Æternum: A Decentralized Voting System with Unconditional Privacy

Christian Killer, Markus Knecht, Claude Müller, Bruno Rodrigues,
Eder Scheid, Muriel Franco, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH
Binzmühlestrasse 14, CH—8050 Zürich, Switzerland
[killer|rodrigues|scheid|franco|stiller]@ifi.uzh.ch, markus.knecht2@uzh.ch, claude.muller@protonmail.com

Abstract—Remote Electronic Voting (REV) systems allow voters to cast their votes in an uncontrolled, distributed environment. At the same time, the REV system must provide ballot privacy and verifiability of the final tally. Research has proposed REV schemes offering ballot privacy based on computational intractability assumptions, but only a few provide Unconditional Privacy (UP).

Therefore, this work proposes Æternum, a REV system with a voting scheme providing UP. Æternum does not require trust in a central authority, nor does it assume computational intractability of an underlying mathematical problem to provide UP. To satisfy UP’s minimal trust assumptions, Æternum uses a permissioned Distributed Ledger (DL), that forms a decentralized network of permissioned nodes, which serve as a transparent, tamper-proof Decentralized Public Bulletin Board (DPBB).

I. INTRODUCTION

Remote Electronic Voting (REV) is an essential topic, considering the further digitalization of society, which demands an increasingly efficient, transparent, and verifiable voting processes. REV allows for casting votes over the internet, applying cryptographic protocols to achieve End-to-End (E2E) verifiable schemes [30]. Furthermore, immutable Public Bulletin Boards (PBB) offer a transparent audit trail of the voting process. REV requires less physical labor at different stages of the voting process and infrastructure [33], if designed securely and correctly, it allows for tallying results faster, in contrast to traditional, paper-based voting systems. REV reduces the threat events of traditional remote voting systems (e.g., potential threat events emerging during tallying, or the manipulation of votes sent by postal mail [31]). However, the complexity and technical operations of a distributed high-security system introduce new challenges, e.g., trade-offs between verifiability, transparency of the process, and voters’ privacy.

Ballot Privacy (BP) is one of the core requirements in any electoral process. In the context of REV, BP is divided into basic, everlasting, and unconditional BP. Basic BP relies on the assumptions of (a) computational intractability and (b) a trusted central authority [27]. Since future adversaries may access Quantum Computers (QC), it would be possible to compromise the confidentiality and integrity of such cryptographic primitives by breaking the underlying hardness assumptions, e.g., Integer factorization, and the discrete logarithm [14]. It is crucial to protect BP against present and future adversaries, thus, not relying on computational intractability.

However, REV schemes often apply cryptographic protocols based on Mixnets or Homomorphic Encryption, which rely on hardness assumptions to achieve BP [27]. Thus, Unconditional BP (UP) does not assume computational intractability and eliminates the need for Trusted Third Parties (TPP) [36]. Minimizing trust assumptions and removing the reliance on TTPs for specific tasks within the voting process is a fundamental argument for decentralized voting systems, e.g., for voting in a consortium of industry participants. Instead of relying on TTPs, the UP-based REV system can be fully decentralized. Since most prior work on REV does not achieve UP, they rely on a TTP or base their privacy on a secret shared between multiple trusted entities [36].

Æternum’s voting scheme provides UP and does not rely on a single trusted authority [36]. In order to achieve UP, the deployment of a decentralized PBB is required. Distributed Ledgers (DL) offer a suitable approach to deploy a Decentralized PBB (DPBB) to achieve such a distribution of trust among authorities [29]. DLs are tamper-proof and immutable, replicate data securely, making them ideal for storing ballots that should not be manipulated by anyone [30].

This paper proposes Æternum, the first DL-based REV system combining a voting scheme with UP and a public permissioned DL, which serves as a DPBB. Æternum’s voting scheme is based on Non-Interactive Zero-Knowledge Proofs (NIZKP) [36], UP holds under the assumption of the existence of a PBB and an anonymous channel [37]. Furthermore, the trust assumptions do not impact UP, but are fundamental for verifiability and correctness i.e., trusted authorities are required for fairness. For instance, homomorphic threshold encryption is used to disallow trend analysis during an open voting period. Computational intractability assumptions are only relevant during the voting period to prevent the creation of invalid votes [36].

The remainder of the paper is organized as follows. While essential definitions are presented in Section II, Section III discusses Æternum’s system design followed by the implementation details in Section IV. Section V renders the discussion, Section VI outlines the security analysis and VII the evaluation. Finally, Section IX draws conclusions and outlines future work.
II. DEFINITIONS

Prior work defined and refined desirable privacy and verifiability properties for REV systems. Verifiability notions for voting schemes are defined as Individual [27], Universal [42] and End-to-End [6], plus the Eligibility Verifiability [32]. Ballot Privacy (BP) [27] can be defined with different assumptions being further categorized into basic BP, everlasting BP [39], perfect BP [28], and Unconditional Privacy (UP) [36].

Definition 1 (Everlasting BP (EP)). A voting system with EP does not allow even a computationally unbounded coalition of voters or outside observers to determine for whom a voter voted [39].

EP removes the assumption of computational intractability [39], but still requires trust in Voting Authorities (VA). The VA is an entity responsible for the setup and execution of an election or poll. Perfect BP is defined as a voting system that does not allow a computationally bounded entity to determine for whom a voter voted [28].

Definition 2 (Unconditional BP (UP)). A voting system with UP does not allow anyone, neither voters, outside observers, nor VAs, be they computationally unbounded or not, to determine for whom another voter voted [36].

Under the assumption of a perfect BP, even if VAs conspire, they are unable to link votes to voters. Therefore, a voting scheme with perfect BP does not require trust in the VAs or other parties. However, this privacy property relies on computational intractability and, therefore, is not everlasting. In this case, by combining everlasting BP and perfect BP, it is possible to achieve Unconditional PB.

III. ÆTERNUM SYSTEM DESIGN

The description of (i) key assumptions of the voting scheme and (ii) of the voting scheme of Æternum itself lay the basis for achieving UP.

A. Assumptions

A REV system is complex and the approach developed enables UP. Thus, the system’s architecture impacts design decisions [12] as assumed and formulated within Table I.

| ID | Description |
|----|-------------|
| A1 | Every eligible voter is uniquely identifiable. |
| A2 | The connection between the voter client and the DPBB is performed over an anonymous channel. |
| A3 | Every voter is only allowed to cast a single vote once. |
| A4 | Initial trust in the VA to set up independent generators. |

A1 assumes the existence of an Identity Management (IdM) system, which allows VAs to authenticate users and generate a list of eligible voters ($U_{eligible}$). Depending on the use case, the IdM is performed by a TTP that is either a governmental authority for votes and elections or, in a corporate setting, an external IdM provider. A2 assumes anonymous communication channels. A3 is given, i.e., by the current Swiss regulation, which does not allow for multiple votes [11]. A4 refers to cryptographic generators that need to be set up by the VA.

Fig. 1: The Four Protocols of the UP Scheme [37], [36]

B. Voting Scheme

Æternum utilizes the voting scheme from [37] (cf. Fig. 1), which does not impose any specific encoding for ballots, since they are not encrypted. It contains four protocols and the main stakeholders are VAs, voters, and the DPBB. The following notations are used to specify the major basic constructs, while the full protocol’s details are available in the source code [40] and [37] due to space restrictions.

1) Registration Protocol (Voter): First, (1) a voter $V$ picks the private credentials $\alpha, \beta$ at random from $Z_q$ and is able to (2) compute a public credential $u = h_1^\alpha h_2^\beta$. $Z_q$ is the set of integers modulo $q$. In the third step, (3) the public credentials $u$ are posted to the DPBB, registering $V$ for the vote [36].

2) Preparation protocol (Voting Authority): First, (1) VA verifies the eligibility of the registered voters and compiles a list ($U_{eligible}$), which links eligible voters to their public credential $u_i$, where $N$ is the electorate size. As mentioned in assumption A1, a suitable IdM is required, which allows the VA to verify a voter’s eligibility. Second, (2) from the set of public credentials, a polynomial $P(X) = \prod_{i=1}^{N} X - u_i$ is calculated. $P(X)$ is required for the proofs generated later in the protocol. The polynomial’s coefficients $A = (a_0, \ldots, a_N)$ are posted to the DPBB along with the list of voters $U$. Anyone can verify the correctness of the polynomial by recalculating it from $U$. Third, (3) the VA chooses an election generator $\hat{h} \in G_q$ and (4) finally posts it to the PBB.

3) Vote Casting protocol: Before starting the vote casting stage, the voter checks whether he/she is included in the list of eligible voters. If yes, the voter (1) selects a vote $v$ and (2) generates an election credential $\tilde{u} = h_3^v$. This is crucial since the voter does not authenticate with the public credential $u$, but instead, the voter sends $\tilde{u}$ with the ballot. This step actively
removes the link between the voter’s identity u from the vote and requires the voter to generate three Non-Interactive Zero-Knowledge Proofs (NIZKP) [37].

Then, two Pedersen commitments [41] are computed: (4) a commitment $c = \text{comp}_p(r, u)$ to the public credential u is created, using random $r \in_R \mathbb{Z}_p$, and also, commitment $d = \text{comp}_q(s, \alpha, \beta)$ to the private credentials is computed, with $s \in_R \mathbb{Z}_q$.

In (5), three NIZKPs are computed. First, a set membership proof $\pi_1$ is generated, proving that $c$ is a commitment to one of the credentials in the list of voters. Second, commitments $d$ and $c$ are used to create the second NIZKP $\pi_2$, proving knowledge of the private credentials $\alpha$ and $\beta$ that were used to generate $u$. Finally, the voter needs to prove that the same $\beta$ was used to generate the election credential $\hat{u}$, which was used in the generation of the public credential $u$. In this regard, $\pi_3$ uses a proof of equality of discrete logarithm, thus preventing the voter from sending multiple ballots with different $\hat{u}$ [37].

Finally, (6) the ballot tuple $B = (v, \hat{u}, c, d, \pi_1, \pi_2, \pi_3)$ is posted to the DPBB using an anonymous channel that does not allow linking its ballot to it. The vote itself can be added in plaintext to the ballot in any format. However, remember that voters need to compute NIZKPs dependent on $v$. When computing the challenge for these NIZKPs, the vote is part of the input into the hash function and impacts the computational effort. Using the vote as input ties $v$ to $B$ because it is used as input again when verifying proofs. If a different $v'$ is sent within $B$, the proof verification will fail.

### Vote Casting Protocol (Voter V)

1: Select vote $v$
2: Compute election credential $\hat{u} = \hat{h}^\beta$
3: Pick $r \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{Z}_q$
4: Compute commitments
   - $c = \text{comp}_p(r, u)$
   - $d = \text{comp}_q(s, \alpha, \beta)$
5: Compute NIZKP
   - $\pi_1 = \text{NIZKP}([u, r] : c = \text{comp}_p(r, u) \land P(u) = 0)$
   - $\pi_2 = \text{NIZKP}([u, r, s, \alpha, \beta] : c = \text{comp}_p(r, u)$
     \land $d = \text{comp}_q(s, \alpha, \beta) \land u = h_1^s h_2^\alpha \hat{h}^\beta$}
   - $\pi_3 = \text{NIZKP}([s, \alpha, \beta] : d = \text{comp}_q(s, \alpha, \beta) \land \hat{u} = \hat{h}^\beta}$
6: Post ballot $B = (c, d, v, \hat{u}, \pi_1, \pi_2, \pi_3)$ to PBB

### 4) Tallying protocol:
After the vote casting is concluded, the tallying protocol is executed. Since all data on the PBB are public, anyone can fetch all ballots $B$ (1), and then verify the proof of each ballot (2). If duplicate votes are allowed, they need to be cleaned at this point (3), e.g., by simply dropping all but the last ballot corresponding to the same $\hat{u}$, and (4), the final tally can be accumulated from the plain text votes.

### Tallying Protocol

1: Retrieve set $B$ of all ballots from PBB
2: For each $B \in B$, verify $\pi_1, \pi_2, \pi_3$
3: Detect duplicates based on $\hat{u}$ and resolve
4: Compute final election result

### IV. IMPLEMENTATION

Æternum is implemented as the up-voting-system Go module available on GitHub [40] for which packages and their dependency hierarchy are shown in Fig. 2.

Fig. 2: Source code package structure

Each of the three cli packages contain a Command Line Interface (CLI) application. The cli/pbbd package contains the entry point for the PBB application, i.e., compiling it generates an executable that can be deployed on a PBB node. The other two packages cli/vcli and cli/acli compile to a voter and a voting administration CLI application respectively. All three cli packages rely on the pbb package that implements the PBB and functionality for CLI interaction. Finally, the crypto package implements the cryptographic primitives required by the Æternum voting scheme, as described in Section III-B. The pbb package implements a Cosmos-SDK module (following a similar approach as e.g., the staking module [15]).

The details of the Æternum architecture are depicted in Fig. 3. The DPBB is composed of a DL network consisting of (i) consensus nodes and (ii) full nodes. Full nodes store a copy of the complete DL and expose web endpoints for clients to post and query. Consensus nodes additionally participate in the consensus mechanism by creating new blocks and validating transactions. In the case of an election or voting, consensus nodes should be operated by trusted authorities, which may not necessarily trust each other, while anyone can operate a full node and directly observe the voting process.

Although the prototype implementation uses Tendermint’s consensus mechanism (requiring at least $2/3$ of all consensus nodes to behave honestly), a suitable alternative is Proof-of-Authority [29], [30]. For instance, in Switzerland, cantons, down to municipalities can participate in the consensus, only requiring at least $1/2$ nodes to behave honestly.

1) Administration Client: The administration client is used by the VA (i) to generate the election configuration and post it on the DPBB, (ii) authenticating voters’ identities that were posted during the registration phase, (iii) and then add authenticated, eligible voters to the list on the PBB, (iv)
compute polynomial coefficients and post it to the DPBB, and (v) define the election generator and post it to the DPBB.

2) Voter Client: The voter client allows voters to (i) Generate private and public credentials and post the public credential to the DPBB, (ii) Check the list of eligible voters for the inclusion of their own public credential, and (iii) Generate and send all necessary parts of a ballot (election credential, commitments, ZKPs, and vote) to the DPBB.

3) Verifier Client: The verifier client allows anyone to verify the correctness of the credential polynomial and the final voting result. Verification is done by re-calculating the polynomial and the election result and comparing it to the ones published on the DPBB. The verifier client can (i) Calculate the credential polynomial from the list of eligible voters and compare it to the published polynomial, and (ii) Verify all ballots, calculate the election result and compare it to the published result. While the prototype implementation does not provide a separate verifier client, the functionality is implemented in the voter client.

4) External Systems: A suitable Identity Management system (IdM) and a CRS setup system are external systems. The IdM provides Æternum with information about the eligibility of voters (cf. Section III-A A1), while the CRS setup system is an abstraction of the mechanism needed to generate a CRS as a basis for the group generators (cf. Section III-A A6).

V. DISCUSSIONS

Previous assumptions as of Section III-A are now considered to determine relevant properties in detail.

A. Identity Management

REV schemes often make use of cryptographic signatures to authenticate encrypted ballots [27] in order to achieve eligibility verifiability. In Æternum, ballots are not cryptographically signed. If a voter would sign the plaintext ballot, BP is no longer fulfilled i.e., UP is defeated. Still, Æternum requires a suitable IdM for the registration step to verify eligibility. In the current implementation, a voter identity, i.e., a key pair, has to be created ad hoc on the voter’s machine. The public key pair would then have to be registered in an IdM system, presumably controlled by the VAs, to which the PBB has access.

During registration, the voter signs the registration transaction with the private key of the identity. The PBB can check if the signature belongs to one of the eligible voters in the IdM system. If successful, the public credential $u$ in the registration transaction is stored in the list of eligible voters. This creates a connection between voter identity and $u$ but does not pose a threat to the privacy because $u$ is not revealed when casting the vote. Therefore, any IdM system can be used with Æternum to add eligibility verification.

B. Legal Dimensions

Depending on the legal jurisdiction Æternum is deployed in, various data protection regulations, or regulations regarding REV systems have to be considered. The two major examples are the Swiss Federal Chancellery Ordinance on Electronic Voting (VEleS) [11], and on a European level, the General Data Protection Regulation (GDPR). Both pose legal challenges for Æternum and its DPBB, since the GDPR applies to anyone processing or controlling the processing of personal data [46], while the VEleS explicitly states trustworthiness assumptions, and also demands specific requirements which are currently not covered by Æternum and its voting scheme (e.g., End-to-End Verifiability).

Further, the applicability of the GDPR to Æternum depends on the use case. If the system is used in public elections, Art. 17 (3) (d) could apply, which means that the voter does not have the right to erasure because the data is needed for achieving purposes in the public interest [21]. In private use cases, the matter is less clear. The major private data connected to the voter is her identity and public credential registered in the protocol’s registration phase. Due to the nature of Æternum’s voting scheme, the DPBB does not store identifying information with the ballots i.e., the ballots may count as anonymous data according to the GDPR.

C. Unconditional Privacy

Even though the voter authenticates during the registration phase, during the voting phase, only perfectly hiding commitments are published. The commitments to $\alpha$, $\beta$ and $u$ are perfectly hiding Pedersen commitments [41]. No information about $\alpha$, $\beta$, and $u$ can be gained from the commitments, even if the discrete logarithm assumption is broken in the future. The election credential is defined as $\hat{u} = h^\beta$ and thus, if the discrete logarithm assumption breaks, $\beta$ can be calculated. Because $u = h_1^\alpha h_2^\beta$ is like a perfectly hiding commitment to $\beta$ with randomness $\alpha$, an adversary cannot decide to which $\beta$ the $\beta$ belongs without also knowing $\alpha$. Furthermore, all ZKPs are perfectly zero-knowledge i.e., no additional information can be gained from them apart from the knowledge they prove.

D. Verifiability

Æternum provides IV, since the voter can query the PBB to find out if the ballot was recorded correctly. However, this assumes that the voter’s platform is trusted, i.e., not infected with
malware corrupting the voter’s view of the PBB, and thus, not providing any voting verification mechanism [29]. Malware on the voter’s device can intercept all communications between the voter and the DPBB, thereby modifying the voter’s ballot and reflecting an incorrect state of the DPBB in order to keep the voter from noticing the modifications. Hence, IV can only be provided if we assume a trusted platform on the voter’s side, but this is impractical. A more practical solution that gives some relief to the problem is to have the voter Challenge-or-Cast the ballot with a second independent device [34]. Also note that the querying of the DPBB for a specific ballot should only happen over a non-anonymous channel (same as casting the ballot). This would compromise the link between identity of the voter and a specific ballot. Instead, the voter should either query a subset, or the full set of ballots, while using an anonymous channel.

Æternum provides UV because anyone can calculate the election result. However, because ballots cannot be linked to specific voters, it is fundamental that eligibility is ensured. This is at the core of the voting scheme, and the purpose of the proof transcripts \( \pi_1, \pi_2 \) and \( \pi_3 \), proving that the voter is (i) in the set of eligible voters, (ii) the voter possesses the private credentials behind the public credential, and (iii) that the same private credentials were used to produce the election credential. To successfully cast a fake ballot, an attacker would have to find \( \alpha \) and \( \beta \) for some \( h^\alpha h^\beta = u \in U \) which is impossible for a present adversary because of the discrete logarithm assumption. The other option is to construct a proof without knowing \( \alpha \) and \( \beta \). Still, that is impossible for a present adversary because the proofs are computationally sound.

E. Fairness

The property of Fairness in REV systems describes principle that no partial tally can be derived before the official end of the vote casting period, since an intermediate result could influence voters who have not cast their ballot yet. The voting scheme does not provide fairness by default. All votes are posted to the DPBB in plaintext, allowing anyone to calculate intermediary election results at any time in the voting period. However, fairness can be achieved by encrypting the vote with a public key provided by the VAs. The corresponding private key should be a shared secret amongst the VAs to distribute trust. Once the voting period is over, the private key can be published on the DPBB, giving everyone the ability to decrypt the votes. Note that the distributed trust needed for fairness does not affect UP. Alternatively, Time-lock encryption could be employed, allowing the encryption to be locked for a specific time-period, without the need for distributed key generation [35].

VI. Security Analysis

Only a detailed security analysis, including Æternum’s threat model, an analysis of the protocol parameters, and their security level can reveal risks and threat events. Due to space restrictions the threat model and security analysis focuses on the most relevant aspects of Æternum.

A. Threat Model

The adversaries are differentiated into a present and a future adversary. A global and unbounded adversary is not considered here but considered in [23]. The threat model is congruent with one of the utilized voting scheme [36]. The present adversary is active during the voting period i.e., at the time an election occurs. A present adversary is bounded by today’s computational resources available. Thus, it is assumed that the present adversary is computationally bound. For instance, the adversary cannot efficiently factorize primes, and thus cryptographic primitives based on prime factorization are considered secure. In contrast, the future adversary has access to future, much more powerful computational resources i.e., may have access to a Quantum Computer (QC) [23], which will make today’s infeasible problems possible to solve.

Æternum provides UP against future adversaries. However, the verifiability properties during a voting period rely on the hardness of finding \( \alpha \) and \( \beta \) for some \( h^\alpha h^\beta = u \in U \) which is impossible for a present adversary because of the discrete logarithm assumption. The other option is to construct a proof without knowing \( \alpha \) and \( \beta \). Still, that is impossible for a present adversary because the proofs are computationally sound.

B. Protocol Parameters

Æternum’s voting scheme relies on the definition of the two groups \( G_p, G_q \), and their generators, which have to be defined and published on the DPBB before execution of the registration protocol. The VA can publish the generators as long as it is guaranteed that the relative discrete logarithms of the generators are not known to anyone i.e., the generators are independent, otherwise the Pedersen commitments are not binding. For example, by knowing the relative discrete logarithm \( x = \log_{g_1}(g_2) \), one can come up with an alternative opening \( (\alpha', \beta') \) for commitment \( \text{com}_p(\alpha, \beta) = g_1^\alpha g_2^\beta \) by replacing \( g_2 \) with \( g_1^{\gamma} \), giving \( \text{com}_p(\alpha, \beta) = g_1^\alpha (g_1^{\gamma})^\beta = g_1^\alpha g_1^{\gamma \beta} = g_1^{\alpha + \gamma \beta} \) and then finding \( \alpha' + \gamma \beta' = \alpha + x \beta \). For instance, an adversary breaking this assumption is able to cast multiple valid votes with only one registered public credential, because it would be possible to generate multiple valid private credentials corresponding to that public credential.

Therefore, Æternum does not require trust in a single party, but generators have to be instantiated in a distributed manner. A possible solution is based on a Common Reference String (CRS) from which generators can be derived. However, producing a CRS without trust in one entity introduces more complexity, as can be seen at the example of a multi-party protocol being designed for the setup of a CRS for ZCash [8]. Nonetheless, such a multi-party ceremony might be necessary for a voting system to fully achieve its privacy properties.

C. Security Level

Æternum’s verifiability depends on computational intractability only during an on-going election. Thus, protocol parameters have to be configured such that a current adversary is unable to break the scheme during the vote casting period. In case of weak parameters an adversary can find \( \alpha' \) and \( \beta' \) such
that for some public credential of an eligible voter $u = g_1^{\alpha} g_2^{\beta}$ [36]. With these credentials, the attacker can post a ballot, i.e., produce valid NIZKPs like any other eligible voter. Therefore, the modulus and order of the groups $\mathbb{G}_p$ and $\mathbb{G}_q$ have to be chosen large enough to thwart such attacks. Also, the security parameter $\kappa$ used in proof $\pi$ needs to be large enough to guarantee the soundness of the proof, i.e., the prover must not be able to make a verifier accept a wrong statement.

### TABLE II: Cryptoperiods’ Security Level Recommendations

| Cryptoperiod      | Security Level | Group Order | Group Modulus | $\kappa$ |
|-------------------|----------------|-------------|---------------|---------|
| Legacy use        | 80             | 160         | 1,024         | 80      |
| 2019 - 2030       | 112            | 224         | 2,048         | 112     |
| beyond 2030       | 128            | 256         | 3,092         | 128     |

In Table II, the recommendations of the National Institute of Standards and Technology (NIST) [24], compare different choices for the security parameter $\kappa$ and the groups’ order and modulus. Numbers in the order and modulus columns are denoted in bit lengths. E.g., for a security level of 80, the group modulus should be a prime of 1024 bit length. The order of the group $\mathbb{G}_q$ determines the security of the private credentials $\alpha, \beta$. i.e., the size of the order is the size of the keyspace for the credentials. Therefore the order’s bit size is derived from recommendations on key size for DL-based cryptosystems [24]. A security level of 80 is not considered secure anymore by NIST [3]. However, because Æternum only assumes computational intractability for the time of an ongoing election, a security level of 80 is probably still enough at the time of writing. Otherwise, NIST recommends using a security level of at least 112 starting from 2019.

### D. Timing of Proof Verification

The ballots’ NIZKPs are either (i) verified when the ballot reaches the PBB or (ii) done after the vote casting phase is concluded, and the DPBB stores all incoming ballots without checking the proofs. In (i), the proof verification’s computational intensity puts a heavy load on the DL nodes, and the consensus mechanism Denial of service (DoS) attacks are easily performed and hard to mitigate. In (ii), anyone is allowed to post any number of invalid ballots, which allows flooding of the DPBB with invalid ballots. Partial proof verification is not a viable approach. Proofs $\pi_1$ and $\pi_2$ are both essential in checking if a ballot comes from an eligible voter and if the sender of the ballot possesses the private credentials corresponding to that voter’s public credential.

In order to mitigate flooding of invalid ballots and keep them from reaching consensus nodes, full nodes always run the proof verification on ballots before further broadcasting them through the network. If a ballot is found to be invalid, it can be dropped immediately at any node, thereby not amplifying invalid ballot flooding. Still, verification on a valid ballot has to happen at every node receiving it. Relying on the approval of only one node is not acceptable in a decentralized DL setting.

### E. Casting Multiple Ballots

The voting scheme [36] allows voters to cast multiple ballots that are resolved either immediately or later in the tallying phase. The current implementation of Æternum does not allow for this to prevent replay attacks by checking, whether the ballot’s election credentials were already used in another ballot. Thus, the PBB does not have to run through the full proof verification. It is guaranteed that the eligible voter will be the first to use the election credential $\hat{u} = h^2$, because only she knows the private credential $\beta$. Only after a ballot with that $\hat{u}$ has been posted an attacker will be able to replay ballots with the same $\hat{u}$.

### F. Insecure Voter Platform

A crucial aspect of REV systems is considering the voter devices and the property of Cast-as-Intended Verifiability [31]. A compromised voter client presents various threat events caused by malware (i) blocking vote broadcasting (and not posting to the DPBB), (ii) changing the vote selection (iii) breaking ballot secrecy by leaking the voter’s choice and identity. While threat events (i) and (iii) are possible with malware outside of the voter client application, threat event (ii) requires that the attacker successfully modified the voter client executable itself. Threat events (i) and (ii) can both be detected by verifying the DPBB, which requires the voter to use a second device to connect to the DPBB. For threat event (iii), no countermeasures exist in the voting scheme of Æternum. Once a voting client device is compromised, the attacker can link any ballot produced with that device to the owner. Protocol extensions presented in [36] can mitigate the problem but is not yet part of Æternum.

The trust assumptions of Æternum complicate the practical deployment of voter client applications. If the voters accept an application distributed by the VA, blindly trusting that software client’s integrity is not optimal, even if it was built from an open-source code base. Verification mechanisms need to be set up that enable anyone to build the voter client and compare the binary to the application distributed by the VA.

### G. Malicious Network Participants

Even though the current consensus of the DPBB requires at least two-thirds of all consensus nodes to be honest (i.e., is BFT-based), individual malicious nodes can cause disruptions. For instance, a malicious node can censor incoming transactions. However, due to the anonymous channel voters are not identifiable and cannot be targeted individually. Still, in the registration phase, a node can effectively block a specific voter’s attempt to register by dropping transactions received from that voter. As a simple mitigation the voter’s client should verify the correct inclusion of the registration transaction. Another possible censorship is to provide false information. A DPBB node can, e.g., respond with false protocol parameters to a voter’s requests. In consequence, the voter will create her credentials based on false groups and generators. In the best case, these credentials will be rejected when posted to the DPBB and, in the worst case, might go unnoticed, if the
public credential is also an element of the correct group. In the latter case, the voter will think she registered successfully, but will not be able to generate a valid ballot in the casting phase, excluding her from the election.

However, these issues can be solved by running a local full-node of the DPBB, which implements censorship prevention mechanisms as well as query result verification mechanisms. Censorship resistance is achieved by choosing multiple peers over a so-called peer selection algorithm and sending the transaction to each selected peer. Queries, on the other hand, are executed against the locally kept DL state, and thus no trust in remote nodes is needed. The drawback of running a full-node instead of connecting to a remote node is increasing resource (disk storage, bandwidth, CPU) requirements. Many BCs and DLs provide so-called light nodes, which considerably reduce these requirements, while still providing high censorship resistance and query result correctness guarantees.

VII. PERFORMANCE ANALYSIS

The ballot transaction is the central transaction for $\text{Æternum}$ because it contains the vote. The computations related to the ZKPs affect both the voter client and the DPBB. The voting client generates the NIZKPs, and the DPBB verifies them. Although the voting scheme [37] does not specify that the proof transcripts have to be verified by the PBB, validating transactions avoids the PBB getting stuffed with invalid ballots. The performance of the NIZKP generation (cf. Table III) and verification (cf. Table IV) was evaluated with three different scenarios (averaged over 100 executions) with varying electorate sizes $N$ of 100, 1,000, and 10,000. Those numbers were selected considering the typical distribution of citizens within cantons in Switzerland, with an average number of residents per municipality being 3,880 [10]. In each scenario, the required number of voter credentials were created and registered on the DPBB. Then, the credential polynomial was fetched from the DPBB, and ballots were created for each voter. The protocol parameters were chosen to provide a security level of 80 (see Section VI-C). For performance measurements, two consensus nodes, one for the voter and one for the VA, were deployed as Docker containers. Voter and VA clients were launched on the same machine for all transactions. The performance tests were run on a virtual machine with eight 2.4 GHz Intel Xeon E312xx CPUs, 16 GB RAM, running on Ubuntu 18.04.4. Tendermint recommends at least 2GB RAM and a 2 GHz CPU with two cores.

| Electorate Size $N$ | $\pi_1$ | $\pi_2$ | $\pi_3$ | Total |
|---------------------|---------|---------|---------|-------|
| 100                 | 105     | 513     | 1.66    | 619.66|
| 1,000               | 292     | 581     | 1.05    | 874.65|
| 10,000              | 1,574   | 579     | 1.63    | 2,154.63|

Table III depicts the average time for the NIZKP generation per voter. Proof $\pi_1$ is the set membership proof, $\pi_2$ is the proof of known representations (of committed value) and $\pi_3$ is the proof of known equality (of discrete logarithms). Effort spent on $\pi_3$ is negligible, whereas proof $\pi_2$ takes a constant amount of time, largely unaffected by the increased electorate size. As mentioned, $\pi_2$ is only dependent on the security parameter $\kappa$. Increasing the security level of $\pi_2$ by increasing $\kappa$ will make the time needed for proof generation grow linearly with $\kappa$.

In Table’s III column Total, results for the proof generation of all three proofs are compared to the results achieved in [37]. There, the proof generation took 1.4, 1.6, and 3 seconds for an electorate of 100, 1,000, and 10,000, respectively. Both UniCrypt and $\text{Æternum}$ do not apply any specific optimizations to the proof generation. In [5], an optimized version of the proof $\pi_1$ is evaluated, which achieves significantly lower execution times. However, the evaluation is based on a smaller order $p$ for $\mathbb{G}_p$ of 256 bit and a larger modulus $o$ of 1,536 bit, compared to 1,024 and 1,034 bit with $\text{Æternum}$’s evaluation.

VIII. RELATED WORK

REV systems were used in various electoral processes around the world e.g., Estonia [20], Switzerland [43], Australia [26] and Norway [45]. Due to conflicting privacy and verifiability requirements [27], none of the existing voting systems managed to solve all challenges in REV [7]. Most systems focus on verifiability and do not consider Everlasting BP as main concern [23]. REV systems use cryptographic instruments to achieve the required properties [27].

| TABLE V: Comparison of Selected Related Work |
|---------------------------------------------|
| HELIOS [2] | PROVOTUM 2.0 [30] | KOINONIA [23] | ÆTERNUM [40] |
| Decentralized PBB | ✓ | ✓ | ✓ |
| Unconditional BP | ✓ | ✓ | ✓ |

As shown in Tab. V, only Provotum 2.0 [30] also provides a DPBB, while Helios and Koinonia use centralized servers as a PBB. In [17], an enhancement to Helios [2] is proposed, where a second channel between the voter and the Helios server is introduced. The encrypted ballot is not published publicly anymore, but only available to the Helios server. Instead, a second message with a Pedersen commitment of the ballot is published. Therefore, Everlasting BP is based on the fact that an adversary does not have access to the encrypted ballots anymore. In [16], Consistent Commitment Encryption is proposed, which allows the VA to derive commitments from encrypted ballots. Similar to [17], only the commitments are publicly available, while the encrypted ballots are only visible to the VA. Everlasting BP is provided to the public, but not to the VA. In [9], the approach of Helios is combined...
with a mix-net, removing the voter identities from the votes. The difference from a usual mix-net approach is that two separate mixes are performed simultaneously, a private one that shuffles the encrypted ballots and a public one that shuffles commitments to those ballots.

A first scheme providing Unconditional BP is [28]. The scheme is self-tallying and only intended for small boardroom elections because it is data intensive. Every voter needs to generate and post data of size linear to the number of participating voters. In [25], a more efficient approach is proposed. However, voters need to take turns casting their ballots, each taking the current state of the election. For this scheme to work, each voter has to use the public keys for all other voters, which limits the protocol’s scalability. Also, both [28] and [25] depend on the discrete logarithm assumption for their BP. Therefore, they do not provide everlasting BP and do not reach unconditional BP.

In order to achieve UP in Æternum, an Anonymous Channel (AC) is required, i.e., vote casting and vote querying require an AC. Even though the ballot does not leak any information about the voter, an adversary can perform Traffic Analysis (TA) on the network during vote casting or querying. The privacy of the communication between the voter and the DPBB is crucial, and suitable solutions require evaluation. In Table VI, an overview of AC protocols is provided [44].

| Routing Class   | Adversary       | Challenges                        |
|-----------------|-----------------|-----------------------------------|
| Mixnet          | Global, Active  | TA (e.g., flooding)               |
| Onion Routing   | Local, Active   | TA (e.g., timing)                 |
| Random Walks    | Local, Active   | Partitioning Attacks              |
| DCNet           | Global, Passive | Collision and Disruption          |

MNs were introduced in [13] and aggregate input messages into batches, shuffle these messages, and output them in a reshuffled form, which removes the relation between input and output messages, disabling the ability to correlate messages [44]. MNs are vulnerable to collusion (of mixers) and to flooding attacks (e.g., if there are not enough honest users). Resilience against TA comes at the price of high latency, which makes them suitable for application in electronic mail [13].

An onion routing network is formed by Onion Routers (OR), which form a bidirectional channel (so-called circuit) between Users, which choose an ordered sequence of ORs and establish a circuit. Then, users encrypt the data in a layered fashion, and each OR in the circuit can decrypt their corresponding layer, forwarding to the next OR. Onion routing protocols, such as Tor [18] have low computational overhead and are thus suitable for low-latency applications (e.g., Web browsing) [44]. However, OR is only considered secure against local, active adversaries and is vulnerable to TA attacks [4]. Additionally, routing classes are Random Walks based on Distributed Hash Tables (DHT), as well as Dining Cryptographers Networks (DCNets). Depending on the threat model (e.g., global or local adversary) for the specific use case where Æternum is applied, an appropriate anonymous channel needs to be selected.

IX. CONCLUSIONS AND FUTURE WORK

This paper introduced Æternum, a decentralized voting system, which does not assume any computational intractability and does not require a Trusted Third Party (TTP) or central trusted authority.

Concluding, Æternum achieves Unconditional Privacy (UP), secured against future adversaries. While other voting schemes’ privacy is vulnerable to future adversaries, Æternum provides UP by utilizing the voting scheme from [36]. Æternum only requires strong parameters for verifiability during the voting period. Additionally, Æternum utilizes a Decentralized Public Bulletin Board (DPBB), which is implemented as a public permissioned Distributed Ledger (DL). A publicly readable DL enables anyone to verify the DL’s election data (i.e., the NIZKPs) and compute the final tally. Further, the DL-based architecture allows for a transparent selection of authorized Voting Authorities (VA) (e.g., authorities or consortia members), which participate in the DL network consensus.

Furthermore, measurements indicate that the main performance bottleneck of Æternum is the computational effort required to generate and verify the ballots’ NIZKPs ($\pi_1$, $\pi_2$, $\pi_3$). Thus, for nation-wide voting and elections, VAs serve as a TTP, and Æternum’s trust assumptions may not be needed. Nevertheless, for use-cases without such structures established and without a central, trusted VA, Æternum provides a suitable alternative. A set of stakeholders can set up and run an election without requiring trust in each other for ballot secrecy.

E.g., an open-source community with the need for community decisions can invite community members to apply as consensus nodes for the DPBB. With a sufficient amount of independent participants, the DPBB’s safety and integrity are assured. The same participants can take part in the distributed setup of protocol parameters. Although a prominent actor is required to initialize the DPBB and set the agreed upon parameters, all actions made by that actor are publicly visible and malicious behavior is clearly detectable.

Possible future work includes (i) the integration of a suitable anonymous channel, which could be executed within Æternum’s DPBB network directly, and (ii) the design of a suitable Multi-Party Computation for the CRS setup. Furthermore, (iii) performance optimizations of the set membership proof [5] need to be considered for large electorates (e.g., country-wide deployments). Also, (iv), future work includes the extension of Æternum with Receipt-Freeness, as proposed in [38]. Lastly, the current implementation’s availability is not bolstered against DoS attacks, and mitigations are possible future work. (e.g., Only allowing a single, cryptographically signed vote transaction per $\hat{a}$).

ACKNOWLEDGEMENTS

This paper was supported partially by (a) the University of Zürich UZH, Switzerland, and (b) the European Union’s Horizon 2020 Research and Innovation Program under Grant Agreement No. 830927, the CONCORDIA project.
REFERENCES

[1] B. Adida, “Github: Helios Voting Server.” [Online]. Available: https://github.com/benadida/helios-server

[2] ———, “Helios: Web-based open-audit voting,” in Proceedings of the 17th USENIX Security Symposium, 2008.

[3] E. Barker, “Recommendation for Key Management – Part 1: General,” pp. 1–142, 2016. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57/tr14.pdf

[4] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, “Low-resource routing attacks against Tor,” WPES’07 - Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society, pp. 11–20, 2007.

[5] S. Bayar and J. Groth, “Zero-knowledge argument for polynomial evaluation with application to blacklists,” Lecture Notes in Computer Science, Vol. 7881 LNCS, pp. 646–663, 2013.

[6] J. Benaloh, R. Rivest, P. Ryan, P. Stark, and V. Teague, “To End-to-end verifiability,” 2015.

[7] D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi, “Adapting helios for provable ballot privacy,” Lecture Notes in Computer Science, Vol. 6879 LNCS, pp. 335–354, 2011.

[8] S. Bowe, A. Gabizon, and M. D. Green, “A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK,” Lecture Notes in Computer Science, Vol. 10958 LNCS, pp. 64–77, 2019.

[9] J. Buchmann, D. Demirel, and J. V. D. Graaf, “Towards a Publicly-Verifiable Mix-Net Providing: Financial Cryptography and Data Security: FC 2013, Lecture Notes in Computer Science, Vol. 7859, pp. 197–204, 2013.

[10] Bundesamt for Statistik, “Regionalportraits 2020: Kennzahlen aller Gemeinden,” 2020. [Online]. Available: https://bch-ev.ch/bfs-gemeinde

[11] S. Bundeskanzlei, “Verordnung der BK über die elektronische Stimmabgabe (VEleS),” 2013. [Online]. Available: https://www.admin.gemeinden.ch/ofp/opc/de/classified-compilation/20132343/index.html

[12] J. Benaloh, R. Rivest, P. Ryan, P. Stark, and V. Teague, “To End-to-end verifiability,” 2015.

[13] D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi, “Adapting helios for provable ballot privacy,” Lecture Notes in Computer Science, Vol. 6879 LNCS, pp. 335–354, 2011.

[14] S. Bowe, A. Gabizon, and M. D. Green, “A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK,” Lecture Notes in Computer Science, Vol. 10958 LNCS, pp. 64–77, 2019.

[15] J. Buchmann, D. Demirel, and J. V. D. Graaf, “Towards a Publicly-Verifiable Mix-Net Providing: Financial Cryptography and Data Security: FC 2013, Lecture Notes in Computer Science, Vol. 7859, pp. 197–204, 2013.

[16] Bundesamt für Statistik, “Regionalportraits 2020: Kennzahlen aller Gemeinden,” 2020. [Online]. Available: https://bch-ev.ch/bfs-gemeinde

[17] S. Bundeskanzlei, “Verordnung der BK über die elektronische Stimmabgabe (VEleS),” 2013. [Online]. Available: https://www.admin.gemeinden.ch/ofp/opc/de/classified-compilation/20132343/index.html

[18] J. E. Burge and D. C. Brown, “Software Engineering Using COSMOS SDK, “Documentation: Staking Module.” [Online]. Available: https://docs.cosmos.network/master/modules/staking/

[19] H. Ge, S. Y. Chau, V. E. Gonsalves, H. Li, T. Wang, X. Zou, and N. Li, “Koinonia: Verifiable E-Voting with Long-term Privacy,” pp. 270–285, 2018.

[20] D. Giry and BlueKrypt, “Cryptographic Key Length Recommendation,” 2020. [Online]. Available: https://www.keylength.com/

[21] J. Groth, “Efficient maximal privacy in boardroom voting and anonymous broadcast,” Lecture Notes in Computer Science, Vol. 3110, pp. 90–104, 2004.