Trustless, scalable censorship-resistant anonymous yet verifiable voting system

Submitted to Lykke Streams - Research

Jonatan Bergquist

December 17, 2017

This is a proposal for how a trustless, scalable, censorship-resistant anonymous voting system could be implemented using verifiable claims on Hyperledger Indy. It is submitted as a research proposal to Lykke Streams. General concepts such as blockchain and basics of cryptography are assumed to be familiar to the reader. A very brief overview of basic concepts relevant for the specifics of Indy are given. Thereafter, a voting system is described based on anonymous credentials, revocation lists and verifiable claims. Finally, 10 questions that were given by the authors of the request, are responded to.

1 Introduction

Blockchain technology is, generally speaking, well-suited for e-voting since it allows for a transparent, immutable way of registering votes, as well as built-in public key-cryptography. However, in most blockchains, issuing new addresses is either very easy and is encouraged (Bitcoin and Ethereum), or is enforced (IOTA, Monero). This means that tying the right to vote to a physical person becomes difficult as the connection person-address can be one-to-many and not one-to-one. Existing blockchain-based solutions include having a trusted third party that verifies that a voter is authenticated to vote [LJGEJK16], requiring a person to register with the authority or using email [TT17]. Suggestions have also been made for using Ethereum as a web-bulletin-board for voting such as in [MTM16].

However, using self-sovereign identities and anonymous credential technology, allow for a user to be in full control of all personally identifiable information (relevant for GDPR as of May 2018) and allows the user to prove a property of the identity (such as the right to vote) without revealing any other information about the identity. Below is a brief description of how this could be implemented using open-source technology followed by explanatory responses to questions given by the publisher.

2 Voting systems based on verifiable claims and revocation lists

2.1 Infrastructure

The suggested blockchain for this voting system is called Sovrin [1] and is a public, permissioned ledger. The core of Sovrin has been included in the Hyperledger Foundation under

[1] https://sovrin.org/, 2017-12-13

the name Indy [2]. Important to know about Sovrin is that it makes use of so-called DID's, or Decentralized Identifiers (defined in section 3 of RFC 4122), and CID's, or Cryptographic Identifiers (defined in section 13.6.2 (Cryptographic Identifiers) of the OASIS XDI Core 1.0 specification). The identities allows for both users who can currently manage their own keys and those who cannot, creating an inclusive and unique identification system. The decentralized public key infrastructure implemented by Sovrin allows for the unlinkability of identifiers, keys and identity owners. Furthermore, identity attributes, credentials, are *unforgeable* and can be *delegated*. Each so-called *agent endpoint*, is a service provided by an agent (a client server) that can be seen as a unique interface for an identity owner with the ledger. Thereby an identity can prove to be in possession of a specific attribute (claim), only relevant to the specific endpoint. This also makes it impossible to correlate different endpoints with claims that may, in reality be connected to an identity owner. Specifically in this case, an identity owner may not wish to reveal to a tallying authority her age, but just the right to vote on a specific issue. [RLH16]

2.2 Authentication

For this voting system, we assume three types of users: Issuer, Prover (the voter) and a Verifier service. The functioning is very similar to that of the anonymous credentials described in [Kho] with the addition of the application of the revocation list as a voting tool. The Issuer (authority or community wishing to initiate a vote) issues a credential C_1, based on an identity A, asserting a property V_1 (1st condition for the right to vote) about A. Now the Prover, the voter, can use this credential along with an additional credentials that might be needed for voting, such as age or residency limitation, C_2 to create a proof for the Verifier. More specifically, the Prover proves that a single, master secret is connected to two credentials C_1 and C_2. In our case, C_2 would be a credential for a property V_2 corresponding to a *non-revocation claim*, proving that the voting right of the identity owner has not been withdrawn. The Prover constructs and sends the proof of the claim pair (C_1, C_2) to a Verifier who can verify these claims using publicly available information from the ledger, without having to interact with any of the Issuers.

The revocation list is published by the Issuer, or rather a cryptographic scheme called an *accumulator*, is published, and the Prover can give a short proof that she is not part of it.

2.3 Voting

Voting requires two actions to be initiated by the Prover:

1. Making a request to the Issuer to be added to a revocation list representing that the Prover has voted. The revocation list consists of a dynamic accumulator and a set of Zero Knowledge Proofs. The dynamic accumulator proposed in [CL02] has many practical properties for revocation lists and anonymous credentials, as explored in the article. Importantly, it is allows for the efficient proof production and verification of group membership. Thereby, a voter can verify that his or her vote is being counted without interaction with a tallying authority. The vote is not tied to anything more than an anonymous credential belonging to an identifier.

2. Publication of the Prover's vote on the public ledger. The vote is signed by the Prover and contains a verifiable claim that the Prover has the right to vote and that the vote has not yet been used. This allows for the public and transparent tallying of votes.

[2] https://github.com/hyperledger/indy-sdk/, 2017-12-13

2.4 Tallying

Tallying is done by the a public count of the votes published on the ledger, along with a verification of claims. There are a few different strategies for verifying votes; the most costly and time-consuming one is the one where all votes are verified, by everyone, another option would be for the verifications to be outsourced randomly to the network, meaning that any peer can be assigned to verify any votes.

2.5 Potential weaknesses of suggested solution

A potential threat is a the existence of a vulnerability in the app used for voting. If the app would be corrupted, the key used for the signing of the vote could be phished and be used to withdraw a vote and re-vote for something else, or to expose the identity owner.

Another weakness of the proposed setting is in relying on another party (such as a government or another KYC-performing application) to verify claims that the voter-registration might rely on. This is of-course dependant on the type of vote one wishes to perform.

Finally, a weakness could be the relative immaturity of the development of Indy and the yet-to-come adoption in institutions. To circumvent this, applications can only be built that use the types of claims that can be issued by already participating Issuers.

2.6 Responses to questions from project publisher

1. Which blockchain should the voting system be based on?

   Sovrin or Hyperledger-Indy.

2. How does the voting start and end?

   See section 2.3.

3. How do the voters obtain the voting power?

   The party initiating the vote needs to set the criteria for who qualifies for a vote. After making those criteria public, a voter only needs to publish a Zero Knowledge Proof for a verifiable claim proving that the criteria are fulfilled.

4. How do the voters cast their votes?

   See section 2.3.

5. How do the voters check that their votes have been counted correctly?

   This can be verified by the voter/Prover, through the use of dynamic accumulators in the revocation lists. This in combination with the publication on the public ledger of votes allows for the verification of the outcome of any member.

6. How do the interested parties independently calculate/verify the voting results?

   See section 2.4

7. How can the anonymity be preserved if needed?

   Anonymity is achieved by the use of anonymous credentials and verifiable claims.

8. How scalable is the proposed solution?

   Technically, scalability in a blockchain is very much dependant on the consensus protocol. Indy/Sovrin uses a consensus protocol called Plenum*, which builds upon the* Redundant Byzantine Fault Tolerance (RBFT) *protocol by [AMQ13]. Plenum*

Table 1: Time-consumption for the voting process for different sizes of participants.

	10k	100k	1m	10m
Registration [s]	10	100	1,000	10,000
Counting of votes [s]	60-240	60-240	60-240	60-240
Verification of votes [s]	5000	50k	500k	5m
Total [h]	∼1.5	∼14	∼149	∼1400

extends it in a number of ways[3]*, e.g. by including the implementation of black-listing identities in the protocol. The more validators that participate in the network, the more faulty nodes the protocol can tolerate. But it also grows slower with the amount of validator nodes. Therefore, a large application would require multiple consensus pools of validators. Each pool will then have their own ledger, the separate ledgers will be synchronized independently and asynchronously. Transactions (client requests) take, in the order of magnitude of, milliseconds and consensus is reached in around a second.*

9. What are the expected costs of running a massive voting (10k - 100k - 1m - 10m voters)? *"Assuming each system has dual core CPU, 4GB RAM and 30 GB disk, cost on AWS can be $4000- $5000 for one year for a 16 node system." - Quote by J. Law, T. Ruff and D. Reed from Evernym [LLP].*

10. What is the expected time frame for running a massive voting (10k - 100k - 1m - 10m voters)?

    Assuming that"time frame" consists of the following parts: voting and tallying of votes, the following calculation can give an estimate of how long a vote could take. For one vote to be registered, ca. 1 ms is required. After 1 s the network is in consensus. This means, naively 1000 votes could be registered per second, implying that the voting times for the group sizes mentioned above would be: 10s , 100 s, 1000 s = 16 min 40 s, 10,000 s = 2 hours 46 min 40 s. The tallying of votes consists of counting and verifying the votes. A reasonable assumption would be that counting the votes does not take more than a few minutes and is a one-time operation regardless of the size of the vote, since the multiple consensus pools required for scalability update and synchronize independently ans asynchronously. The verification of the votes can be done offline without access to the ledger and consists of verifying a signature. According to [BKND] the verification of one Elliptic Curve Digital Signature Algorithm (ECDSA)-signature of 256-bit keys (the specific version used in Indy is Ed25519) takes about half a second. This would mean that the complete voting process for the different sizes mentioned can be seen in Table 1. Although, one must consider that those computations were done in 2004 on a 50-Hz, 32-bit ARM. With multiple processors, this can of course be done much faster.

References

[AMQ13] P. L. Aublin, S. B. Mokhtar, and V. Quéma. Rbft: Redundant byzantine fault tolerance. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 297–306, July 2013.

[3] More at https://github.com/WebOfTrustInfo/ID2020DesignWorkshop/blob/master/topics-and-advance-readings/scaling-a-bft-consensus-protocol-for-identity.md

[BKND] Amit S. Bhala, Vivek Kshirsagar, Meghana Nagori, and Mandar K. Deshmukh. Performance comparison of elliptical curve and rsa digital signature on arm7.

[CL02] Jan Camenisch and Anna Lysyanskaya. *Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials*, pages 61–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[Kho] Dmitry Khovratovich. Anonymous credentials with revocation. Technical report.

[LJGEJK16] Kibin Lee, Joshua I James, Tekachew Gobena Ejeta, and Hyoung Joong Kim. Electronic voting service using block-chain. 11:123–136, June 2016.

[LLP] KPMG LLP. Consensus – immutable agreement for the internet of value. appendix 3. Technical report.

[MTM16] Patrick McCorry, Ehsan Toreini, and Maryam Mehrnezhad. Removing trusted tallying authorities. Technical report, Newcastle University, 2016.

[RLH16] Drummond Reed, Jason Law, and Daniel Hardman. The technical foundations of sovrin - a white paper from the sovrin foundation. Technical report, 2016.

[TT17] Pawel Tarasov and Hitesh Tewari. Internet voting using zcash. In *14th International Conference on Applied Computing, Lisbon*, October 2017.