A Blockchain-based E-petition system

Xiangman Li

August 2021

1 Introduction

Petition systems have always been a means of conveying public opinion to those in power. In the 21th century, petitions are one of the most common forms of political activity [?]. With the development of the Internet, e-petitions are becoming more and more popular. The most popular e-petition system was the UK parliament's upgrade e-petitions systems in 2015. 14 million people signing at least one petition, and over 30,000 e-petitions are received [?], becoming the highest usage parliamentary e-petitions system worldwide. At the same time, there are many petition sites to help people start and run non-political petitions. The e-petition system is expected to solve some of problems with the paper-based petition system. First, the paper petition ballots are counted by the Counting Agent, which has possible to change the number of votes accidentally or intentionally. Second, since the counting process and results are managed and stored centrally, it the public is hard to verify the correctness of the paper petition result. At the same time, the result would be incorrect due to the paper petitions could get lost in the mail by accident.

However, the petition system still has plenty of critics because the public's expectations are high, and little has been achieved. The public still cannot check the result on time and verify if the result is fair. In addition, personal information is not sufficiently protected, and petitioners are concerned that their information will be leaked on the Internet and cause unnecessary trouble. In addition, the paper-based petition would have a hard time dividing and statistic the petitioners, and it is challenging to arrange different groups casting different numbers of voting. Hence, using Blockchain and Attribute-Base Signature (ABS) to build a petition system is considered.Blockchain is decentralized through the participation of members.No single point of failure. No single user and group can have control, and the data entered on Decentralized Blockchains are irreversible, which means anyone can participate and validate the data. These features of Blockchain promise that the process and result of the petition are fair and transparent. In an attribute-based signature (ABS), users sign messages with using their attributes from an attribute authority, which means a signature is a claim regarding the attributes that signer owns; it does not show the signer's identity.[?] The primary ABS considers the single attribute authority, but El Kaafarani et al.[?] and Okamoto et al.[?] provide a scheme that can avoid central authority and involve multiple attribute authorities, which means different signers would satisfy the different policies and sign different messages. This solution avoids having a designated tracing authority but also adds accountability to attribute-based signatures. In addition, it allows signers to link some of their signatures with the same verifier directly and keep anonymity.[?]

Contribution. Consequently, in this report, our target is improving a blockchain-based and attribute base signature e-petition systems. The contributions are listed as follows.

1) We formalize the system model of e-petition systems based on blockchain and attribute base signature.

2)A construction of e-petition system based on blockchain and attribute base signature is described. It proves that it satisfies fairness, multiple attributes, keeps identity privacy,and unforgeability. In addition, we modify the blockchain-based voting system scheme in [?] and the ABS in [?] to establish the basic e-petition system and add statistical steps to complete the self-tallying part in the e-petition system.

2 Related Work

3 Syntax of e-Petition system

3.1 System model

As mentioned before, users who have different attributes will sign the different messages, representing different voting numbers. In addition, in Attribute-based Signature, a set $\mathbb{AA} = \{AA_i\}_{i=1}^{n}$ of attribute authorities are established, and $\mathbb{AA}_i$ is the space where attribute authority AA_i manages attributes. Suppose that $\alpha \subset \mathbb{AA}$ is a set of attributes that satisfy a specific predicate, and $a \in \alpha \implies \exists \mathrm{A}_i$, so attribute a belongs to AA_i.

• **Setup** (1^λ): This algorithm is executed to set up the whole system. It returns a public parameters pp when input a unary string, which λ is the security parameter.

· Account Establishment: This protocol is executed to establish the user's account. It has three algorism AASetup, UkeyGen, AttKeyGen.

· AASetup(aid,**pp**): Generating public/private key pair(pk$_{AA}, sk_{AA}$) by attribute authority AA$_{aid}$. The authority keep sk$_{AA}$ privity and release pk$_{AA}$ to the public.

· UkeyGen (id,pp): Generating user's private key sk_{id} by user id.

· AttKeyGen ($f(sk_{id})$,id,a,sk_{AA}): Is run by AA that manages attribute a. AttKeyGen generate the private key skid of user id, and bound id and injective one-way function $f(sk_{id})$.

After establishing the account, users will get a privacy account and are eligible for signing in the next step.

• **Sign** (p, Ω, $sk_{id,\alpha}$,sk_{id}): This algorithm is executed to produce a signature if the user's attributes satisfy the predicate Ω. Users can use corresponding secret keys $sk_{id,\alpha}$ =$\{sk_{id,\alpha_i}\}_{\alpha_i \in \alpha}$ to generate a signature $\sigma = \sigma_{ABS}$ on the petition p.

• **Verify** (σ, $\{$pk$_{AA}\}_i$, Ω, p): Take the signature σ on the petition p and verify the signature. The attribute authorities' verification keys $\{pk_{AA_i}\}_i$ manage attributes that are involved in Ω. It will return 1 if the signature is valid.

• **Tally** (verify(), c$_{AA_i}$,(t$_i$)$_{(i \in n)}$,MP): This algorithm statistic valid signature, and different attribute authority will has different count number. After the petition number is equal to the required number, it will show the percentage of each petition received and output the petition Result.

3.2 Security Model.

The security requirement of an e-petition system is shown as follows.

Anonymity. This require that the signature secret the user's identity and the attributes. We consider the adversary as follows,

·Adversary's Power: The adversary control all attribute. It can take the signer's private key who is chosen. At the same time, it can get any attributes' secret key and it owns a signing oracle which can question messages on behalf of honest users.

Unforgeability. Users cannot refer to signature policies that their attribute set is not satisfied to output signatures on messages, even if they collect attributes together. This policy guaranteed clustering resistance.

·Adversary Power: It owns a signing oracle, and it can ask for user's personal secret key. At the same time, it can corrupt attribute authority and ask for attribute's secret key.

Correctness. Signatures generated by honest users must be properly verified.

3.3 Block model.

Block model. Bilinear Group. A bilinear group is a tuple P =($\mathbb{G}_1$,$\mathbb{G}_2$,$\mathbb{G}_T$,p,g_1,g_2,e), where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of a prime order p, g_1 and g_2 generate $\mathbb{G}_1$ and $\mathbb{G}_2$. [?] We would like use Type-3 in [?] to build our bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, which $\mathbb{G}_1 \neq \mathbb{G}_2$, $\mathbb{G}_1$ and $\mathbb{G}_2$ do not have computable isomorphisms.

Digital Signature. We would like to use Digital Signature (DS) scheme which called one-time signature. It is correct and unforgeability, which means the adversary can just sign one-times only if it generates a new public/private key pair. We will use a different type of full Boneh-Boyen signature scheme according to [?], namely, BB † scheme. The schemes are described as follows,

· KeyGen(P): Let $x, y \leftarrow \mathbb{Z}_p$, set $(X, Y) = (g_2^x, g_2^y)$, which (X, Y) is the public key and (x, y) is the private key.

· BB.Sign(sk,m): Take the signature $\sigma = g^{1/(x+ry+m)}$ on massage $m \in \mathbb{Z}_p$. In σ, $r \leftarrow \mathbb{Z}_p$ and $x+ry+m \neq 0$. In addition, the signature $\sigma = (g_1 \cdot h_1^z)^{(1/x+ry+m)}$ in BB†.

· Verify: In BB† scheme, if $e(\sigma, X \cdot Y^r \cdot g_2^m) = e(g_1 \cdot h_1^z, g_2)$ output 1, otherwise 0.

Non-interactive Zero Knowledge Proofs. In this system, each proof needs its own new public reference string. Common reference strings are usually not random strings. According to [?], a Non- interactive Zero-Knowledge Proof system is a tuple of algorithms for relation R. R is an NP relation on pairs (x, y) with a language C'= $\{y| \ \exists x, (x, y) \in R\}$. The NIZK system is composed by (Setup, Prove, Verify, Extract, SimSetup, SimProve). NIZK is required Completeness, soundness and zero-knowledge. Completeness required that the honest verifier will be trust this statement if the statement is true. Soundness required that only those who can prove that they know the intended secret will get the information. Zero-knowledge required that no witness information is shown.

4 Construction of e-petition system

4.1 Overview of construction

In this section, the construction of e-petition system is described. In Setup algorithm, the common reference crs is generated for NIZK system. Then a pseudo-key pair $(\mathbf{pk_{psdo}}, \mathbf{sk_{prdo}})$ is generated for Digital Signature scheme $\mathbf{D_2}$. At the same time, a public parameter $\mathbf{pp} = (\mathbf{pk_{psdo}}, \mathbf{crs}, \mathbb{A}, \mathcal{H})$, where $\mathcal{H}$ is the collision-resistant hash function.

In account establish algorithm, a public/private key pair$(\mathbf{pk_{aid}}, \mathbf{sk_{aid}})$ is generated for Digital Signature Scheme $\mathbf{D_1}$ after a new attribute authority is created. If the user **id** needs to generate a signing key that the attribute is $a \in \mathbb{A}$,the managing attribute authority the managing attribute authority need to sign **id**, a and one-way function on user's private key $\mathbf{sk_{id}}$ together. This signature is the private key for user's attribute.

Then the user will use this signature to sign the messages that the attributes satisfied. The signature will generate a NIZK proof that the modified predicate Ω' will still has proper user's attribute be satisfied.

According to the [?], the languages for the NIZK proofs are defined as follows. We define a monotone Boolean function $\Phi : \{0, 1\}^n \rightarrow 0, 1$, and a matrix in a field F has the size $l \times w$. Then a labelling function make a transition from l to n. If every $(x_1,\ldots,x_n)\in\{0,1\}^n$, M is a monotone span program for Φ *over* $\mathbb{F}$, *and* v is the secret vector in it, which means,

$[\Phi(x_1,...x_n)=1]\Leftrightarrow[\exists v\in\mathbb{F}^{1\times t}: v\cdot M=[1,0,...,0]\bigwedge(\forall i: x_{a(i)}=0\Rightarrow v_i=0)]$

Base on the definition of the span program, the satisfiability of Ω' can be proved. User need to prove the property of the $v\in\mathbb{Z}_p^{|\Omega'|}$ with $v\cdot M=[1,0,...,0]$. The zero element in $v\cdot M$ is the attribute that user does not need, and user need to prove the property of the attribute and pseudo-attribute if the element is non-zero.

In Tally algorithm, c_{AA_i} need to count different times base on the petition rule. If Verify $(\sigma,\{pk_{AA}\}_i,\Omega,p)=1$, then c_{AA_i} will be counted until c_{AA_i} get the maximum number of petitions MP. Then, the algorithm will calculate the ratio of c_{AA_i} to the MP, and output the result.

○ **Setup**(1^λ):

· $(crs,xk)\leftarrow NIZK.Setup(1^\lambda)$

· $(\mathbf{pk_{psdo}},\mathbf{sk_{prdo}})\leftarrow DS.KeyGen(1^\lambda)$

· Compute a Collision-resistant hash function $\mathcal{H}:\{0,1\}^*\rightarrow\mathcal{M}_{DS}$

· $\mathbf{pp}=(\mathbf{pk_{psdo}},\mathbf{crs},\mathbb{A},\mathcal{H})$

· Return **pp**

○ **Account Establish**:

· $(\mathbf{pk_{aid}},\mathbf{sk_{aid}})\leftarrow DS.KeyGen(1^\lambda)$

· Return $(\mathbf{pk_{aid}},\mathbf{sk_{aid}})$

· $sk_{id}\leftarrow$ UKeyGen(id,pp)

○ **Sign** $(p,\Omega,sk_{id,\alpha},sk_{id})$:

· Return $\perp$ if $\Omega(A)=0$

· $\Omega':=\Omega\bigvee a_{psdo},\mathbf{M}\in\mathbb{Z}^{\alpha\times\theta}$ is the Ω' span program.

· Compute $\alpha:=\{a_i\}_{i=1}^{|\Omega'|}$ to be the attribute in Ω'

· $\pi\leftarrow NIZK.Prove(crs,x,y)$ if R$(x,y)=1$.

· $((\mathbf{pk}=\{pk_i\}_{i=1}^{|\Omega'|},\mathbf{a}=\{a_i\}_{i=1}^{|\Omega'|})$, (sk$_{id}$, id,v,$\boldsymbol{\sigma}=\{\boldsymbol{\sigma}_{\boldsymbol{a_i}}\}_{\boldsymbol{i=1}}^{\boldsymbol{|\Omega'|}}$)) :

($\boldsymbol{v}$M=[1,0,...,0]) $\bigwedge\limits_{i=1}^{|\Omega'|-1}$

$(v_i=0\bigvee$ D$_1$.Verify(pk$_i$,id,sk$_{id}$,a$_i$, σ_{a_i})=1)

$\bigwedge$(v$_{|\Omega'|=0}\bigvee$ D$_2$.Verify(pk_{psdo}, a_{psdo},$\sigma_{a_{psdo}}$)=1)

· Return σ:=$\{\sigma_{a_i}\}_{i=1}^{n}$

○ **Verify** $(\sigma$, $\{$pk$_{AA}\}_i$, Ω, p)

· Return NIZK.Verify(crs,π)

○ **Tally**(verify(), c$_{AA_i}$,(t$_{AA_i}$)$_{(i\in n)}$, MP)

· $c_{AA_i}=c_{AA_i}+(t_{AA_i})_{(i\in n)}$ if Verify(σ, $\{$pk$_{AA}\}_i$, Ω, p) $=1$

· Return $\sum c_{AA_i}$ if $\sum c_{AA_i}$= MP

· Return ($\frac{c_{AA_i}}{MP})_{i=1}^{n}$

4.2 Detail of construction

User's task in e-petition system is make a zero-knowledge proof of π.

The first step is user need to prove a secret vector v is spanned the public matrix **M** which we talked about in span program earlier. The purpose of this step is shown how to hide which subset of user's attributes has used to satisfy the modified predicate Ω' [**?**]. The Proof process is shown as follows,

$\boldsymbol{v}$**M**=[1,0,...,0],which also can be written the following form,

$$\sum_{i=1}^{|\Omega'|} v_i \mathbf{M}_{ij} = \begin{cases} 1 & j=1 \\ 0 & 2 \leq j \leq \theta \end{cases}$$

· Commitment of v

$$\beta_{v_i}, \beta_{w_i}, w_i \leftarrow \mathbb{Z}_p, i = 1...\alpha$$
$$\overset{\wedge}{v_i} = g_1^{v_i} \cdot k_3^{\beta_{w_i}}$$
$$\mathcal{V}_i = g_1^{\beta_{v_i}} \cdot k_3^{\beta_{w_i}}$$

· Proof of statement

$$\forall j \in [1,\theta]:\ \Lambda_j = \prod_{i=1}^{\alpha} k_3^{w_i \cdot M_{ij}},\ \lambda_j = \prod_{i=1}^{\alpha} (k_3^{w_i \cdot M_{ij}})^{\beta_{t_i}}$$

The second step is proof whether the signature can be verified base on the verification key.

In Digital Signature, a verification function can be used.

$$e(\sigma_{a_i}^{v_i}, X, Y^r \cdot g_2^{a_i||id}) = e(g_1, g_2) \cdot e(h_1^{sk}, g_2)$$

According to the BB and BB† scheme, the signature can be described as follows,

$$\sigma_{a_i} \begin{cases} (g_1 \cdot h_1^{sk})^{1/x_i + y_i r_i + a_i||id} & attribute \\ (g_1^{1/x_i + y_i r_i + a_{psdo}} & pseudo - attribute \end{cases}$$

The public key of a_i is $X_i = g_2^{x_i}$ and $Y_i = g_2^{y_i}$. We need to replace the zero-element v_i in secret vector v to the random number. The purpose is hide subset of the attributes which satisfied the predicate Ω.