スマートコントラクト（エンターテイメント：ファン投票システム）

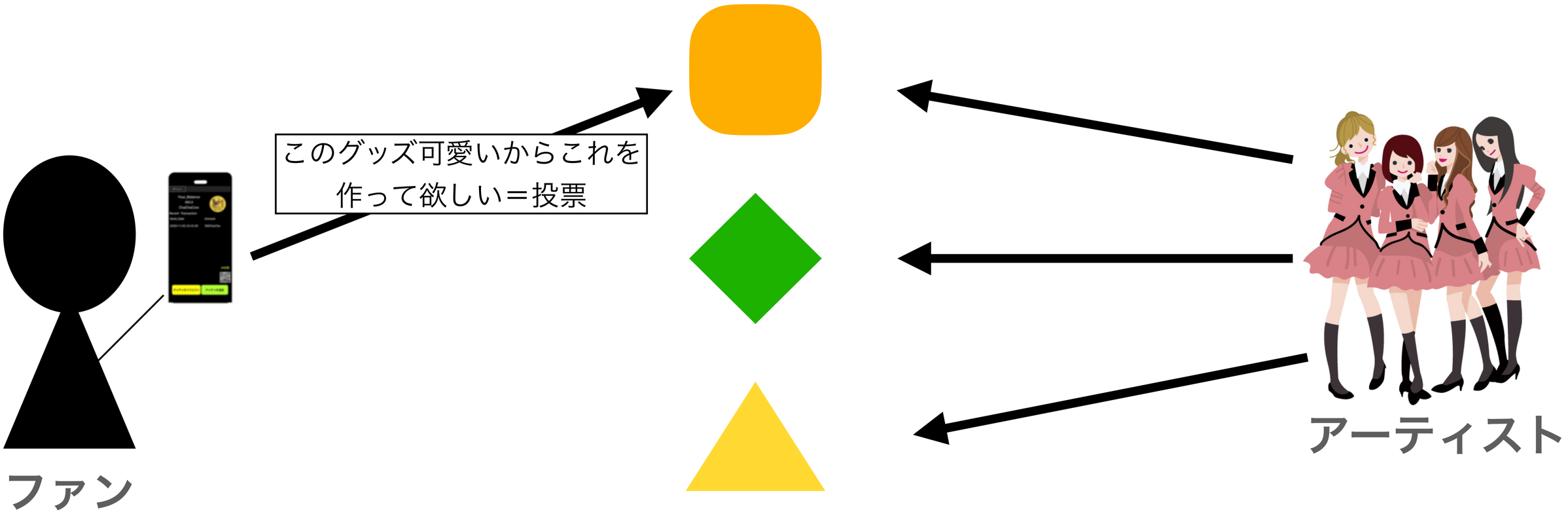
開発背景：

自分がオタ活やってて思いついた！新たなアーティストの応援の形を全てのファンで

共有したい！＝アイドルやアーティスト応援の具現化

（アーティストの運営元に採用されるかは別の話）

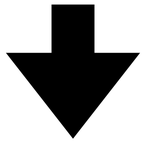
具体例：グッズ投票）　　リリース予定投票対象（アーティスト考案）



スマートコントラクト（エンターテイメント：ファン投票システム）

今までの自分が思う投票の懸念：
SNS投票：匿名で誰が投票してるか管理できない！＝コミュニティ投票で使うのは厳しい
（投票を求めたい対象以外からの投票が入っている可能性あり）

中央集権的サーバー投票の問題：
①サーバーハッキングなどで個人情報が抜かれるリスクとデータが改竄されるリスク
（特定の人に投票権を配る場合＝例）ファンクラブ,大規模コミュニティイベント（コミケetc））
②アクセス集中でシステム停止リスク
（配信ライブの時にアンコールファン投票でサーバー圧迫のため投票できなかった経験あり）



分散型のID(DID)でブロックチェーンを使って改竄できない価値のあるデータを
スマートコントラクトで制御をすることで実現する

スマートコントラクト（エンターテイメント：ファン投票システム）

考えられる使用用途

配信ライブ

グッズ決め投票

ランキング投票

①新型コロナに伴い配信ライブになり既存アンコール投票システムだと
　サーバー負荷増大により投票システムが平等に機能しないことがある

②グッズ作成の新しい形・・・全部ではなくグッズデザインを一部ファンで投票して決めるのもあり？

③マンガや小説など一定規模のコミュニティイベントの投票などでの利用・・・オンラインコミケとかも出てきそう

スマートコントラクト（エンターテイメント：ファン投票システム）

Why miyabi?

①アンコール投票のシステムとして使用する場合

リアルタイムでの高速処理が求められる！

⬇

TPS 4000 Tx/sec→TPM 240000 Tx/sec（24万件/分）

この処理数がリアルタイムでの投票システムに向く

②サーバー負荷に関して：

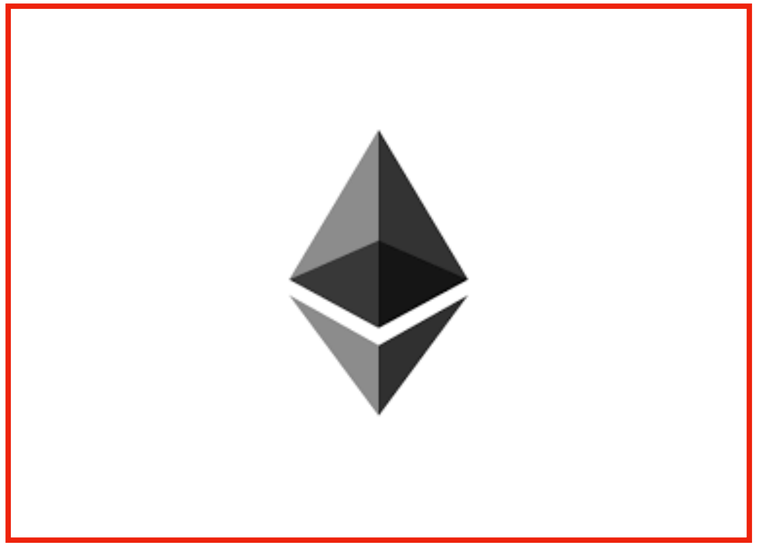
マイナンバーとかの行政システムに提案できるレベルのブロックチェーンなので問題はないと思われる！ブロックチェーン はメモリープールで処理を待機できる

➡このため一回入れた投票は覆されない（キャンセルにならない）

＊規約違反の投票はキャンセルになります

スマートコントラクト（エンターテイメント：ファン投票システム）

TPS比較（トランザクション処理速度）



TPS 25 case/sec

https://blog.blockchain.bitflyer.com/n/n749874d49464

(bFB:金光さんブログ参考)

TPS:100~900 case/sec

諸所報告がばらつき有り

TPS:4000 case/sec

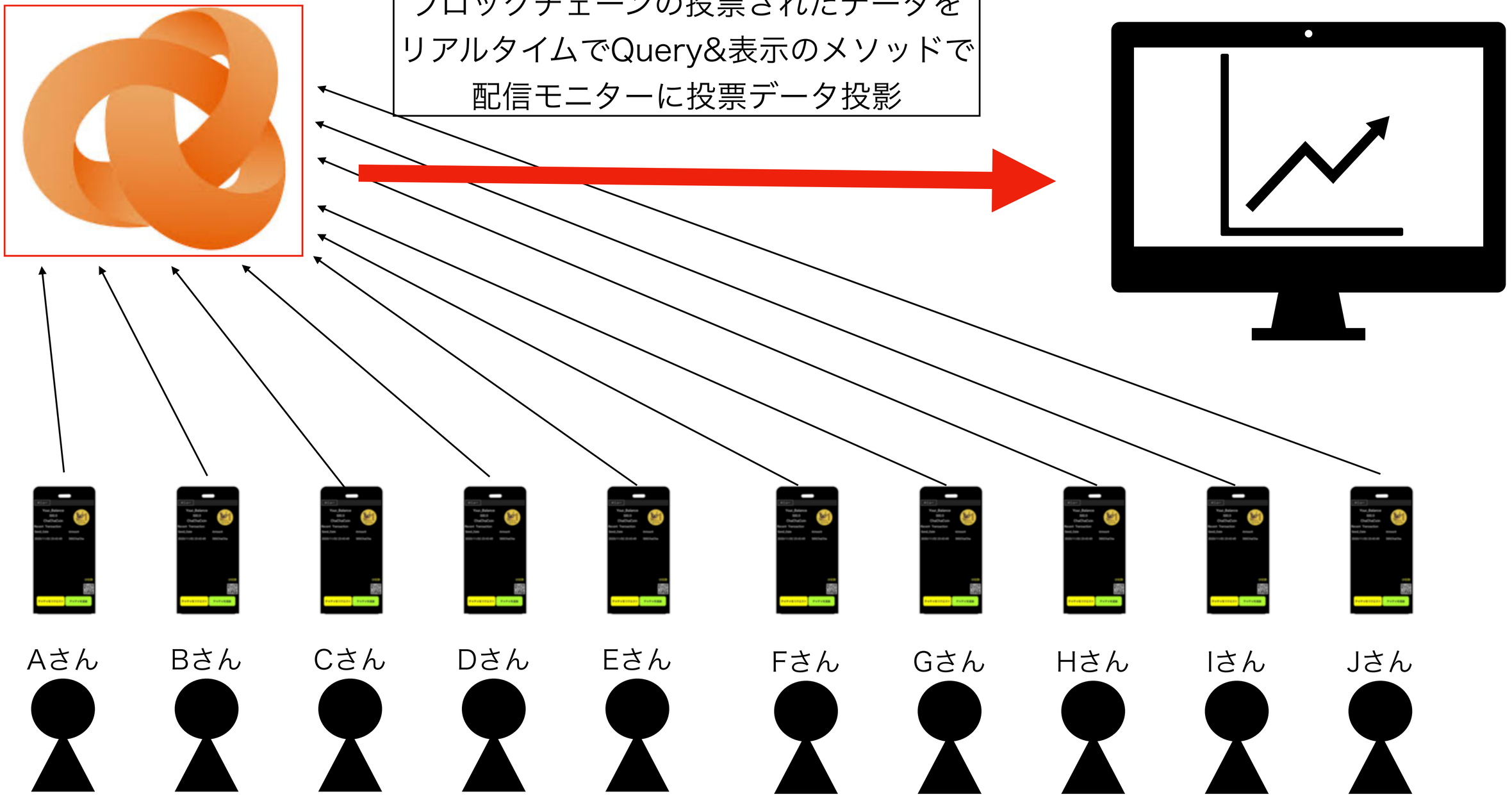
https://blockchain.bitflyer.com/miyabi/

諸所いろいろな報告があり実測値を測っていないため（自分調べ）

miyabiに関してはチェーン設計をいじることでさらに追加でTPSを早くできる！

伺ったことがあります！

エンターテイメント：ファン投票システム配信ライブ用途を例に！



ChaChaCoin Webアプリケーション構成図

フロントエンド

バックエンド(ASP.NETCORE)

Mobile Client App

モバイルクライアント

使用プログラム言語

・HTML

・CSS

・Javascript(Jquery)

WEBアプリケーション

Rest API

C#(Miyabi SDK付)

Miyabi(ブロックチェーン)

SC

投票AssetTable

・アプリユーザ投票

・運営元の投票アドレスの登録

・投票データの集計(運営)

投票条件の管理（binary）

投票参加者の定義

（イベントへの参加登録）

投票対象の数をイベント毎に登録（集計用）

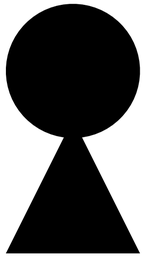
key:イベント名

value:投票ターゲット数

スマートコントラクト設計図（エンターテイメント：ファン投票システム）

スマートコントラクトに投票！（アンコール投票）

ライブに参加登録



参加者として公開鍵情報
をバイナリーデータとして登録
（ライブの特別ページなどで自身の公開鍵(ID)を登録！）

投票用アセットテーブル

Smart
Contract

Aさん　Bさん　Cさん　Dさん　Eさん

参加者管理用テーブル

バイナリで下記情報が記入

Key:公開鍵　Value：任意の値

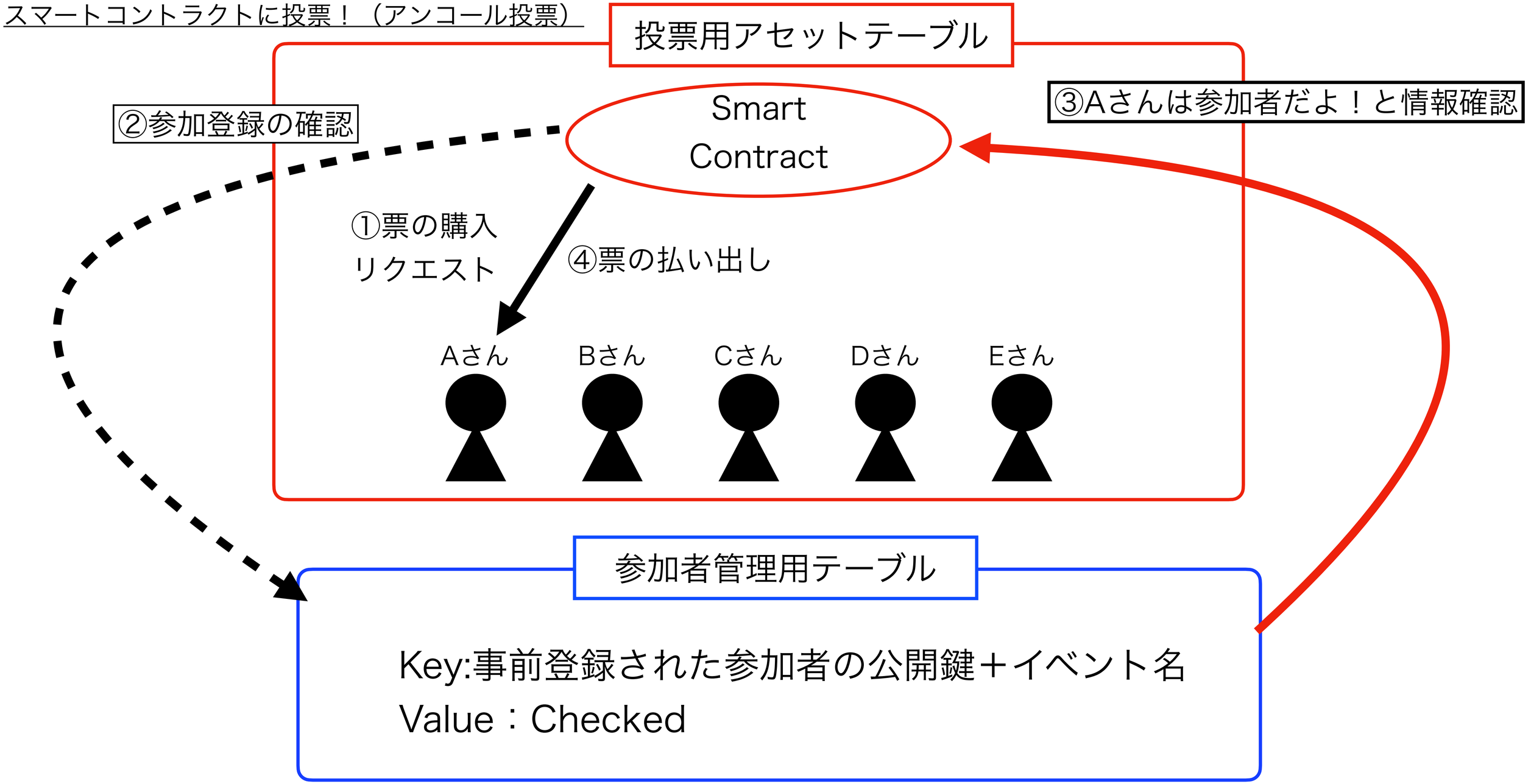
GetValue(Key,value)

Key:公開鍵+イベント名　Value：“checked”

GetValue()の値確認メソッドで値があるかどうかでイベントや投票参加者かどうかを判断

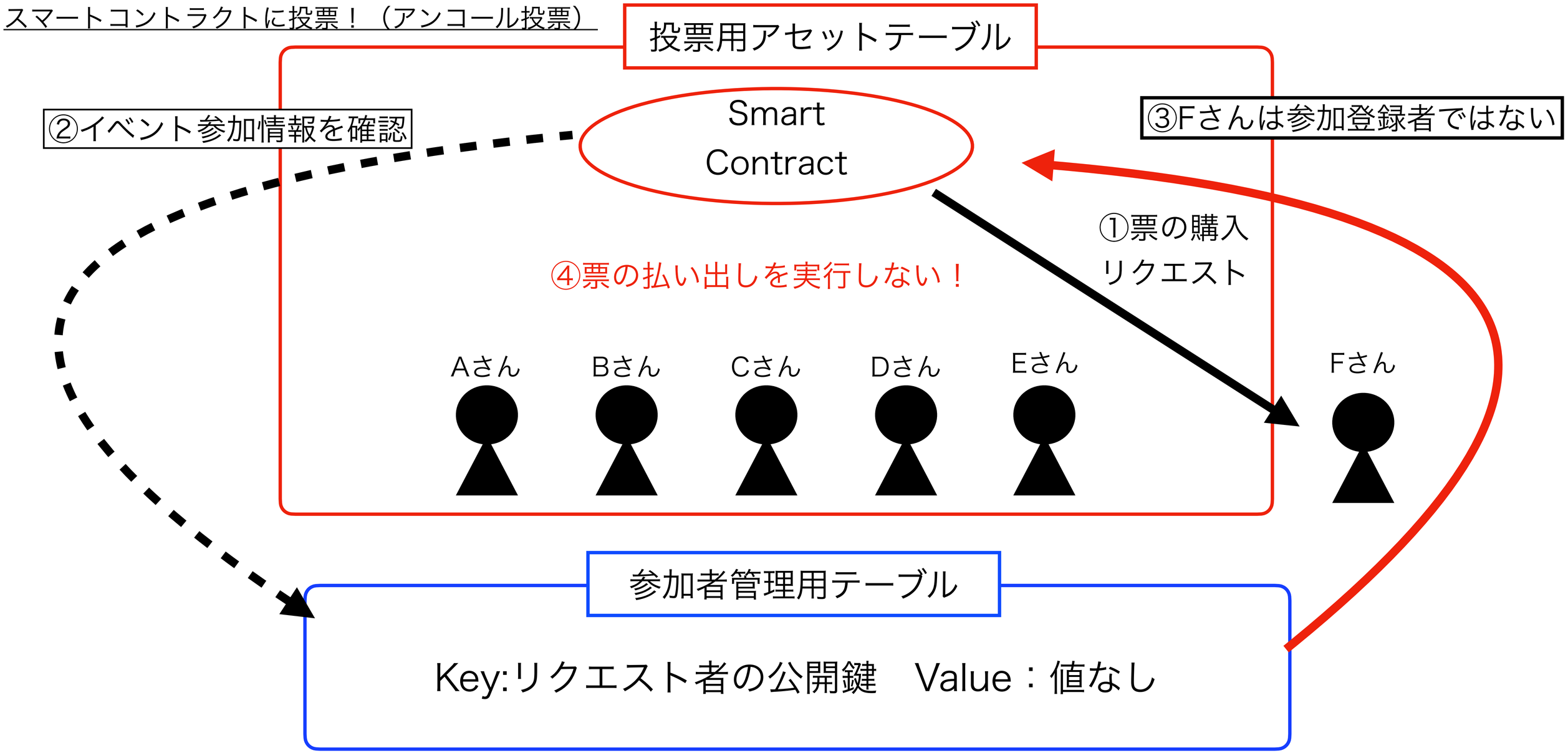
スマートコントラクト設計図（エンターテイメント：ファン投票システム）

スマートコントラクトに投票！（アンコール投票）



スマートコントラクト設計図（エンターテイメント：ファン投票システム）

スマートコントラクトに投票！（アンコール投票）



スマートコントラクト設計図（エンターテイメント：ファン投票システム）

スマートコントラクトに投票！（アンコール投票）

投票運営元

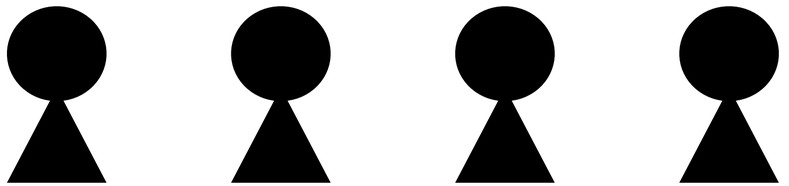
イベントの投票ターゲット情報を登録
をバイナリーデータとして登録

このデータはコントラクトのテーブルの
データ集計用

投票用アセットテーブル

Smart
Contract

Bさん　Cさん　Dさん　Eさん



参加者管理用テーブル

バイナリで下記情報を登録

Key:イベント名　Value：投票ターゲットの数

例）3つの場合

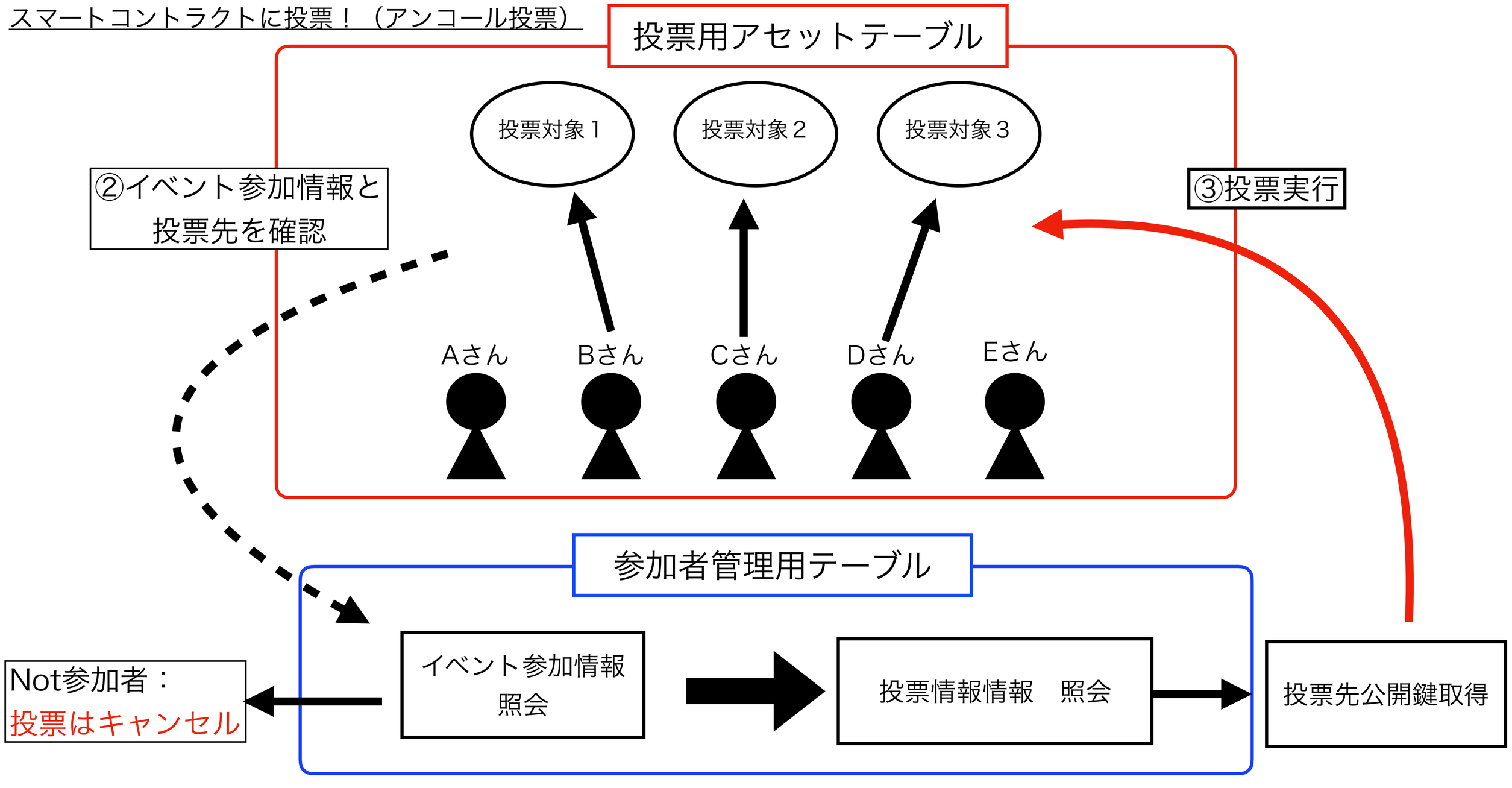
key:イベント名+”1” value:対象公開鍵その１

key:イベント名+”2” value:対象公開鍵その２

key:イベント名+”3” value:対象公開鍵その3

スマートコントラクト設計図（エンターテイメント：ファン投票システム）

スマートコントラクトに投票！（アンコール投票）



エンターテイメント：ファン投票システム配信ライブ用途！

チャチャコイン

（アセットテーブル）

例）100チャチャ ＝１票

複合トランザクション
チャチャコイン を票にスワップ

投票用テーブル

ターゲット

投票

これはサーバーサイドのスマコン Invokeのmethod内に複合トランザクションを実装

あくまで、処理の流れでUI/UXはボタン一個で簡単に投票できる形を実現します!

エンターテイメント：ファン投票システム配信ライブ用途！

・票の配布はスマートコントラクトからのみ行うようにする

スマートコントラクトからにする理由は以下

①票は正しい参加者に配るため（複合トランザクションで実装するため）

　秘密鍵も必然で署名検証が入っている