Self Sovereign Identity Based E-petition Scheme

Rıdvan Karataş[1] and Isa Sertkaya[1,2]

[1]Cybersecurity Engineering, Graduate School of Natural and Applied Sciences, Marmara University, Turkey.
[2]MCS Labs & BCLabs, TÜBİTAK BİLGEM UEKAE, Kocaeli, Turkey,
e-mail: karatasrdvan@gmail.com, isa.sertkaya@tubitak.gov.tr

ORCID iD: 0000-0001-9921-9536, 0000-0002-4739-0515

Research Paper Received: 15.04.2020 Revised: 26.09.2020 Accepted: 29.12.2020

Abstract—Electronic petitions are one of the important tools used for democratic participation. Citizens can express their support or dissatisfaction with government investments or regulations. Citizens can also provide sufficient feedback to organizations or government institutions through electronic petitions. In this paper, we have designed an electronic petition scheme using blockchain as a source of trust for protecting e-petition's users' privacy and security. With our e-petition system, citizens can anonymously participate an e-petition and expresses their support or dissatisfaction with a social and political issue freely. We have used Sovrin blockchain as source of trust by using registry for Decentralised Identifiers (DIDs) and their associated public keys and communication endpoints. We have also used other Sovrin components such as verifiable credentials, proofs and agents for implementing our privacy and security preserving e-petition scheme. Lastly, we have analyzed security, privacy and performance of our e-petition system.

Keywords—electronic petition, e-petition, self sovereign identity, identity management, anonymous credentials, verifable credentials, blockchain, security, privacy.

1. Introduction

The usage of technology is increasing day by day in our daily lives. As an example, technology has become a part of democratic participation such as electronic voting. Replacing the traditional pen and paper scheme with a new election system has the potential to limit fraud while making the voting process traceable and verifiable [17].

Blockchain that is a database which is tamper-resistant and distributed and is not controlled by a single entity, but can be accessed and shared by all people has potential as a tool for implementing a new modern voting process such as blockchain based election [8].

Electronic petitions are also one of the important tools used for democratic participation. Electronic petition is a form of petition which is signed online, usually through a form on a website. Citizens can express their support or dissatisfaction with government investments. Citizens can also provide sufficient feedback to government institutions or organizations through electronic petitions. On the other hand, e-petitions may lead to security and privacy risks. Because mostly petitions deal with social and political issues such as abortion, gay marriage, immigration law, issues related to religious freedom. In other words, they may leak sensitive information about users such as individual's political ideology, religious beliefs or sexual preferences. Therefore,

any e-petition system must be designed not to allow such personal information leakages. At this step, blockchain providing suitable components can be ideal choice for designing privacy and security based electronic petitions. As known, blockchain technology can be used many different areas such as healthcare, IoT security and asset management. Recently, blockchain as a decentralized and distributed public ledger technology in peer-to-peer network has received considerable attention in identity management systems especially self-sovereign identity management systems which provides called "self-sovereign identity" which is not dependent on any online centralised registry, identity provider, or certificate authority in order to make authentication.

This paper presents an electronic petition scheme using Sovrin Ledger as source of trust by using registry for Decentralised Identifiers (DIDs) and their associated public keys and communication endpoints.

The paper is structured as follows. Section 2 includes related works. Section 3 expresses the main technologies we have used. Section 4 expresses the actors and their interactions in our petition design, privacy and security assumptions as well as requirements, protocol details and also attacker model of our petition design. Section 5 discusses evaluation. Section 6 presents privacy, security and performance analysis. Section 7 includes the implementation overview of our petition scheme and lastly section 8 includes future works.

2. Related Works

In this section, we present some blockchain based e-voting and e-petition systems that we have investigated for designing our e-petition scheme.

Agora [11] provides voting features designed for institutions and governments. After purchasing Agora token for each individual voter, governments and institutions can use Agora for voting purposes.

Netvote [1] is a decentralized blockchain-based voting network on the Ethereum blockchain. Netvote provides decentralized apps (dApps) for the users for interacting with the network. Users are divided into two groups admins and voters in Netvote. Admin users determines election policies, create ballots, establish registration rules and open and close voting by using their Admin dApp. The Voter dApp is used for for registration and voting by individual voters.

Another blockchain based e-voting solution presented in [13]. In this work, they present a internet voting protocol with maximum voter privacy using the Blockchain, called The Open Vote Network (OVN) which is written as a smart contract on the public Ethereum blockchain.

Diaz *et al.* in [5] presents a privacy preserving electronic petition using Belgian e-ID as source of authentication, and anonymous credentials used for sign petitions. In this e-petition system, petition users authenticate to credential issue server using their Belgium e-ID card and then receive required credentials to sign a petition. After obtaining credentials, users contact to petition server to sign a petition. But this work provides neither a description of the underlying protocols, nor a thorough evaluation.

Another privacy preserving e-petition system was proposed in [23]. In this work, they use the Belgian eID card in a registration procedure and they also use Idemix credential system for anonymity of a user and unlinkability of a user's signature. One of the good aspect of this e-petition system is that PetAnon allows petition organizers to provide potential signers with multiple choices. Today, many petitions only have one version: 'in favour'. But using Belgian eID card for bootstrapping the users

is not privacy preserving mechanism.

Another privacy preserving e-petition was proposed by [24]. In this work, Wahab proposed an electronic petition system which supports anonymity and unlinkability by using Coconut selective credential disclosure scheme. Coconut provides computationally efficient and short credentials. Therefore, Coconut cryptographic primitives take just a few milliseconds on average, with verification taking the longest time (10 milliseconds). Coconut provides credential scheme supporting distributed threshold issuance. But as any threshold system, Coconut is vulnerable if more than the threshold number of authorities are malicious and also adding and removing authorities is cumbersome in Coconut.

3. Technologies

This section presents main components of Sovrin Trust Framework used for implementing our e-petition scheme.

3.1 Sovrin Network

The Sovrin Network is the public-permissioned blockchain designed for self-sovereign identity for people, organizations and other things in need of identity they control and own. The Sovrin Network includes servers nodes administered by a diverse group of trusted entities called Stewards [20] responsible for operating other Sovrin nodes.

Sovrin Network consists of three layers [20].

Sovrin Ledger: This layer includes multiple public permissioned ledger.

Agents: This layer provides interfaces for communicating client and ledger.

Clients: These are distributed identity owners.

3.1.1 Sovrin Ledger

Sovrin Ledger consists of several components. These are several types of ledger, Plenum Consensus protocol and several types nodes.

Sovrin ledger consists of multiple public permissioned ledgers.

The Identity Ledger: This is the primary ledger. All identity records are written to this ledger by Sovrin user.

The Pool Ledges: This ledger provides records of node's actions like what Sovrin nodes are delete or permitted are recorded on this ledger.

The Voting Ledger: Operations of consensus process' actors(trustees and stewards) are performed on this ledger.

The Config Ledger: Network configuration transactions created by Sovrin Foundation Technical Governance Board [20] and approved by the Board of Trustees [20] are recorded on this ledger.

3.1.1.1 The Plenum Consensus Protocol

Plenum Byzantine Fault Tolerant Protocol that is a kind of Redundant Byzantine Fault Tolerance (RBFT) [3] is the distributed consensus protocol implemented in Hyperledger Indy. Plenum is one of new implementations of RBFT. In addition to RBFT, Plenum enhances a high-performance, fault-tolerant communications protocol on top of UDP called Reliable Asynchronous Event Transport (RAET) Protocol [9]. RAET leverages Daniel J. Bernstein's Curve25519 [4], a highly-secure high-performance Elliptic Curve Digital Signature Algorithm [10].

Unlike RBFT which uses Message Authentication Codes Plenum Byzantine, Plenum uses Digitally Signed messages using CurveZMQ [7].

In addition to RBFT which does not provide the election process, Plenum protocol has added voting

to selection of the primary process. This process is pluggable meaning that could be shifted easily for different security and performance requirements.

3.1.1.2 Sovrin Nodes

Two types of nodes are used in the ledger. Validator nodes and observer nodes.

Validator Nodes: Validator nodes responsible for writing new records to ledger. Validator nodes also operate the Plenum consensus protocol and add new Sovrin transactions to the ledger.

Observer Nodes: Observer nodes include a read-only copy of the Sovrin ledger [18]. Observer nodes are also necessary for scaling the network.

3.1.2 Sovrin Agents

Agents are in middle layer in Sovrin architecture. Both Sovrin clients and agents are known as clients in Sovrin architecture. But main difference between them is that Sovrin agents also serve as servers which has a addressable network endpoint.

Agents operate below operations in the Sovrin Network.

Coordination endpoints for multiple clients: From identity owner perspective, Sovrin agents are the actors of managing messages and the status across multiple Sovrin clients run on several edge devices (laptops, smartphones, etc.)

Persistent P2P messaging endpoints: Sovrin clients which does not have own endpoints run on edge devices (smartphones, laptops, etc.). These endpoints are known as network services (web servers, email servers, domain name servers, etc) are provided by Sovrin Agents.

Encrypted data storage and sharing: Sovrin Agents are capable of sharing and storing data which is encrypted and managed by the identify owner [18].

Encrypted backup of Sovrin keyrings: Sovrin agents provide encrypted backup service for recovering keys easily [18].

3.1.3 Sovrin Clients

Sovrin Clients, the last layer of Sovrin architecture, are vital for Sovrin Network.

Keychains

One of the most important tasks of the Sovrin Client is to protect and manage the identity owner's identity key chain which provides self sovereign identity for Sovrin users. Sovrin keychain architecture is very similar to the OS specific architecture. What makes Sovrin keychain special is that it is based on self sovereign identity manner completely [18] .

Local Containers

Sovrin Client can replicate portion or all of container of identity owner's data. Sovrin Client's main task here is to manage the physical storage of this identity owner data by particular operating system.

Another task of Sovrin Client is to share these identity owner's data containers stored on that particular device to other Sovrin Clients synchronously when needed.

First-Time Provisioning

Like other email server or file sharing client, Sovrin Clients must first connect to the network to perform some operation. In this step, called provisioning, the Sovrin Client is authenticated to the network.

Sovrin Client who does not yet have an identity first connects to the trust anchor [18] which adds new identities to the network.

4. SSI Based E-Petition Scheme

This section includes our e-petition design based on Sovrin self-sovereign identity framework mentioned chapter in 3.

4.1 Actors and Interactions

We have specified a single, Petition Registration Service (R) and several Petition Organizer $(O_1, O_2, \ldots, O_n)$ such as public institutions or organizations, and several Petition Users $(U_1, U_2, \ldots, U_n)$ and a several Credential Issuers $(I_1, I_2, \ldots, I_n)$. In our petition scheme, each petition user and petition organizer has one or more credentials used for signing a petition or getting a petition organizer role.

Main operations illustrated with figure 1 are performed by these actors in our petition.

E-Petiton Registration Service: Petition Registration Service has two important task. These are registering Petition Organizer and User. In order to do that, Petition Registration Service issues some credentials to Petition Organizers and Petition Users. For example, Petition Registration Service issues petition detail (Petition Organizer Credentials) credentials to Petition Organizer. A petition organizer acts as Petition Organizer role with these credential and creates several petitions. Also Petition Registration Service issues unique number for each petition to Petition User for registering users. Also this credential is used for preventing double signing. In other words, Petition Registration Service registers petition users and petition organizers by issuing some credentials to these entities.

E-Petition Credential Issuers: These actors are responsible for issuing credentials to petition users. These actors can be organizations, universities or institutions. These actors can generate their specific schema definitions and then credential definition that depends on these scheme definitions. When Petition User contacts to obtain a credential, Credential Issuer issues this credential to Petition User using Sovrin schema definition and credential definition. For example, petition user wants to participate a petition requested driver licence from petition users. In this case, petition user contacts to Driver Licence Department credential issuer. Then Driver Licence Department issues these credentials to users.

E-Petition Credential Organizer: These actors are responsible for creating petition namely proof request in Sovrin. Organizers firstly request proof from e-petition users. Petition users generate proofs by using Sovrin ledger and their credentials in their wallet.

Petition Users present their proofs to Petition Organizer namely verifier in Sovrin Trust Framework. And then Petition Organizer verifies or rejects the proofs by querying Sovrin ledger. For example, an public institutions can be e-petition organizer.

E-Petition Users: Anyone has identity number can be an-e petition signer in our design. These actors can participate and sign an e-petition. Before signing petition, e-petition users may have to hold some credentials such as over 18 year old or driver licence. Petition Users receive these credentials from Credential Issuers. Shortly, Petition Users receive these credentials from Credential Issuers and stores these credentials on their wallet. And then these credentials are used for signing step by generating proof.

For example : Disabled persons (Citizens) may participate a petition about traffic fines. In this scenario, these actor can be defined as follows.

Citizens: prover and credential holder

Driver Licence Department: credential issuer

Petition Registration Service: credential issuer

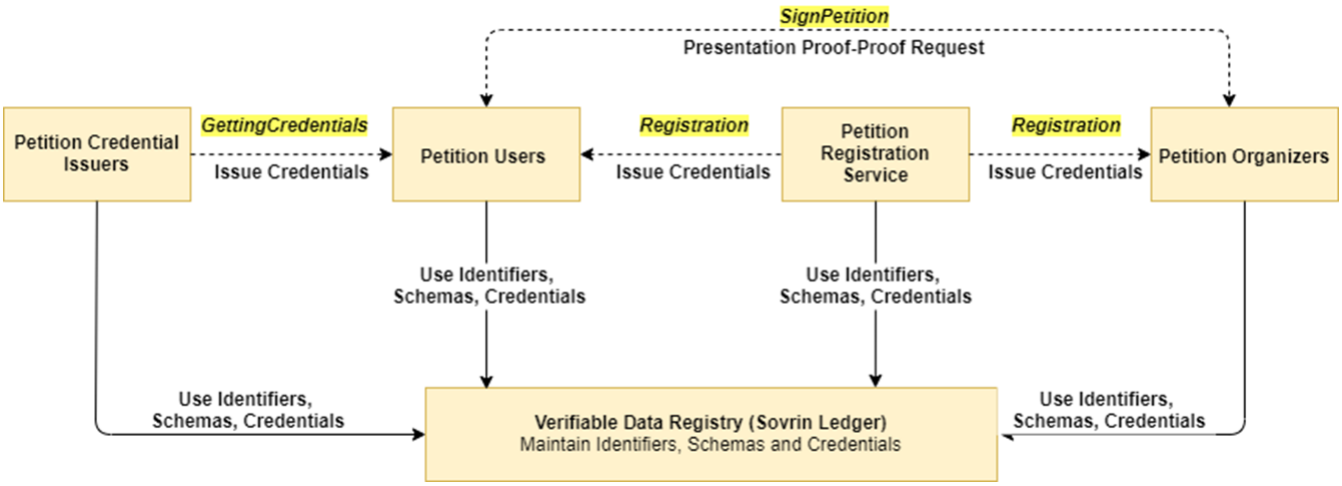

Fig. 1: SSI Based E-Petition Scheme

Petition Organizer : credential verifier.

These actors mainly perform following operations.

Registration of Petition Organizer

In this step, petition organizers are received petition information credentials from petition registration service. In other words, Petition Registration Service registers any Petition Organizer. In order to do that, Petition Registration Service issues some credentials to Petition Organizers. These information could be petition name, id, petition definition and potential petition users or conditions etc.

After getting required credentials for being Petition Organizer, Petition Organizer can create petition.

Getting Credentials from Credential Issuers

In this step, petition users obtain required petition credentials from credential issuers. For example, a petition can request driver licence from petition user. In this case, petition user interacts with driver licence department and then obtains driver licence credentials and stores in his/her wallet. Then petition user can use these credentials for creating proof, in other words, signing a petition.

Registration of Petition User

In this step, Petition Registration Service registers a petition user for participating a petition. In order to do that, Petition Registration Service issues some credentials to Petition User. In this step, Petition Users contact to Petition Registration Service for registering, in other words, for getting registration credential namely unique number credential in our system. Petition Registration Service sets this value for each user individually. This value controlled by Petition Organizer in privacy preserving way using zero knowledge proof for preventing double signing.

Signing and Counting Petition

After Petition Organizer receiving signatures (proofs) from petition users, it counts and publishes them in a way that users can check whether their votes have been counted or not.

4.2 Security and Privacy Assumptions

The following assumptions are made.

Secure connections

Connections preserve integrity and confidentiality. Each actor builds a pairwise, peer to peer secure connection by using their endpoint DIDs, pairwise pseudonymous DID and agents. In our petition design, each actor has a DID. This DID has a corresponding private agent which has own pseudonymous network address from which the an e-petition actors can exchange verifiable e-petition claims and any other data with other actor over an encrypted private channel.

We assume that this private channel is secure.

Enhancing Anonymity:

Each actor can use different DID for different connection to enhance anonymity and preventing correlation.

4.3 Protocol

The implementation of the methods in section 4.1 are discussed in more detail.

Registering Petition Organizers and Creating a Petition

Petition Organizer builds a pairwise peer to peer connection between Petition Registration Service by using their endpoint DIDs and agents. And then Registration Service creates Petition Organizer Credentials such as unique petition name for this petition organizer and then sends credentials to the organizer. After receiving these credential, Petition Organizer acts as Petition Organizer. Also Petition Registration Service sends unique number list that is used for preventing double signing and controlling registered user through this private channel. And then Petition Organizer prepares petitions known as "proof request" in Sovrin. Petition Registration Service determines required conditions (known as requested attributes and requested predicates) for any specific petition. Petition Registration Service sends these information via these private channel or other means of communications such as https etc. After that Petition Organizer creates a petition (proof request) and publishes it on site.

Registering Petition User

In this case, any Petition User contacts to Petition Registration Service for registering, in other words, getting unique number credential. Petition Users build a pairwise peer to peer secure connection with Petition Registration Service by using their endpoint DIDs and agents. Petition Registration Service sets this value for each user individually. This value controlled by Petition Organizer in privacy preserving way using zero knowledge proof for preventing double signing.

Getting Credentials from Credential Issuers

In this step, Petition Users build a pairwise, peer to peer connection with a Credential Issuer by using their endpoint DIDs and agents. Petition Users obtain required petition credentials from Credential Issuers. For example, an petition can request driver licence from petition user. In this case, Petition User interacts with driver licence department and then obtains driver licence credentials and stores in his/her wallet. Then Petition User can use these credentials for creating proof. In other words, signing a petition.

Participating a Petition

In this step, Petition Users build a pairwise, peer to peer connection with a Petition Organizer by using their endpoint DIDs and agents. A petition known as proof request in Sovrin requests some credentials with using Sovrin proof request attributes such as requested attributes or requested predicates. Petition User receives proof request, and generates proofs depends on credentials she/he obtains from Credential Issuers and Petition Registration Service. The proof contains information such that the requirement of the Petition Organizer's proof request can be satisfied. And then Petition Organizer

receives the proof from Petition User. Inside the proof, Petition Organizer sees the information and condition required, and verifies that they are coming from Credential Issuers and Petition Registration Service (for unique number). Then Petition Organizer accepts this proof.

Counting and Publishing Signatures

After Petition Organizer receiving and accepting signatures (proofs) from petition users, it counts and publishes them in a way that users can check whether their votes have been counted or not.

4.4 Functional and Threat Model

This section mentions the actors and their interactions in our e-petition scheme. We have mentioned privacy and security assumptions of our petition design in this section. And then we have presented also the attacker model and requirements of our petition design.

4.4.1 Attacker Model

Attacks can be grouped in malicious actors or revealing user signature.

Malicious actors: by a malicious Petition User, Petition Organizer or Petition Registration Service.

1 Malicious User: Malicious User participates the same petition several times by obtaining multiple credentials,
2 Malicious Organizer: Malicious Organizer tricks users to participate another petition.
3 Malicious Registration Service: A Malicious Registration Service issues faked or altered credentials.

Revealing user signature to the specific petition user's sign attribute (1) or to another signature (2).

1 To the signer's sign attribute: Actors may come together to reveal users' sign attribute.
2 To another signature: Petition Organizer can create votes of the same user on different petitions. And then Petition Organizer can reveal specific user by using user's sign value.

4.5 Requirements

This section includes our e-petition scheme requirements depend the attack model and actor interactions. We have grouped into security and privacy requirements.

Security Requirements

S1. A petition user can not participate a specific petition multiple times.
S2. A petition can request only persons with specific population. So, the petition user has to prove that he/she belongs to this population.
S3. A petition can request some qualification from petition users. User has to prove that he/she have required qualifications for specific petition.
S4. Petition issuers can revoke a user credential during participating petition time.
S5. A user can verify his/her signature information in the petition organizer database.
S6. Everyone can verify that counting of his/her signature. No one is able to delete or alter the petition records.

Privacy requirements

P1. The user's identity should not be revealed from the petition signatures. Moreover, signatures of the same user for different petitions can not be revealed.
P2. The user may or may not share some of his/her identity's attributes such as age and gender with petition organizer when petition organizer requests.

5. Evaluation

5.1 Security and Privacy Evaluation

This section includes evaluating of the our petition protocol mentioned 4.3 depends on attacker model and the requirements in the section 4.4.1.

S1. For each petition, Petition User has a unique number. Each petition requests this value using Sovrin requested_predicates feature. The actual unique number value is not revealed because this method uses zero knowledge proof. Even if the Petition Registration Service and Petition Organizer come together, they can't reveal Petition User unique number from the proof.

S2. Each user can prove whether he/she has specific credentials requested by the petition. In order to do that, Petition User obtains several credentials from different Credential Issuers. Then Petition Users use these credentials for creating proofs. For example, a petition requests driver licence using Sovrin requested_attributes and requested_predicates from Petition Users (targeted population). Petition Users have driver licence can create required proof and present to Petition Organizer. And then Petition Organizer validates these proofs using Sovrin Ledger.

S3. Petition User can prove certain personal properties requested by petition (proof request) such as age, location. Also Petition User can disclose additional properties.

S4 : We will implement this feature for our petition scheme by using Sovrin revocation registries.

S5. Each Petition User can verify his signature in the petition organizer database. Because each user knows his/her signature information, Petition Users can verify that whether his/her signature counted easily.

S6. In our petition scheme, we can not detect Petition Organizer's database actions. We assume that Petition Organizer, Credential Issuer or Petition Registration Service are trust entities. Because these entities are called trust anchor. The Sovrin Trust Framework specifies the qualifications for selection and ongoing verification of trust anchors, and thereby establish the basis for maintaining a sustainable, organic web of trust of permissions on the Sovrin network [18]. We plan to move some tasks (such as storing signatures) of Petition Organizer to the blockchain platform for benefiting its transparency, immutability and security properties. Because we assume that all actors are trusted such as Petition Organizer. But this assumption could lead to security or privacy problems. Petition Organizer has many critical tasks such as collecting signature information and publishing them. These tasks can be performed by blockchain using smart contracts such as Hyperledger Fabric.

P1. Petition Users create different signature information for different petitions. Therefore, finding correlation between Petition Users' signature information can not be possible.

P2. Petition User can prove certain personal properties requested by petition (proof request) such as age, location and also Petition User may or may not disclose additional properties.

5.2 Evaluation of Possible Security and Privacy Issues to Be Occurred in Petition Systems

Problem (Centralization): While PKI certificates are widely used in online for specific purposes such as secure connections or authenticating digital transactions and more, they have some shortcomings. From e-petition perspective, digital signatures are used for several operations such as verification or integrity e-petitions. These system depends on two required keys. The first key called the private key

or signing key is used to sign the document, and is kept by the issuer secretly. The second key called the public key or verification key is used to verify the signature. Also this key ensures that the e-petition has not been tampered with. This key is shared publicly. It means that everyone knows this key. The primary challenge in this architecture is verifying that whether you have the correct public key for the issuer or not. In PKI, several (a few hundred) certificate authorities (CAs) are to be the roots of trust.

The main problem with PKI is that the trust in this system depends on a chain of CAs' trust. Such a centralization can lead to censorship and single points of failure. CAs taking part in digital trust infrastructure can lead to vulnerabilities. If a CA makes a mistake on a digital certificate, if their service may be unavailable or may have security leakage, if they raise their prices, if their business collapse, the whole system may be fall apart.

Prevention Method: Blockchain technology that is decentralized root of trust can solve this problem. It uses decentralized network and consensus algorithm that is operating over many different nodes (machines) and replicated by many different nodes according to consensus algorithm in a decentralized network, instead of relying on CAs, organizations, or governments to be a cryptographic root of trust.

Every transaction in a blockchain requires a private key for signing. Associated public keys are stored on the blockchain. Every public key can now have its own address and Decentralized PKI (DPKI) [6] depends on this approach. This address is called DID. This identity known as self-sovereign identity, where every person, organization or thing can have its own truly independent digital identity that is not dependent to any person, company or government. In this approach, blockchain serves as registry for DIDs. For example, Sovrin Ledger acts as registry for DIDs and corresponding public keys and communications endpoints.

In our e-petition design, we have used Sovrin DIDs for signing and verification e-petitions. In our design, each DID associated with a DID document-JSON-LD file-which is used for proving ownership and control of a DID and also contains cryptographic keys and resource pointers (endpoints) to establish trusted peer to peer interactions between Sovrin entities. Operations of DDIs such as registering, resolving, updating, and revoking of DIDs on the Sovrin Network is determined by Sovrin protocol.

Because Sovrin Ledger provides cryptographic verification of the DID document which includes verification key known as public key used by the identity holder, e-petitions users can sign and verify e-petitions without relying on any centralized authority. In this approach, e-petition user completely relies on Sovrin for the trust and solves one of the biggest challenge with existing PKI (public key infrastructure) depends on centralized authorrities.

Problem (Credential Revocation): There are some credentials that lose your validity under certain circumstances in real life. In other words, the credential issuer no longer entitles for the credential and then credentials are revoked by other users. Driver licence might be an example. In some cases, user's driver licence might be revoked by driver licence department.

From this perspective, we assume that e-petition user presents some attributes for participating an e-petition. Before user participating the petition, the credential issuer may want to revoke some credentials. Issuer has ability to revoke any user attributes before e-petition validity time expired. If e-petition system are not capable of this functionality, e-petition user that does not have necessary qualification may participate an e-petition.

Prevention Method: In our design, Sovrin revocation registry feature might be used for above problem. A revocation registry is data structure which contains a single (long) number called a cryptographic accumulator and points the credential definition and written to the Sovrin ledger by the issuer like other data structure such as schema definition or credentials definitions. When needed, this value can be checked to whether data is valid or not by the verifiers via using zero-knowledge cryptography.

Only the credential holder can generate a zero knowledge proof of non-revocation by using his/her own credentials. A verifier that wants to determine whether the credential is still valid or not, verifier uses this proof which contains non-revocation with the cryptographic accumulator issued by its issuer on the Sovrin ledger. And then decides whether credentials are valid or not. When an issuer needs to revoke a credential, issuer subtracts the credential hash from the cryptographic accumulator and sends the new value to the Sovrin ledger. From that point, the identity holder can not generate a valid proof of non-revocation.

Problem (Losing or Compromising of Keys): Suppose that e-petition user's keys are compromised or stolen by an attacker. In this case, user's identity or sensitive information can be used for malicious purposes. For example, if any e-petition system uses this certificate information, user's opinions about a political or social issue may be disclosed by attacker. Therefore, users can revoke keys or certificates as soon as stolen his/her keys. In centralized authority system, this process may be costly, cumbersome and centralized.

Prevention Method: In Sovrin, key revocation is as easy as the identity owner generating a new Sovrin DID on Sovrin ledger. If any attacker stolen someone's key may attempt to revoke and replace it with a new key or key set governed by attackers.

E-petition users may have lost his/her keys when losing of a device (if the owner's Sovrin keychain only lived on a single device), or loss of an entire set of devices. In this case, key recovery steps are required for getting an key or key set. If identity holder lost the key, thereafter identity holder can not perform any interactions with Sovrin ledger.

In Sovrin, key management depends on a "web of trust" architecture. Key recovery relies on trustees specified by identity owner. Any identity owner could specify their trustees as much as his/her want. In order to start key recovery process, identity holder asks for trustees for new keys set. But identity owner waits for a while. This waiting period is called timelock. The timelock period is required for thwarting an attacker. Because an attacker may want to immediately tries to change the owner's identity records, including her designated trustees, thus preventing key recovery [18].

6. Security, Performance Analysis and Legal Issues

6.1 Security and Privacy Analysis

Strong authentication: In our design, the initial authentication of the citizen is achieved through the his/her agent. Before participate e-petition, citizens need to obtain required credentials. Citizens need to interact with credentials issuers to get required credentials for participating an e-petition. Citizens may interact with credential issuer in different ways. For example, some credential issuer may publish their DID used for accessing this credential issuer with email, another credential issuer issuer may publish their DID on their website for interacting with citizens. Some issuer may use 2 factor authentication to communicate citizens.

Authorization: Our system relies on Sovrin Ledger as trust source given that public keys, credential and schema definitions are stored. In our system, Steward is responsible for onboarding new actors and assigns role to them in the system. Credential and scheme issuers called Trust Anchor [19] are authorized by Steward. In real life, for a well established Hyperledger Indy network, Stewards (and Trustees) are responsible for maintaining the level of trust and authorization of whole network.

Any organization that wants to run a node on the Sovrin public blockchain can become a steward by following the rules defined in the trust framework. The first 24 stewards [22] span 11 countries.

Data Integrity: Data integrity is provided by digital signatures in our system. Because Sovrin is a DLT, Sovrin provides :

1. Each transaction in the blockchain is digitally signed by its generator.

2. Each transaction in block is chained to previous transaction's digital hash.

3. Validated transactions are replicated across all machines according to using a consensus algorithm.

Every transaction in Sovrin requires a private key for signing. For example, credential or schema definition are signed by public DIDs of credential issuers. Credential verifiers use digital signatures which is presented with proof by credentials holder (citizens in our case) for verification.

Confidentially: In our system, each Sovrin DID (such as citizen or credential issuer) has private agent-with its own pseudonymous network address. All the data exchanges between agents (for example citizen and credential issuer) are transferred through secure (encrypted) private peer to peer channel.

Signer anonymity: In our system, signer anonymity is achieved by the zero- knowledge properties of the anonymous credential protocols and privacy agents. Sovrin enables selective disclosure of credential attributes. Selective disclosure lets e-petition users control which user's data is shared in a specific context. For example, when you prove that you are older than 18 at an e-petition. E-petition user does not need to disclose date of birth.

Our system also provides separate agent point for each identity. If all of those Sovrin identities shared the same agent endpoint, attackers would find correlation between them.

Multiple signing prevention: In our system, every citizen connects to petition organizer via secure peer to peer fashion. Duplicate signatures are detected by petition server using DIDs of petition user without compromising petition signer anonymity.

We have used unique number credential for providing this functionality. Every petition user receives a unique number credential from Petition Registration Service. Using Sovrin predicates in proof request, petition verifier validates this number without revealing actual value of unique number credential. However, unfortunately Sovrin is not ready yet for any type of predicates at the time of writing. We could only use one type predicate for preventing double signing. But only one type predicates does not provide necessary our privacy requirements.

One time usage credential can be another solution for multiple signing problem. Even if theoretically it is possible, Sovrin has not provided yet required APIs or wrapper to developers for implementing this feature at writing this paper [12].

Public verifiability: In our e-petition system, the e-petition organizer publishes petition information of every signatures. This information can be used by a citizen to verify that her own signature information has been counted.

We have implemented this functionality by using

Sovrin self attested attributes. Petition User can create his/her own credential by using Sovrin self attested attributes. Petition User creates a value and sends to Petition Organizer when participating a petition. Petition Organizer validates proof presented by Petition User. And then Petition Organizer publishes this self attested value for verifying by Petition User. Our system relies on only petition verifier for collecting and publishing signature information. This approach could be lead to security and privacy problem. In order to improve privacy and security, the process of the collecting and publishing of signature information can be moved to blockchain supporting smart contracts such as Hyperledger Fabric [2].

6.2 Performance Analysis

Sovrin network are capable of around 100 reads 10 writes per second [21]. These values can be considered as low for a permissioned blockchain. But in Sovrin, many of network operations performed by peer DIDs. This may reduce %99 of all network reads and writes and provides better scalability with perfect horizontalness. In other words, peer DIDs don't use the ledger directly [21]. They use required bandwidth and storage from two parties that use those DIDs. Many of network operations performed by via peer message exchange. Network's load for the increased DID storage and communication burden does not change even if the number of peer DIDs increases considerably. The most important part of is that using the ledger is only required for defining credentials. The number of credentials issued or the number of users of those credentials is completely independent. For example, the millions of credentials that they issued to different organizations performed only a few of ledger writes by The Government of British Columbia.

The petition server basically has two tasks: registering user and validating user credentials(signatures). Based on the above-mentioned, there is no ledger effect for these operations. These operations are independent processes from the ledger.

As might be expected, as with any messaging application, there will be network delays due to exchanging messages on the petition server according to the increasing number of users. We have analyzed these two operations according to different number of users without considering network delays mentioned above.

Setup

The infrastructure that the our experiments are conducted on is a virtual machine with the Intel i5-2450M 2 core CPU, 8GB RAM, 50GB SSD hard drive and running Ubuntu 18.04.1.

Method

We have used Straight Line Fitting [15] method for measuring code execution time to eliminate systematic and random measurement errors [15] and obtain more reliable results. The basic idea is to first measure the time of one function call, then the time of two, then the time of three, and so on.

Then fit a straight line through these measurements. The slope a from the straight line y = a x + b gives the overall execution time of a function. For example, assume that the straight line is y = 205.91 x + 29.56; therefore, the execution time equals 205.91 millisecond

Performance Assessment of Registering Users

As we have explained in section 4, the petition server issues unique number credentials for users to register. The Petition server runs the issuerCreateCredential() function provided by indy SDK to create credentials. The execution of this function is independent from the ledger. We have computed execution times of this function according to different

number of user(N=10, 100, 1000, 10000) according to straight line method.

We have created the equation y = ax + b according to these execution times and different number of users. Then we have found that average execution time of registering users is 357,17 milliseconds.

Performance Assessment of Validating Proofs

As we have explained in section 4, the petition server receives proofs from users then validate them. The Petition server runs the verifierVerifyProof() function provided by indy SDK to validate proofs. We have computed execution times of this function according to different number of user(N=10, 100, 1000, 10000) according to straight line method.

There is a linear relationship between the execution times and the number of users. We have plotted this linear relationship with these values.

Then we have looked at the slope of this line, we have seen that the average working time of validating proofs of petition server is 500,02 milliseconds.

6.3 Legal Issues

The most important issue needs to be examined in any privacy preserving petition system like blockchain based ones is that whether e-petition system requires the processing of personal data. And also petition system has to be examined according to the legal frame-work on data protection law of the governments or institutions that organizes e-petitions for a political or social issue.

The data protection legislation only applies when the usage of personal data takes place according to data protection laws. If the data are anonymous and can not be related to a real person, their usage does not to be included data protection legal frameworks. Privacy risks in the blockchain based petition implementation are related to the possibility of reidentifying anonymous citizens by attacking system or analyzing the system. Thus, any blockchain based e-petition system does not allow any kind of reidentifying and does not provide any mechanisms for deanonymization like our petition system does.

Abusing of personal data obtained reidentifying individuals to categorize, profile the people based on their ideas, individuals are being targeted and harassed, or being subject to social pressures by their family, friends, neighbors, or work environment.

7. Implementation

We have considered a scenario where only disabled people with a driver's license can participate and implemented this scenario using Hyperleder Indy.

In this case, our Indy network are consist of four indy nodes. These nodes hold ledger and execute consensus algorithm and communicate each other for performing ledger operations. Also we have used five Indy agents represents each actor. In our demo, these are Petition User, Petition Registration Service, Driver Licence Department (Credential Issuer for driver licence), Health Department (Credential Issuer for health status), and Petition Organizer(for Petition Server). In this demo, we have roughly divided them into various stages.

Preparation

All actors have a public Decentralized ID (DID). This public DID (or endpoint DID) represents the players in the ledger. Also each actor has Citizen-Id as credential for knowing each other.

In this step, Credential Issuers create schema and credential definitions and submit Sovrin Ledger. Our issuers are Petition Registration Service, Driver Licence Department and Health Department. Petition Registration Service creates the Unique Number Schema and Petition Organizer Schema definition. Driver Licence Department creates Driver Licence

Schema definition and lastly Health Department creates Disabled Status Schema definition.

Driver Licence Department creates the Driver Licence Credential Definition derived from the Driver Licence Schema. Similarly, Health Status Department creates the Health Status Credential Definition derived from the Health Status Schema and Petition Registration Service creates the Unique Number Credential Definition derived from the Unique Number Schema and Petition Organizer Credential Definition derived from the Petition Organizer Schema.

Registering and Getting Credentials From Issuers

1 Peer to peer secure communication is established between Petition User and Petition Registration Service, Driver Licence Department or Health Status Department. This can be requested by either side and accepted by the other.
2 Petition Registration Service, Driver Licence Department or Health Department sends Credential Offer to Petition User
3 Petition User accepts offer and requests credentials.
4 Petition Registration Service, Driver Licence Department, Health Status Department sends credentials to Petition User. Petition User stores these credentials in his/her wallet.

Participating and Publishing a Petition

1 Peer to peer secure communication is established between Petition User and Petition Organizer. This can be requested by either side and accepted by the other.
2 Petition Organizer sends proof request to Petition User.
3 Petition User responds with proof based on the credential she/he has.
4 Petition Organizer validates the proof and accepts the proof.
5 Petition Organizer publishes user's sign values.

All the components mentioned above have implemented as Docker [14] containers running in localhost. We have used two docker images in our demo. These are indy-node ve indy-agentjs from bcgovimages/von-image [16].

indy-node: This Docker image is the software that is used for initiating ledger nodes, generates transaction and other ledger functions. Four containers are instantiated from indy-node. Also the ledger explorer (called webserver) is also using this image. This docker file for this image is pool.dockerfile.

indy-agentjs: This Docker image is the agent software. Agent specific information is given through environment variables. Our agents' app is web application and uses Node.js JavaScript runtime for server side operations and REST APIs for requesting and receiving responses via HTTP protocol such as GET and POST. We have also used indy-sdk Nodejs API wrappers for implementing various ledger operations such as creating schema and credential definition, creating a wallet, connection a pool or creating a DID etc.

In this demo, five agents have been instantiated. We have used several ports for demonstration and observation.

Port 9000: Ledger explorer

Port 3002: Driver Licence Department

Port 3003: Health Department

Port 3004: Petition User

Port 3005: Petition Organizer

Port 3006: Petition Registration Service

8. Conclusion and Future Work

Citizens can express their support or dissatisfaction with government initiatives, and provide feedback to government institutions through electronic petitions. Mostly petitions deal with social and political issues such as abortion, gay marriage, immigration law, issues related to religious freedom. In other words, they include sensitive information about users such as individual's political ideology, religious beliefs, and sexual preferences. Abusing of this information may influence users negatively in their private life. For protecting these issues, electronic petitions system provides required functionality.

In this paper, we have expressed our privacy preserving electronic petition design which depends on Sovrin self-sovereign identity framework. We have introduced Sovrin Privacy by Design architecture including pairwise pseudonymous identifiers, peer-to-peer private agents, and selective disclosure of personal data using zero-knowledge proof cryptography. Citizens anonymously participate a petition and express their support or dissatisfaction with any social and political issue freely with our system using Sovrin Trust Framework.

One of the future work is that moving some tasks of petition organizer to the system implemented on top of a blockchain platform in order to benefit from its security, immutability, and transparency. Because, we assume that all actors are trusted such as petition organizer. But this assumption could lead to security or privacy problems. Petition organizer has many critical tasks such as collecting signature information and publishing them. These tasks can be performed by blockchain using smart contracts such as Hyperledger Fabric. Some of petition server tasks could be distributed to Hyperledger Fabric by developing a custom Membership Service Provider in Fabric for managing digital identities using Sovrin decentralized identity management system. Our petition system's reliability can be increased by incorporating two blockchain platforms.

During the implementation, we have realized that one time usage Sovrin credential could be needed for some cases. However, unfortunately Sovrin is not ready yet for use at the time of writing [12]. We plan to use this feature on our petition system when Sovrin(INDY-SDK) supporting required APIs or wrappers.

Acknowledgment

We would like thank for the insightful comments of the anonymous reviewers.

References

[1] J. Alexander, S. Landers, and B. Howerton. Netvote: A Decentralized Voting Network, 2018.

[2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the 13th EuroSys conference*, pages 1–15, 2018.

[3] 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. IEEE, 2013.

[4] D. J. Bernstein. Curve25519: new Diffie-Hellman speed records. In *International Workshop on Public Key Cryptography*, pages 207–228. Springer, 2006.

[5] C. Diaz, E. Kosta, H. Dekeyser, M. Kohlweiss, and G. Nigusse. Privacy preserving electronic petitions. *Identity in the Information Society*, 1(1):203–219, 2008.

[6] C. Fromknecht, D. Velicanu, and S. Yakoubov. A Decentralized Public Key Infrastructure with Identity Retention, 2014. IACR Cryptology ePrint Archive.

[7] P. Hintjens. *ZeroMQ: messaging for many applications*. O'Reilly Media, Inc., 2013.

[8] F. Þ. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa, and G. Hjálmtỳsson. Blockchain-based e-voting system. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 983–986. IEEE, 2018.

[9] S. Inc. The Reliable Asynchronous Transfer Protocol. (accessed 28.11.2019).

[10] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.

[11] N. Kshetri and J. Voas. Blockchain-enabled e-voting. *IEEE Software*, 35(4):95–99, 2018.

[12] J. Law and D. Hardman. Anonymous Credentials in Sovrin. (accessed 14.12.2019).

[13] P. McCorry, S. F. Shahandashti, and F. Hao. A smart contract for boardroom voting with maximum voter privacy. In *International Conference on Financial Cryptography and Data Security*, pages 357–375. Springer, 2017.

[14] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), Mar. 2014.

[15] C. Moreno and S. Fischmeister. Accurate measurement of small execution times-getting around measurement errors. *IEEE Embedded Systems Letters*, 9(1):17–20, 2017.

[16] B. G. V. O. Network. BCGov VON Docker Images. (accessed 04.12.2019).

[17] H. M. Patel, M. M. Patel, and T. Bhatt. Election Voting Using Block Chain Technology. *International Journal of Scientific Research and Review*, 7(05):1–4, 2019.

[18] D. Reed, J. Law, and D. Hardman. The Technical Foundations of Sovrin. Technical report, Technical report, Sovrin, 2016. Retrieved from: https://www. evernym. com/wp, 2016.

[19] Sovrin. Sovrin Board of Trustees. (accessed 29.11.2019).

[20] Sovrin. Sovrin Glossary V2. (accessed 14.11.2019).

[21] Sovrin. Sovrin Network Performance. (accessed 22.02.2020).

[22] Sovrin. Sovrin Network Stewards. (accessed 24.11.2019).

[23] K. Verslype, J. Lapon, P. Verhaeghe, V. Naessens, and B. De Decker. Petanon: A privacy-preserving e-petition system based on idemix, 2008.

[24] J. Wahab. Coconut e-petition implementation, 2018. arXiv preprint arXiv:1809.10956.