

オンライン投票システムの投票者インタフェースのための Web API

東 知哉¹ 白石 善明¹ 掛井 将平² 毛利 公美³ 森井 昌克¹

概要: 紙投票に比べて参加がしやすく集計が容易なオンライン投票は、コロナ禍のような人との接触機会を減らさなければならない不測の事態にも対応できるという点で今後より注目されると考えられる。これまでのオンライン投票システムの提案においては、オンライン投票に求められる性質を満たすための暗号技術やブロックチェーンなどを用いた理論的あるいはシステムの貢献が主題であった。必然的にそれらは独立した実装となり、投票者の操作は統一感を持たない。投票者の操作がシステムごとに大きく変わらなければ投票率の向上が期待できる。本論文では、ブロックチェーンを用いた投票システムについて調査し、それらのどの投票システムでも適用可能な Web API フレームワークを提案している。9つのオンライン投票システムをバックエンドシステムとして本フレームワークを用いてフロントエンドを実装し、バックエンドの違いが UI に影響しないことを確認している。フロントエンド開発者のコード記述量は最大 47 行の変更でバックエンドシステムを切り替えることができています。

Designing Web API for Voter's Interface of Online Voting Systems

TOMOYA AZUMA¹ YOSHIAKI SHIRAISHI¹ SHOHEI KAKEI²
MASAMI MOHRI³ MASAKATU MORII¹

1. はじめに

紙投票に比べて参加がしやすく集計が容易なオンライン投票は人同士の接触を避けながら一連の投票を行えることができるという点で今後の活用が期待される。投票用紙を投票箱に入れて集計する従来の紙投票に代わり、オンライン投票では公開掲示板を用いて投票の暗号化や電子署名による証明等の暗号技術を用いたやり取りを行うことで投票者のプライバシーや検証可能性を実現している。最近では公開掲示板にブロックチェーンを用いたオンライン投票システムを設計している研究[1]-[9]がある。ブロックチェーンとして形成された分散台帳はネットワーク参加者によるコンセンサスアルゴリズムによって改ざんが不可能である。また、スマートコントラクトと呼ばれる参加者のノード内に存在するプログラムコードは、参加者間のやり取りを自動化することができる。オンライン投票システムはブロックチェーンの分散台帳とスマートコントラクトによって投票の公平性、透明性を更に高く保証できるとされている。

投票システムの投票者から見えないバックエンドシステムは用いられる暗号技術やブロックチェーンによって異なる様相を呈することになり、それらは必然的に独立した実装となる。このとき投票者のユーザインタフェース (UI) は当然異なることとなり、投票者であるユーザは参加するシステムごとにその操作方法を理解し、操作しなければならない。また、開発者はバックエンドにオンライン投票シ

ステムを導入する際に、用いる投票システムによってその開発の手順が大きく異なれば、システム変更に多くの労力を要することになる。投票システムの機能的な違いによる UI の違いはあるとしても、オンライン投票の基本的な機能に対して UI に統一感を持たせることは投票率の向上に向けての課題であると考えられる。

本論文では、バックエンドの投票システムが異なってもユーザが統一した操作感で投票可能とするフロントエンドの Web アプリケーションの開発を企図して、まず、既存のブロックチェーンを用いる投票システムを調査し、そしてそれらの投票システムに適用可能な Web API フレームワークを提案する。

2. 既存システムのインタフェースの調査

Web API の設計にあたり、まず、ブロックチェーンを用いたオンライン投票システム[1]-[9]のインタフェースを調査した。システムごとに異なる投票者の処理は以下のようなものからなる。

- 投票者用パラメータ (暗号鍵、署名鍵) の生成
- 投票者権限の取得 (ユーザ認証)
- 投票用紙作成用パラメータを取得
- 投票内容を選択
- 秘匿のための暗号化
- 証明データ (署名) の生成
- 第三者による証明データ生成のリクエスト送信
- 投票用紙の登録

1 神戸大学
Kobe University, Hyogo, 657-8501 Japan

2 名古屋工業大学
Nagoya Institute of Technology, Aichi, 466-8555 Japan

3 岐阜大学
Gifu University, Gifu, 501-1193, Japan

さらに、それらの処理の内容はシステムによって異なる。例えば、①投票に必要な証明データとなる署名を自身で生成するもの[1]がある一方、第三者によるブラインド署名をリクエストすることで署名を得るもの[2][3]も存在する。また、②暗号の演算方法の違いにより、投票用紙生成に伴う投票内容の暗号化の際に自身の持つ個別の暗号パラメータを入力して用いるもの[4]や、公開掲示板情報を利用してシステム内で自動的に演算されるもの[1][2][5][6][7][8][9]があるなど、同様の処理でも指定するパラメータに違いがある。

以上のようなシステムにより処理の内容やユーザの操作数が異なっていることがユーザの操作感の違いを生むことにつながる。そこで、システム内で行われている処理について共通しているものを集約することで、図1のように投票者の操作を次のようにまとめることができる。

- 準備フェーズ Pre-SignUp, SignUp
- 投票フェーズ GetParam, RequestSign, Sign, PrepareBallot, CastBallot, CheckBallot
- 集計フェーズ CheckResult

上で述べた①は、RequestSign, Sign と操作を分けている。②は集約した各処理に必要なパラメータをそれぞれ不足なく付与できるようにする。このように、各フェーズで行われる処理とそのパラメータをまとめる形で一般化することでこれら操作を Web API として利用できるようになる。

図1の投票フェーズにある CastBallot を例として一般化の方法を述べる。なお、CastBallot はすべてのシステムに共通して存在する、投票を実行する処理である。

表1は調査した9つの文献について、CastBallot に該当する処理内容と入力するパラメータの対応を表している。先に述べたように、システムに応じて事前に投票操作の有無やその内容に違いがあるため、完成された投票用紙を公開掲示板に送信するだけのものや、署名や暗号化用のパラメータを用いて投票を行うものなどがある。また、公開掲示板として用いるブロックチェーンに違いがあることから匿名 ID の有無の違いもある。このようにシステムによって処理内容が異なる CastBallot や他の処理において、用いられるパラメータを全て適切に付与できる Web API を与えることができれば、ユーザが統一した操作感で投票可能と

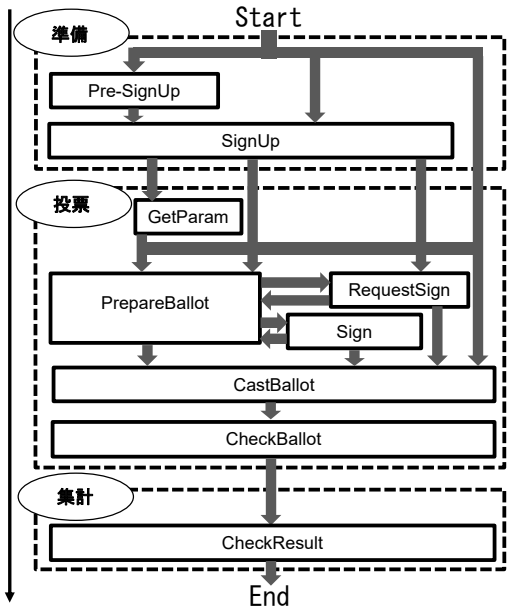


図1 システムごとに異なる処理を集約

なるフロントエンドの Web アプリケーションを開発できるようになる。

3. Web API とバックエンドのシステム変更

ユーザの UI であるフロントエンドの Web アプリケーションが Web API の機能を持つシステムに対して HTTP リクエストを送信することで、リクエスト内容や付与したパラメータに応じた結果が送られてくる。バックエンドのオンライン投票システムが共通の Web API を備えることで、バックエンドの投票システムが変わったとしても、フロントエンドは用いる Web API とパラメータを変更するのみで異なる投票システムに対して統一した操作感でユーザに投票機能を提供可能となり、UI への影響を限定的にすることができる。加えてアプリケーション開発者は少ないコードの書き換えでバックエンドシステムを切り替えることができる。

表1 調査対象論文の投票（CastBallot）に関わる処理内容と入力値

	処理内容	匿名 ID	投票用紙	投票内容	認証用鍵	暗号化用公開鍵	署名鍵	署名
[1]	投票用紙に署名を付けて投票		✓					✓
[2]	投票用紙に署名を付けて投票		✓					✓
[3]	投票内容に署名を付けて匿名 ID から投票	✓		✓				✓
[4]	投票内容を暗号化し投票			✓		✓		
[5]	認証鍵により権限取得、公開パラメータから投票用紙を生成し投票				✓	✓	✓	
[6]	投票用紙番号と投票内容を投票		✓	✓				
[7]	投票用紙に ID を付与して投票	✓	✓					
[8]	投票用紙を投票		✓					
[9]	投票用紙を投票		✓					

4. 投票に関わるインタフェースの Web API 化

4.1 インタフェースの一般化

CastBallot を例にして、任意のシステムで適用可能にするためのインタフェースの一般化について述べる。ユーザが行う投票を任意のシステムで行える CastBallot の引数として以下のパラメータを用意する。

- ID : 投票用紙の識別子として用いられる (匿名 ID, 個別に所持する公開鍵)
- ballot : 公開掲示板に登録する投票用紙
- option : 自身が選んだ投票内容
- auth_key : 投票に用いる個別の情報 (第三者によって付与されたブラインド署名, 事前準備されている投票用紙) を得るための認証用鍵
- enc_param : 投票用紙生成のために投票内容を暗号化する公開鍵
- sign_param : 署名生成用秘密鍵
- signature : 生成された署名

表 2 各 Web API でのパラメータの役割

パラメータ	使用する API	内容
ID	PreSignUp	各投票者の ID
	SignUp	
	CastBallot	投票者秘匿用 ID
password	PreSignUp	投票者のパスワード
	SignUp	
name	SignUp	ユーザネーム
voter_list	SignUp	投票者リスト
option_list	SignUp	候補者リスト
vote_policy	SignUp	分散化された投票用紙開票のポリシー
secret_key	SignUp	投票者用秘密鍵
public_key	SignUp	投票者用公開鍵
auth_key	GetParam	公開掲示板からパラメータを取得する際の認証用鍵
	CastBallot	
message	RequestSign	署名を付ける対象の文
	Sign	
voter_id	RequestSign	リクエスト送信者識別用公開鍵, ID
verify_key	Sign	検証用パラメータ
sign_param	Sign	署名鍵
	PrepareBallot	
	CastBallot	
signature	CastBallot	生成された署名
enc_param	PrepareBallot	暗号化用公開鍵
	CastBallot	
ballot	CastBallot	投票用紙
option	PrepareBallot	投票内容
	CastBallot	

これらのパラメータは CastBallot において以下のように機能と関係する。

- 投票用紙の生成(option)
 - 投票内容を暗号(enc_param)
 - 暗号文に対しての署名を生成(sign_param)
- パラメータ, 署名の取得(auth_key)
- 投票用紙の提出(ID,ballot, signature)

その他 8 つのインタフェースも同様に一般化を行う。表 2 は各 Web API に用いられるパラメータの役割を示す。これらパラメータを用いた Web API を以下のように設計する。

投票者情報登録 :

Pre-SignUp (ID, password)

投票権認証 :

SignUp (ID, password, name, voter_list, option_list, vote_policy, secret_key, public_key)

投票情報の取得 :

GetParam(auth_key)

署名生成のリクエスト :

RequestSign (message, voter_id)

署名の実行 :

Sign (verify_param, sign_param, message)

投票用紙の作成 :

PrepareBallot (option, sign_param, enc_param)

投票用紙の送信 :

CastBallot (ID, ballot, option, auth_key, enc_param, sign_param, signature)

投票結果の確認 :

CheckBallot ()

集計結果の確認 :

CheckResult ()

4.2 Web API の実装例

4.1 節で提案した Web API フレームワークの実装例について述べる。フロントエンド上での本 Web API を操作は、バックエンドシステムの URL とパラメータで行われる。AP サーバ, Web サーバ, 投票システムを実装したバックエンドサーバをそれぞれコンテナとして実装し, AP サーバと投票システムの間を本フレームワークを用いて通信する (図 2)。

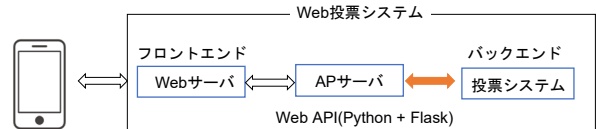


図 2 投票システム実装例

AP サーバは Python + Flask で実装する。投票システムは入出力部分を実装し、提案した Web API を用いて通信することで、レスポンスを得る様子を確認することができる。以下に、文献[2]のバックエンドシステムに対して実行する PrepareBallot の様子を示す。

文献[2]のシステムの PrepareBallot の実行コード

```

@app.route('/PrepareBallot/2', methods=("GET", "POST"))
def PrepareBallot():
    option = request.form["option"]
  
```

```

sign_param =request.form["sign_param"]
enc_param =request.form["enc_param"]
payload = {
    "option":option,
    "sign_param":sign_param,
    "enc_param":enc_param
}
json_data = json.dumps(payload)
r = requests.post(
    'http://system2:5001/api/PrepareBallot',json=json_data
)
response = json.loads(r.text)
return render_template("Output.html",result=response)

```

文献[2]のシステムからのレスポンス

```

{
  "Encrypted_ballot":
  "5591752457686226735598262367839506510645995588905
  4350610287208764890590394130902591795377592421092
  319172646059324942411701252639144924817004666348"
}

```

5. 評価

5.1 評価対象

関連研究で調査した 9 つの投票システムをそれぞれコンテナとして入出力部分を実装し、アプリケーションサーバとなるコンテナから提案した API を用いて HTTP リクエストで操作する。フロントエンド側では各投票システムに対して同じ操作画面を用意し、対応する値を入力して実行することで入力値をパラメータとして付与させた HTTP リクエストがバックエンドシステムに送られ、受け取ったレスポンスをそのまま画面に表示する。一つの API に対しての実行コードは 4.2 節で示したように、URL に対応する関数の定義、入力パラメータの用意、API を実行しレスポンスを return する構造を持つ。これによりバックエンドシステムに依存せずユーザ視点で統一された手法で投票システムの操作が可能であるかを確認するとともに、フロントエンド開発者がバックエンドシステムの変更に対して記述量の大きな差がなく実装できるか評価する。

5.2 結果

コンテナを起動させ、ブラウザから投票システムを操作する。図 3、図 4、図 5、図 6 は投票実行用 API “CastBallot” を用いて同じフロントエンドから文献[3]、文献[5]、文献[6]、文献[7]のシステムを実行する様子を示している。ユーザは投票実行画面で各システムに応じた値を入力して実行する

CastBallot - System3

Menu
995f16ac664dc9a646ac2268e06cac022882d8ceeb6d415ea4a8281786045114f183af3f55899f08c225a66ac61354f97925434dbb9c4b2415c23
A
auth_key
enc_param
sign_param
["3271331030a5ac3942ec75a710b3bd66fc6290754832a8102cb5380e1837a390bd729add7d2598519c8947b9f4fd0d296382462f4616b292dcd10", "fa150511ef9e21ee5c853bf3157ffd90d143ee929fe854865f7ead36a6238014280742364228349303809546343146272722644188"]
実行

図 3 文献[3]の CastBallot 入力画面

CastBallot - System5

Menu
ID
ballot
option
c085f77faa86ed00aefe62b128cd9f45d1d9c4ba1da4eabd52298c2121e2b4f865f65a2f
3913218737093418682566482155573718284197216742227640291525554660561425225367259634232024506224504609572853863214947435420747552848450195647176368717151328793566229677543084432559077606002783859266853889209627466067814270869
97742487143481404761243684585807844231228195422215358880803742773619184292201258989491042771580051513668021863248569840105623573563242529647527597186345160819230069370477556436449644280742364228349303809546343146272722644188
signature
実行

図 4 文献[5]の CastBallot 入力画面

CastBallot - System6

Menu
ID
2532633
1
auth_key
enc_param
sign_param
signature
実行

図5 文献[6]の CastBallot 入力画面

CastBallot - System7

Menu
215
<pre> {"pi": "49587234745", "cipher_text": "7845942105431907114728249984058860748515872436635241722 16713944032770597889903327223331744253356471167350941585 65271438238434397424763057612199937582812605812059184910 </pre>
option
auth_key
enc_param
sign_param
signature
実行

図6 文献[7]の CastBallot 入力画面

ことでバックエンドシステムが異なっても全く同じ UI で操作を行うことができる。内部構造の違いにより投票実行の段階で各システムが行う動作や必要なパラメータは異なるが、不足なくパラメータを付与できる Web API の設計により、ユーザは同じ操作感で投票システムを操作することができる。

ユーザからの視点ではアプリケーション内部の変化に影響を受けることは無く、用いられているバックエンドのことを知らずとも簡単に利用が可能となる。他方のフロントエンド開発者はバックエンドの投票システムに合わせて提案したフレームワークを用いれば、少ないコード記述量でシステム変更に対応できる。各投票システムのインタフェースの違いはユーザの行う操作の数、つまり実行する Web API の数の差とそれに付与するパラメータの数の差である。各 API のパラメータの量については、必要なものは入力し不必要なものは予め空文字になるようにしておくことで、システムごとのパラメータの数の変化量を無くすることができる。記述の変更量について、UI の HTML, CSS, 各 API 操作に必要な入力画面の実装部分はアプリケーションのデザインに当たるので今回は考慮しない。つまり、用いる Web API の数の差のみがフロントエンド開発者がシステムを変更する際にコードを書き換える違いとなる。各 API に必要なコード行数と、バックエンドシステムごとの

コード記述量の比較を表 3 に示す。コード行数は空行を除いて示している。

投票システムの変更に伴うコード記述量の差をまとめた表 4 からは、先に述べたようにそのシステムで操作する API の数のみに依存し、あわせて各 Web API が有するパラメータの数で決まっている。その最大値は 47 行であり、全体の高々 24% の変更量だけでバックエンドシステムを変更できる。

オンライン投票システムは暗号方式やブロックチェーンの構造上操作する手順やパラメータは異なるものの、フロントエンド開発者は提案するフレームワークを用いることでコードの記述を大幅に削減しながら、それらを簡単に使い分けることができる。

6. おわりに

オンライン投票システムでは、投票者の UI となるフロントエンドの Web アプリケーションの操作性の良否は投票率に影響すると考えられる。本論文ではオンライン投票におけるユーザビリティの向上を目的として、任意の投票システムに適用可能な Web API フレームワークを提案した。調査した範囲において一般化できたことを確認しているが、その他のシステムやブロックチェーンを用いていない投票システムに適用可能かは引き続き検討したい。また、フロ

表3 各投票システムに必要な Web API の数とフロントエンドのコード記述量

使用 API	行数	(共通)	(個別)	パラメータ数	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
PreSignUp	13	6	7	2	✓	-	✓	-	-	✓	-	-	✓
SignUp	25	6	19	8	✓	✓	✓	✓	✓	✓	✓	-	✓
GetParam	11	6	5	1	-	✓	-	-	-	✓	-	-	✓
PrepareBallot	15	6	9	3	✓	✓	✓	✓	✓	-	✓	✓	✓
RequestSign	13	6	7	2	-	✓	✓	-	-	-	-	-	-
SignUp	15	6	9	3	✓	✓	✓	-	-	-	-	✓	-
CastBallot	21	6	17	7	✓	✓	✓	✓	✓	✓	✓	✓	✓
CheckBallot	6	6	0	0	✓	-	✓	✓	✓	-	✓	✓	✓
CheckResult	6	6	0	0	✓	✓	✓	✓	✓	✓	✓	✓	✓
全コード記述量					181	186	197	157	157	150	157	157	177

ントエンドの Web アプリケーションに対して統一的なインタフェースを提供する本論文の成果を活用した、Web アプリケーションの UI について検討することが今後の課題である。

参考文献

- [1] Adiputra, C.K., Hjort, R., and Sato, H.: A Proposal of Blockchain-Based Electronic Voting System, 2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4). IEEE, p. 22-27(2018).
- [2] Gajek, S., and Lewandowsky, M.: Trustless, Censorship-Resilient and Scalable Votings in the Permission-based Blockchain Model, IACR Cryptol. ePrint Arch., 2019, 617.
- [3] Liu, Y., & Wang, Q.: An E-voting Protocol Based on Blockchain, IACR Cryptol. ePrint Arch., 2017, 1043.
- [4] McCorry, P., Shahandashti, S.F., and Hao, F.: A Smart Contract for Boardroom Voting with Maximum Voter Privacy, International Conference on Financial Cryptography and Data Security, Springer, Cham, pp. 357-375(2017).
- [5] Yu, B., Liu, J.K., Sakzad, A., Nepal, S., Rimba, P., Steinfeld, R., and Au, M.: Platform-independent Secure Blockchain-Based Voting System, International Conference on Information Security, Springer, Cham, pp.369-386(2018).
- [6] Chaieb, M., Yousfi, S., Lafourcade, P., and Robbana, R.: Verify-Your-Vote: A Verifiable Blockchain-Based Online Voting Protocol, European, Mediterranean, and Middle Eastern Conference on Information Systems, Springer, Cham, pp. 16-30(2018).
- [7] Lee, J., Choi, J., Kim, J., & Oh, H.: SAVER: Snark-friendly, Additively-homomorphic, and Verifiable Encryption and decryption with Rerandomization, IACR Cryptol. ePrint Arch., 2019, 1270.
- [8] Lai, W., Hsieh, Y., Hsueh, C., and Wu, J.: DATE: A Decentralized, Anonymous, and Transparent E-voting System, 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), IEEE. p. 24-29(2018).
- [9] Gao, S., Zheng, D., Guo, R., Jing, C., & Hu, C.: An Anti-Quantum E-Voting Protocol in Blockchain With Audit Function, IEEE Access, 2019, 7, 115304-115316.

表4 バックエンドシステムを切り替える際のコード変更量

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[1]		-5	-16	+24	+24	+31	+24	+24	+4
[2]	+5		-11	+29	+29	+36	+29	+29	+9
[3]	+16	+11		+40	+40	+47	+40	+40	+20
[4]	-24	-29	-40		±0	+7	±0	±0	-20
[5]	-24	-29	-40	±0		+7	±0	±0	-20
[6]	-31	-36	-47	-7	-7		-7	-7	-27
[7]	-24	-29	-40	±0	±0	+7		±0	-20
[8]	-24	-29	-40	±0	±0	+7	±0		-20
[9]	-4	-9	-20	+20	+20	+27	+20	+20	