast in a centralized store must be guaranteed. This is usually achieved by the voters returning after elections and verifying that their votes remain intact (a property known as end-to-end verifiability, or E2EV).

E2EV, in due turn, is a weak measure because it will not avert spurious votes thrown in by an adversary. Furthermore, E2EV implies an electronic receipt each voter is issued with in order to, later on, verify their vote. Such receipt can be abused to coerce the voter into voting the way a coercer demands.

The approach I present seeks to achieve the following:

- Voting as digital publishing to make tabulation inclusive and eliminate government's invisible hand.
- Cumulative work (Proof-of-Work scheme) by personal computing devices tackles DoS and hardens E2EV.
- Coercion-resistant receipt, such that a voter can verify their vote (E2EV) and also appease a coercer.
- Seed-isolated TOTP to pin vote-casting to designated (observable) sites. That is, polling stations persist.

Today, active research in electronic voting tends to build on sophisticated cryptography, such as homomorphic encryption, for example [4, 5]. Such approaches do not offer openness and transparency required to build public trust. Instead, treating the process of voting as a special case of publishing could build on content distribution networks (CDNs) that are, in effect, security pillars of the modern internet.

Today, practical implementations of electronic voting fulfill country-specific customary requirements. These systems tend to allow fully remote voting, for example, which could not be universally accepted due to opportunities for coercion it opens. Another example — a long standing Estonian system [6] identifies voters with government-issued IDs, which, again, cannot be generalized due to opportunities for governments to oppress.

It is worth noting that inception of Proof-of-Work (PoW) scheme predates its blockchain application by more than a decade. Its original purpose was pushing burden of request validation away from a central point of failure and, in effect, decentralize it among requesters. In blockchain applications the burden is distributed among "miners". Here, I suggest the greatest degree of distribution — among all voters. Chaining would not add much value since votes are cast within a limited timeframe (such as a day).

2. Secrecy

Going electronic makes it more difficult to observe secrecy in the process. A vote can be cast only once, that is, a vote-voter packet must be controlled for uniqueness, and a collector of such packets must see them coming in. A voter in the vote-voter packet is preferably a person's authorization (mandate) to vote, which needs be unique yet not divulging personal identity to the public. If the collector is the same entity that issues voting authorizations, such entity can easily trace a link between a vote and a person who has cast it through the authorization. Therefore, a collector of electronic votes, seeing vote-mandate pairs, must not communicate with an issuer of the mandates, holding mandate-real-identity associations, in order to preserve secrecy.

As governments are best positioned to define voters and issue their voting mandates, it is argued that an independent role, collecting and publishing raw data suitable for all further checks and final tabulation, should be established. Such role would take the agency conflict away from the state and strengthen secrecy in electronic approach.

I would like to set out a role model for further analyses. The following list of roles is deemed to be complete because they contain all functions of interest, and any data exchange between them are deemed to be private by means of TLS [7] protocol for example.

2.1. A role of authority

An authority—electoral commission, governmental portal or similar. We may have no better option than trusting the authority to define voters. The authority can electronically sign a one-time-pass (unique but not personally identifiable token) and share it with the voter as an authorization (mandate) to vote. The authorization identifies the voter to the authority; it gets published in the list of voters. Anyone could verify electronic signature on such authorization and, thus, establish that the holder was authorized to vote without knowing who they are.

One way of signing is by hierarchical chain of trust implemented with certificates. For instance, biometric travel documents use two-level hierarchy; country level and issuer level [8]. On the other hand, light-weight Ed25519 signatures [9] could be used in a flat trust model, where a list of public keys verifying voting mandates is maintained by the authority.

2.2. A role of platform

A platform outside the authority's control collects voting data packets from voters. We need such platform to reliably sever the link between the authorization to vote (personally identifying the voter to the issuing authority) and the vote itself. The

platform is a mere publisher of two uncorrelated lists: a list of voted authorizations (voters) and a list of cast votes. The platform's software is preferably open source. Concluding elections' outcome from the published lists is trivial and it may or may not be in the scope of the platform.

The platform is a voting-specific function, that is, it deals with voting-specific challenges, which may be contrasted with scalability-specific (non-functional) challenges. For example, while collecting data, the platform can run relevant sanity checks on the data, verifying that the voter is authorized by the authority, verifying that this voter has voted only once, etc. The platform operates redundant and fault-tolerant storage architecture.

Furthermore, the platform permits variety of publishing options. Although majority of jurisdictions ban real-time publishing to keep swing voters unpressured by instantaneous results [10], the platform should preserve frequently updated snapshots (near real-time) of the lists to be published. Such snapshots underpin a temporal structure of the data, which can be used in post-factum analyses of potential votes' throwing-in/out. Besides, real-time publishing of the list of voters should still be allowed as it doesn't contain votes, yet its dynamics can be matched against throughput of people having voted at polling stations.

Scale-related considerations are handled by the next role — a content distribution network (CDN), which the platform may or may not be a customer of.

2.3. A role of CDN

A content distribution network [11] or CDN is a pillar of the modern internet. It is designed to abstract and own scalability of digital services. The word "distribution" is dated, and content "collection" would be applicable today as well. The role of CDN is instrumental because it runs a massive fleet of hardware distributed at the edge i.e., close to the user-base or voters in our terms. That hardware faces internet users and tackles huge traffic they could generate, isolating and, thus, protecting functional components of a service (e.g., the platform) from threats commonly known under "Denial of Service" (DoS) umbrella. Figure 1 shows a high-level architecture envisioned.

CDNs work by creating content replica and caching them at the edge, so that user queries are answered by the CDN from its points of presence — not by the origin of the content. When content flows from the origin to users, caching reduces rate of queries landing at the origin — such queries "penetrate" CDN layer only when the content being requested by a user is missing from the cache.

Conversely, when content flows from users to the origin ("collection" mode) a feature called "edge computing" can be engaged. A user may be required to attach to their query a proof of a threshold amount of computational work their device has

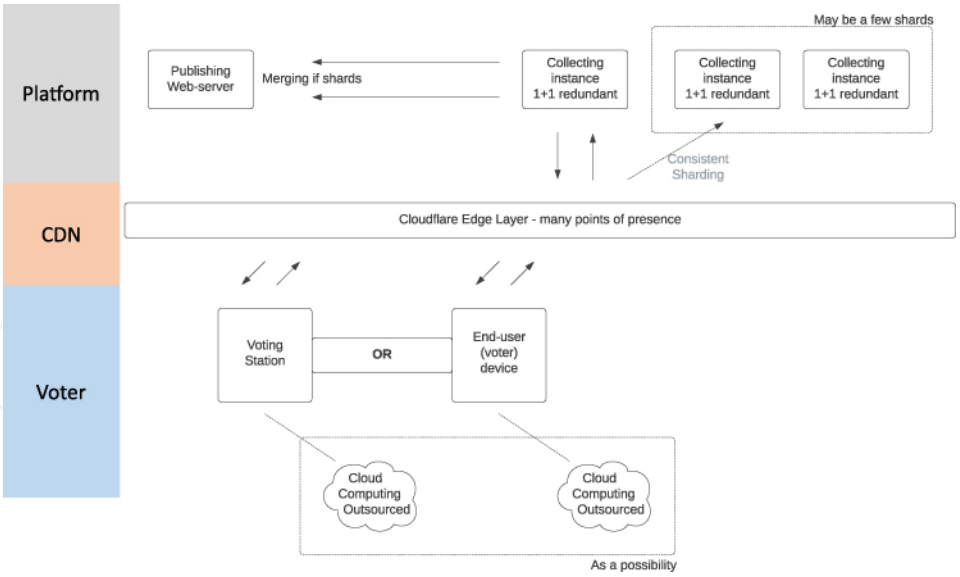

Figure 1. A high-level architecture of an electronic voting system with distributed integrity guard (proof of voters' work) and isolating edge layer (CDN) to mitigate potential denial-of-service attacks.

performed. The PoW algorithm [12] builds on asymmetry in time needed to calculate the proof (long) vs. its verification (fast). In other words, a malicious party would not be able to bombard the system with queries without spending considerable amount of time on each; and queries without valid proof could be detected early by the CDN and dropped without harm inflicted. We further rely on caching of the proof in order to eliminate replaying attack, wherein a once-valid query is being played out in cycle.

Broadly speaking, CDN can be seen as part of the platform. For instance, CDN nodes process unencrypted data from voters. One caveat, however, is that CDN could assume voter's role in business of coercion (discussed later). As such, CDN might have data to not be shared with the platform.

2.4. Role of voter

A voter casts their vote with some software running on their device(s). This software is preferably open sourced by the platform. The software calculates PoW and attaches it to the vote being cast. The voter can latterly verify that their vote in the platform's list is published as cast.

3. Integrity

End-to-end verification (E2EV) [13] is presumably the strongest guarantor of integrity in electronic voting to the date. When voter can act and verify that their

vote is counted as cast, it eliminates very need of any guarantees around processes (or software for that matter) between the voter and the final tally.

There are at least two E2EV shortcomings one could anticipate. First of all, E2EV relies on a second deed an individual is expected to voluntarily perform i.e., not only is a person expected to come to vote, they will further be expected to return to verify their vote. It is reasonable to anticipate that only a fraction of voters will engage in E2EV. Moreover, as E2EV feature matures and becomes habitual, that fraction is likely to shrink as people will assume success by prior experience and withdraw. If a fraction of votes got fraudulently changed, it is "a fraction of a fraction" who would notice.

Secondly, E2EV cannot tackle stuffing. If a fraudulent government, or whoever is able to issue a valid voting authorization on its behalf, votes with the authorization not backed by a real person, there will be no one to engage in E2EV from the outset.

I was looking to mitigate those shortcomings and aid E2EV with something that would be as much process/software-agnostic while completely passive — not relying on deferred user actions.

3.1. Proof of work as a hedge

PoW was first mentioned in relation to defense from DoS attacks in paragraph 2.3. That defense can be readily extended onto the list of votes as a whole. To change a vote an adversary would have to invest the same amount of computational work as the voter invested originally. The same holds if an adversary was to throw in a spurious vote. Cumulative work performed by voting devices in this highly distributed manner can represent a significant hedge against tampering at scale. That concept is process/software-agnostic because anyone can determine amount of work over every vote on the public list regardless the process/software the list was formed by.

Let's allude to PoW basics. Every record on the list of votes is expected to encompass a so-called nonce (proof) — a sequence of bytes that could only be obtained by repetitive calculations of a cryptographic digest (work), and the digest is calculated over the record being protected. Obviously, a vote is just another member of the record. The nonce is found by trial and error brute-force endeavor. When the record, containing the nonce, is fed back to the digest function [14], a digest produced signals amount of work expected to have been performed. For instance, SHA-256 digest with 20 leading zero bits can conventionally signal 2^{20} = 1,048,576 repetitions of SHA-256 calculation the record holder has been through on average. With that mechanics in mind, we will consider what other members a voting record could benefit from.

3.2. Trust and conspiracy

No role can be trusted, and every possibility to cheat should be explored and discussed. For instance, a rogue authority can use bots to vote with authorizations not representing real people. This is presumably mitigated by the rate-limiting implication of the PoW on one hand, and by correlating polling stations' throughput with growth of voters list on the other (both discussed above).

The authority could further forge a bunch of PoW-protected votes in advance i.e., taking virtually unlimited time prior elections. To tackle this scenario the platform must issue a non-fudgeable stamp. That platform stamp will be the next member of a voting record covered by PoW. That is, a voter must request the stamp from the platform before embarking on PoW computations. Generally, the platform stamp forces all PoW runs to happen within a limited time frame.

Conversely, the platform could conspire with an adversary or just be one. To ensure the platform couldn't prepare fraudulent voting records in advance voter-side non-fudgeable stamp must be the next member of a voting record covered by PoW. The voter stamp shares all the properties of a voter receipt used in E2EV; we will later discuss it in detail.

The authority could further conspire with the platform or CDN to de-anonymize votes (breach secrecy) — that remains unaddressed. Finally, organized sabotage by the voters remains a conundrum. For example, if thousands of voters made any false yet coherent claims with the aim to derail elections, we would have little choice but oblige.

Figure 2 shows a data packet that a voter sends to the platform by casting their vote.

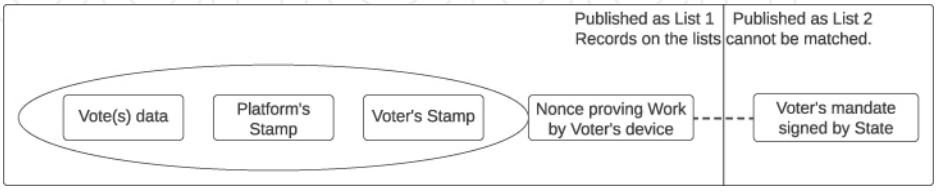

Figure 2. A data packet en route from a voter to the platform. Stamps ensure the nonce proves the work done within a certain time frame. Voter's mandate ensures that the voter was authorized and voted once. The platform splits the packet and updates two uncorrelated lists.

4. Coercion

I acknowledge that voters can be coerced either by means of technology or regardless of technology whatsoever. Both cases are addressed, and we start with the former.

One quick way to make the voter stamp (Figure 2) would be shaping an entropy generated by the voter's device into a universally unique identifier [15]. A coercer could then provide any such identifier generated beforehand and known to the coercer. Since that identifier appears on the list of votes, coercer can find out the voter's vote.

This is a case, where a CDN being the closest role to the voter could potentially assume voter's role from platform's point of view and provide some voter-side entropy that is verifiable by the CDN.

4.1. CDN-controlled entropy as a service

It is worth noting that a service offering publicly verifiable entropy at infrastructure level exists [16, 17]. It is not practical for us because production is too slow, and tokens are too cumbersome. Instead, we could take advantage of CDN's highly distributed pool of entropy — Internet users. Specifically, symmetric key or 48-byte master secret in terms of TLS protocol [7] shares entropy from client and server. Although the secret's derivatives could be used as the sought voter's stamp (E2EV identifier), the stamp could only be verified within a single TLS session (assuming CDN doesn't maintain state outside TLS connection). That is prohibitive in practice. Although TLS state is a good source of entropy, we need better control over its CDN-side verification. I envision the following functions in edge-compute environment for example:

```
edge.verand_get(int n, int s, char* salt)

edge.verand_check(int n, int s, char* verand, char* salt)
```

where a verifiable random of (n + s) bytes consists of n-byte entropy and first s bytes of HMAC signature [18] of that entropy. This design allows false positive verification with probability 2^{-8s} traded off against size of the entropy portion. HMAC signature should depend on a salt and a CDN-side secret with a certain lifecycle.

CDN-verifiable and non-fudgeable in that sense UUID appears on a voter's receipt against their vote. The platform can further stuff that receipt with all other voting options and populate corresponding UUIDs belonging to other voters from a cache (meaning those other votes were cast contemporaneously). Figure 3 shows an example of a receipt where nine solar planets are on ballot:

A voter must remember the choice they made at voting as their UUID on the receipt is against that choice.

Figure 3. An example of coercion-resistant receipt a voter gets for E2EV. Voter stamps are on the left-hand side and all the ballot options are on the right. Every vote on the receipt is positively verifiable at E2EV.

4.2. Polling stations

As much as electronic voting is appealing for convenience of voting from home, remote mode is inherently prone to coercion, regardless of the capabilities technology holds. Besides, polling stations are indispensable when it comes to verification of the number of voters.

What I hope to improve on, however, is the cost of equipment polling stations operate. Whether voters use own devices or those provided by a polling station — they will be commercial off-the-shelf (COTS) hardware as opposed to specialized builds of today.

Polling stations pose a couple of problems for electronic approach. First of all, the platform needs a way to verify that a voter casts their vote (sends Figure 2 packet to that effect) from a polling station, not somewhere else. Secondly, autocratic states could coerce *by* polling stations, rigging them up out of sight of the platform and the public.

Both problems can be addressed by "a hand of platform" at every polling station in operation. That "hand" produces time-based one-time passwords (TOTP) [19] for example. In authentication use case, where TOTP commonly serves as a "second factor", a secret underpinning the algorithm (a seed) can be accessed/extracted by the user, which is fine. Our use case, in contrast, requires the seed to be isolated from the untrusted environment and withstand extraction attempts by the state actors. I could see other pursuits of cryptographic proofs of location in contexts of network infrastructure and witnessing neighbors [20], none of which fits the bill.

4.3. Secure element

A secure element (SE) is a system-on-chip (SoC) electronic component targeting primarily IoT sector [21]. Since NXP Semiconductors [22] manufactures similar components for biometric passports, I used their EdgeLock SE050 chip in this research. I further used a high-accuracy real time clock (RTC) DS3231 and a microcomputer Raspberry Pi model 4B (RPi).

The SE stores a secret (key) shared with the platform and calculates HMAC-SHA-256 with that secret and from any input data. It further stores a key-wrapping key as per RFC 3394 [23], which allows updating the secret with its AES-256 cyphertext. All that happens without crossing the boundaries of the chip.

Software running on the RPi supplies an RTC-based input to the SE, reads back HMAC bytes and converts them into TOTP. The platform stores a public registry of associations between polling stations and SEs. Since the platform could deny votes not vouched for by an SE from the registry, a coercive authority might struggle with unregistered stations. Figure 4 shows an experimental SE in action.

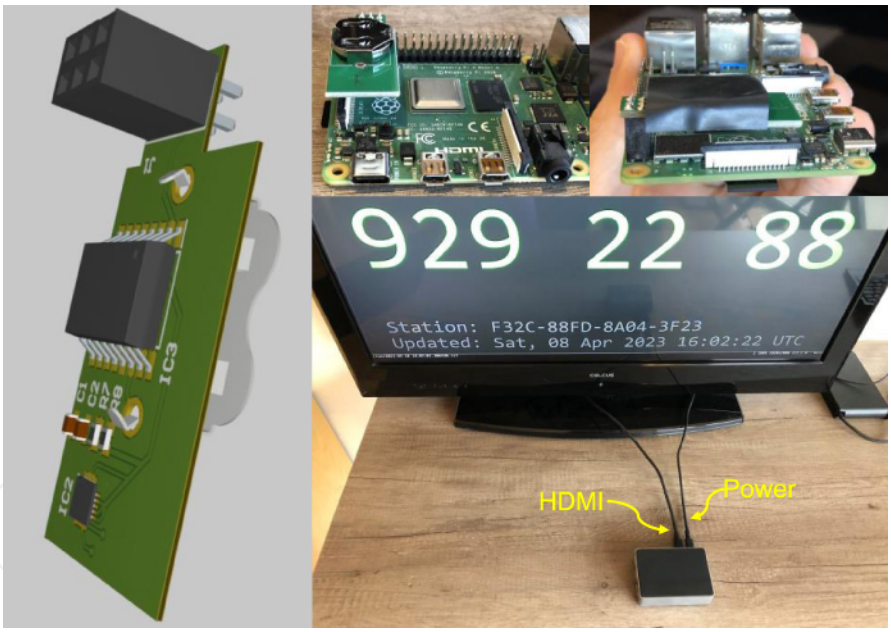

Figure 4. A "hand of platform" in a polling station: SE and RTC are integrated with RPi to reliably isolate and update a secret shared with the platform. TOTP derived from the secret is for voters to enter next to their votes.

4.4. TOTP call-flow

The "hand of platform" in polling stations is supported by a pluggable process on top of a query a voter sends to obtain stamps (Figure 2). Figure 5 demonstrates that

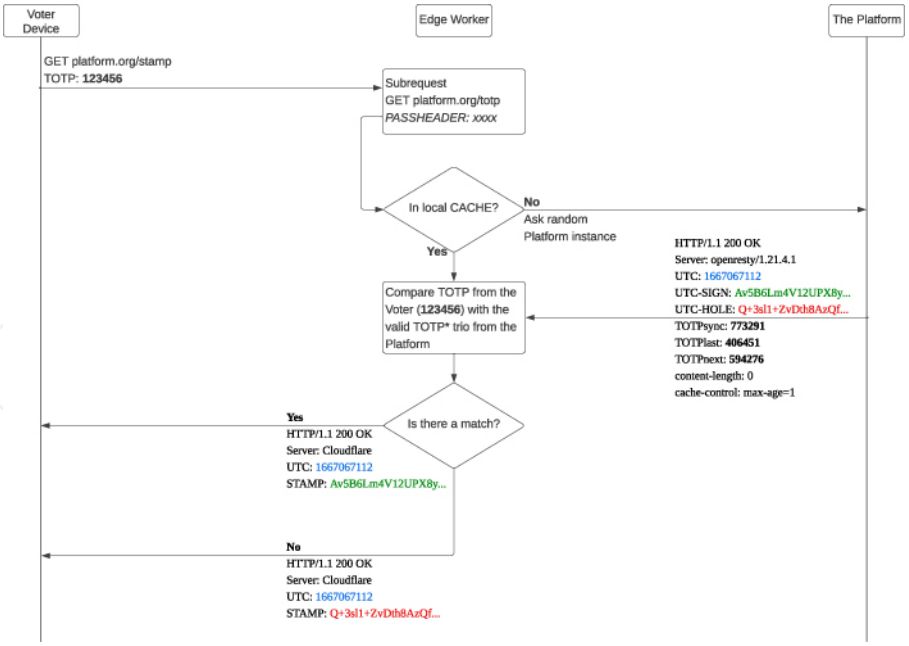

Figure 5. TOTP call-flow — part "/stamp" HTTP-request handling. The platform owns time-based one-time passwords (TOTP) that are only available at polling stations. Edge logic run by CDN prevents potential attacks.

process by example of a DoS- and brute-force-resistant call-flow with the following features:

- Platform's responses are cached at the edge for one second. Therefore, the load the platform is subjected to is limited by a number of edge machines in any one second. That holds even if the edge is getting millions requests per second in a DoS scenario.
- Colored elements in Figure 5 change every second. TOTP changes every 30 s. Stamp requester (voter device but may be an adversary) must not be able to discriminate between a valid (green) or an invalid (red) stamp returned, so that a brute-force attack would lack criteria of success.
- Stamps (UTC-SIGN, UTC-HOLE, and STAMP) are base64 encoded Ed25519 signatures of the UTC header. Public keys verifying them should still remain secret during voting period, so that an adversary couldn't tell valid and invalid stamps apart.
- PASSHEADER guarantees the platform only respond to the edge — not anyone else.
- All shown transactions are encrypted by means of TLS.

5. Procedure and limitations

Zooming into a voting moment, these fast pathways are to be traversed: obtaining a vote and TOTP from the voter, obtaining stamps (Figures 2, 5). Next, a search for PoW can start. It is not possible to predict how long the search will last. Furthermore, the longer the search lasts on average — the stronger the hedge against potential tampering will be. Although cumulative number of computational cycles invested by all voters will be evident, effectiveness of the hedge against capabilities of the state actors has not been assessed. It is difficult, therefore, to decide on a threshold amount of work as it is a trade-off between unspecified effectiveness and suboptimal user experience.

Since a voter has to wait in the PoW search after casting their vote, it may make sense to reverse traditional order and ask the voter to vote first (in a booth or designated area) and to see the station's staff to obtain their authorization (incl. ID checks) afterwards. The latter may have form of scanning a QR code by the voter's device. Provided that the search of proof of threshold work is over by then, the voting packet (Figure 2) can be sent and the receipt (Figure 3) — received right on the spot at the staff.

Alternatively, if a voter is using station's equipment, the order reversal is unnecessary. Instead, the voter can spend a few minutes, otherwise required for PoW, typing their authorization into a computer at the station manually (refer to Crockford's encoded examples in the repository provided below).

TOTP workflow does not fully cover coercive opportunities. A problem that a rogue voter could electronically transmit TOTP from a polling station to a group of voters being coerced is arguably addressed by the circumstance that every member of the group would still come through a "booth" area on their way to authorization, where they could have their "last say". That is assuming the software on their devices is genuine. A coercer could build their own software, in which case, leaked TOTPs remain an issue.

On the other hand, abusing leaked TOTPs or, generally, stamps (Figure 2) beyond a certain scale would transpire in the aftermath of elections, where pattern recognition algorithms may detect outliers in published data. The stamps have temporal granularity of one second, and outliers on a trend of their accumulation per candidate could flag the abuse in relatively sensitive manner.

6. Digital artefacts

Architecture described herein is in Proof-of-Concept stage and available for testing at https://clairvote.org. One could gather booth TOTP and voting authorization, cast a vote protected by PoW, and download voting records to process them locally

and display results. The code and low-level API documentation as well as python implementation can be found at https://github.com/psvz/clairvote.

Conflict of interest

The author declares ownership of active and pending patents in the United States that can be related to this work.

References

1 Olorunnipa T, Lee MYH. *Trump's election fraud falsehoods have cost taxpayers $519 million—and counting* [Internet], The Washington Post; 2021 Feb 6 [cited 2023 Apr 30]. Available at: https://www.washingtonpost.com/politics/interactive/2021/cost-trump-election-fraud/.

2 Oostveen A-M. Outsourcing democracy: losing control of e-voting in the Netherlands. *Policy Internet*. 2010;**2**: 201–220. doi:10.2202/1944-2866.1065.

3 Cortier V, Debant A, Gaudry P. A privacy attack on the Swiss Post e-voting system. In: *Presented at the RWC 2022 - Real World Crypto Symposium, April 13*. Grand Est: LORIA; 2022.

4 Saproo S, Warke V, Pote S, Dhumal R. Online voting system using homomorphic encryption. *ITM Web Conf*. 2020;**32**: 03023, doi:10.1051/itmconf/20203203023.

5 ElectionGuard - Structures and Processes [Internet] [cited 2023 Jul 30]. Available from: https://www.electionguard.vote/concepts/Structure_and_Processes/None.

6 Ehin P, Solvak M, Willemson J, Vinkel P. Internet voting in Estonia 2005–2019: evidence from eleven elections. *Gov Inform Quart*. 2022;**39**: 101718, doi:10.1016/j.giq.2022.101718.

7 Rescorla E, Dierks T. *The Transport Layer Security (TLS) Protocol Version 1.2*. Internet Engineering Task Force; 2008 Aug. p. 104. doi:10.17487/RFC5246.

8 ICAO Uniting Aviation. *Doc Series* [Internet]; [cited 2023 May 1]. Available from: https://www.icao.int/publications/pages/publication.aspx?docnum=9303.

9 Bernstein DJ, Duif N, Lange T, Schwabe P, Yang B-Y. High-speed high-security signatures. *J Cryptogr Eng*. 2012;**2**: 77–89. doi:10.1007/s13389-012-0027-1.

10 Epstein R, Robertson RE. The search engine manipulation effect (SEME) and its possible impact on the outcomes of elections. *Proc Natl Acad Sci USA*. 2015;**112**: E4512–E4521. doi:10.1073/pnas.1419828112.

11 Wikipedia. *Content delivery network*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=Content_delivery_network&oldid=1149449619.

12 Wikipedia. *Proof of work*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=Proof_of_work&oldid=1149255768.

13 Bernhard M, Benaloh J, Alex Halderman J, Rivest RL, Ryan PYA, Stark PB, Teague V, Vora PL, Wallach DS. Public evidence from secret ballots. In: Krimmer R, Volkamer M, Braun Binder N, Kersting N, Pereira O, Schürmann C, editors. *Electronic Voting: Second International Joint Conference, E-Vote-ID 2017, Bregenz, Austria, October 24–27, 2017, Proceedings*. vol. 2, Cham: Springer; 2017. p. 84–109. doi:10.1007/978-3-319-68687-5_6.

14 Wikipedia. *Cryptographic hash function*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=Cryptographic_hash_function&oldid=1171216567.

15 Wikipedia. *Universally unique identifier*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=Universally_unique_identifier&oldid=1172593602.

16 Syta E, Jovanovic P, Kogias EK, Gailly N, Gasser L, Khoffi I, Fischer MJ, Ford B. Scalable bias-resistant distributed randomness. In: *2017 IEEE Symposium on Security and Privacy (SP)*. Piscataway, NJ: IEEE; 2017. p. 444–460. doi:10.1109/SP.2017.45.

17 Wikipedia. *League of Entropy*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=League_of_Entropy&oldid=1172999149.

18 Wikipedia. *HMAC*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=HMAC&oldid=1172552533.

19 M'Raihi D, Rydell J, Pei M, Machani S. *TOTP: Time-Based One-Time Password Algorithm*. Internet Engineering Task Force; 2011. p. 16. doi:10.17487/RFC6238.

20 Gambs S, Traoré M, Roy M, Killijian M-O. PROPS: a privacy-preserving location proof system. In: *Proceedings of the IEEE Symposium on Reliable Distributed Systems, Nara, Japan*. vol. 2014, Piscataway, NJ: IEEE; 2014. p. 1–10. doi:10.1109/SRDS.2014.37.

21 Wikipedia. *Secure element*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=Secure_element&oldid=1120230117.

22 Wikipedia. *NXP Semiconductors*. The Free Encyclopedia [Internet]; 2023 [cited 2023 Sep 4]. Available from: https://en.wikipedia.org/w/index.php?title=NXP_Semiconductors&oldid=1170392061.

23 Housley R, Schaad J. *Advanced Encryption Standard (AES) Key Wrap Algorithm*. Internet Engineering Task Force; 2002. p. 41. doi:10.17487/RFC3394.