Augur: a Decentralized, Open-Source Platform for Prediction Markets

Dr. Jack Peterson & Joseph Krug
www.augur.net

Augur is a trustless, decentralized platform for prediction markets. It is an extension of Bitcoin Core's source code which preserves as much of Bitcoin's proven code and security as possible. Each feature required for prediction markets is constructed from Bitcoin's input/output-style transactions.

I. INTRODUCTION

A prediction market is a place where individuals can wager on the outcomes of future events. Those who forecast the outcome correctly win money, and if they forecast incorrectly, they lose money. People value money, so they are incentivized to forecast such outcomes as accurately as they can. Thus, the price of a prediction market can serve as an excellent indicator of how likely an event is to occur [1, 2]. *Augur* is a decentralized platform for prediction markets.

Our goal here is to provide a blueprint of a decentralized prediction market using Bitcoin's input/output-style transactions. Many theoretical details of this project, such as its game-theoretic underpinning, are touched on lightly or not at all. This work builds on (and is intended to be read as a companion to) the theoretical foundation established in [3].[1]

A. Why decentralize?

Historically, the actions required of a prediction market – accepting wagers, deciding on the outcome of an event, then paying the wagers back out according to the results – have been centralized. The simplest approach to aggregating wagers is to have a trustworthy entity maintain a ledger. The simplest way to determine the outcome of an event is to get the answer from a wise, impartial judge, whom all participants in the market trust.

Upon closer inspection, these straightforward answers fray at the edges: who is this 'someone' who is trustworthy enough to maintain a ledger for everyone else? Who is the judge that every participant trusts? And, even if such paragons were found and agreed upon, how could participants be certain they will *remain* trustworthy once they are granted more power? "Opportunity makes the thief", after all. And, of course, "Power corrupts. Absolute power corrupts absolutely." [4]

In practice, a larger issue is that these trusted entities represent single points of failure. Prediction markets are often disliked by powerful interests. As the experience of centralized prediction markets – such as InTrade and TradeSports – over the past decade has shown, if governments or special interest groups want to shut down a website, they will find a way: InTrade, after all, was an Irish company, shut down as a result of the U.S. Commodity Futures Trading Commissions actions [5, 6]. Even the Defense Advanced Research Project Agency and Central Intelligence Agency were forced to end their foray into prediction markets as a result of Congressional interference [7].

Contrast this with the history of Bitcoin [8]. Bitcoin is a cryptographic currency and payment platform that has found an enemy in powerful nation-states and financial institutions. Nevertheless, Bitcoin has thrived: as of November 2014, it has a market capitalization of over five billion U.S. dollars, and it is the anchor of a thriving ecosystem of startups, trade, and technological innovation. This is possible because Bitcoin is decentralized: once it was released, it could not be shut down. Even its pseudonymous creator, Satoshi Nakamoto, cannot stop the cryptocurrency. This is by design: he (or she or they) purposely did not make himself Bitcoin's single point of failure.

Augur is a decentralized prediction market platform. The Augur Project's goal is to revolutionize prediction markets, and, in doing so, change the way that people receive and verify 'truth'.

B. Strategy

Augur is built as an extension to the source code of Bitcoin Core.[2] It includes the features – the betting and consensus mechanisms – required by prediction markets. However, it is intentionally the same as Bitcoin in all other respects, to capitalize on Bitcoin's security and scalability. Our intention is to use the 'pegged sidechain' mechanism to make Augur fully interoperable with Bitcoin [9]; if this technology is not available, we will instead use our own internal currency as the store of value.

C. Tokens

Augur has three types of tokens or *units*. To keep track of these units, every transaction input and output `value`

[1] Additional details and discussions on the theory behind Augur can be found at `www.truthcoin.info`.

[2] Alternatives, such as smart contract-based implementations, are discussed in Appendix B.

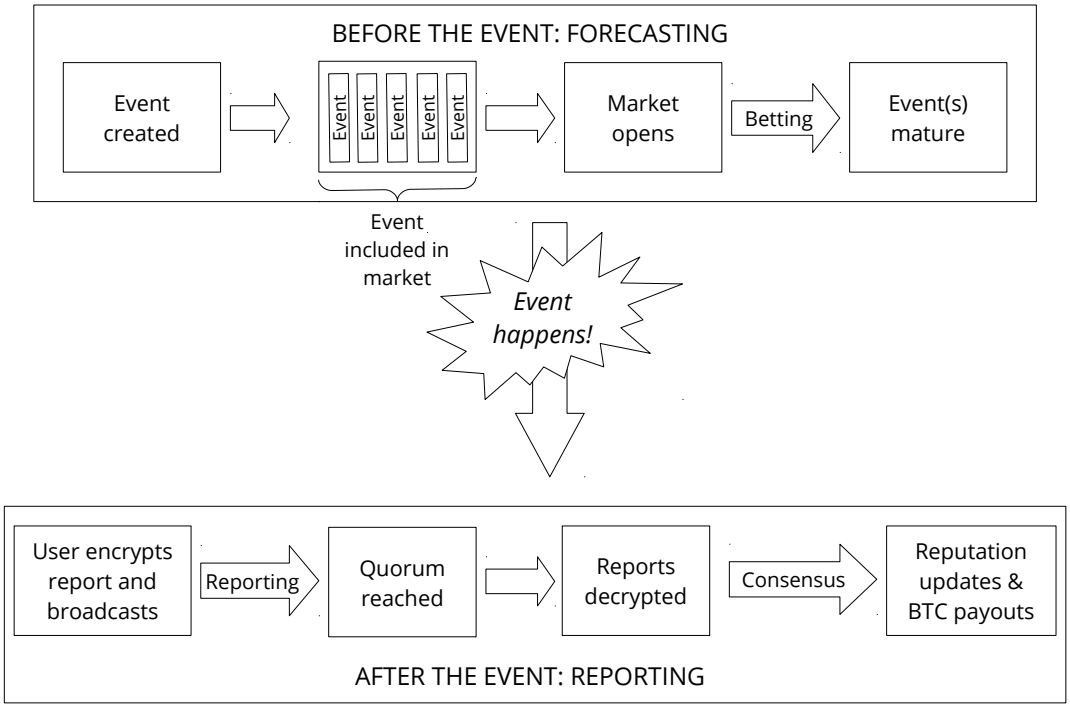

FIG. 1. Simplified outline of the actions in a prediction market, separated into before and after the event.

fields is accompanied by a `units` field.

Although there are three types of tokens, users have a single cryptographic private key.[3] Augur addresses are simply base-58 hashes of their cryptographic public keys, to which users can send tokens. The different types of token are distinguished from one another solely through the 'units' field. Therefore, users sign the transaction data of a Reputation payment in exactly the same way as they would sign a Bitcoin payment.

The first token is Bitcoin. *Sidechains* allow us to transfer Bitcoin by creating a transaction on the Bitcoin blockchain that locks up the Bitcoin in an address [9]. Then a transaction is made on our blockchain with an input that has a cryptographic proof that the lock was made on the Bitcoin blockchain along with the genesis blockhash of its originating blockchain. A simplified payment verification (SPV) scheme is used to verify these locks. A SPV has a list of block headers for proof-of-work and cryptographic proof that the locking transaction was created on one of the blocks in the header list.

A key feature of Augur is tradeable *Reputation*. The total amount of Reputation is a fixed quantity, determined upon the launch of Augur. Holding Reputation entitles its owner to report on the outcomes of events, after the events occur. Reputation tokens are similar in other respects to Bitcoins: they are divisible to eight decimal places, they are accounted for by summing over unspent transaction outputs, and they can be sent between users.

A significant way that Reputation differs from Bitcoin is that Reputation is not intended to be a stable source of value: Reputation tokens are gained and lost depending on how reliably their owner votes with the consensus. Reputation holders are obligated to cast a vote every time the network 'checks in' with reality (by default, this is every 8 weeks). Another significant difference is that Reputation is not mined. Instead, Reputation will be distributed by means of an auction, as proposed in [3]. This solves two problems. First, it will provide funding to complete the development of the Augur network. Second, it bootstraps the network with an initial group of Reputation-holders who are invested in the network's success.

Using a Bitcoin sidechain is a straightforward way to purchase shares. Users will simply transfer Bitcoins to the Augur network and use them to buy shares of a prediction. However, this poses a problem. If two people want to make a wager on whether event X will occur within a year from now, they would be exposed to the price volatility of Bitcoin. A potential solution is a seigniorage modeled coin. This would create a new

[3] Augur uses the same public/private key cryptography system as Bitcoin: private keys are random 256-bit numbers, with a valid range governed by the secp256k1 ECDSA standard.

cryptocurrency, say 'USDCoin', and put them on a few exchanges to be traded. If the price were above one dollar, the coin algorithm would print more coins; if below a dollar, the algorithm would discontinue printing. This allows the price of the coin to stay relatively close to a dollar.[4]

Another pertinent question is how to distribute the coins used to buy shares of predictions. A few options are available: 1) having a Bitcoin sidechain be the mechanism to buy shares, 2) using a seigniorage model to create a 'Cashcoin' or 'USDcoin' that maintains its value with respect to the US Dollar, and 3) using a hybrid approach of 1 and 2.

In our view, the best option is to sidechain Augur to Bitcoin, and have a new seigniorage based coin to allow users to both wager with Bitcoin, *and* another currency which would be used more for longer term wagers in which volatility would become an issue. This provides the benefits of allowing interoperability with Bitcoin holders as well as a currency suited for holding shares of predictions.

II. PREDICTION MARKET CREATION

The life of a prediction market can be separated into two principal 'phases' (Fig. 1): before and after the event. To walk through the life cycle of a typical prediction market, consider a user, Joe, who wants to wager on the outcome of the 2016 U.S. Presidential Election.

A. Event Creation

Joe turns on his Augur program, which plugs him directly into the peer-to-peer network, and clicks on the 'politics' group. This brings up a list of all active political prediction markets, along with their current prices. No one has created a market for the 2016 election yet, so Joe decides to take the initiative. He clicks on 'Create Event'. Here, he enters the necessary information about this event (Fig. 2):

- Description: a short description of the event.
- Type: binary (yes or no), scalar (numeric), or categorical
- Range of valid answers: yes/no/maybe for boolean, the minimum and maximum allowed values for scalar.
- Topic: the category the event falls into. Topics are created by users, and can be as fine-grained as users want them to be – examples might be 'politics', 'science', 'funny cat videos', 'serious cat videos', and so on.
- Fee: a small fee (on the order of 0.01 Bitcoin) required to create an event. Half of this fee goes to users who Report on the results of the event, after it happens (details in Section IV).
- Maturation: duration of the forecasting phase. This is the period where users buy and sell shares of the market. In Augur, time is marked by the block interval, which is expected to be constant, on average. Users have the option of entering a 'block number' (which is exact, but not intuitive) or an 'approximate end time' (which is not quite exact, since the block interval is not always *exactly* the same length, but is much more intuitive).
- Creator's address: the address of the event's creator.

CreateEvent Transaction

Joe pays a fee of 0.01 Bitcoin to create this event. As shown in Fig. 2, this is the CreateEvent transaction's only input. The CreateEvent transaction also has a single output, which contains the event's data.

The output's data includes the event's automatically assigned unique ID, which is a 160-bit hash[5] of the other event output fields. To prevent accidental duplicate event creation (due to network latency), the same user is not permitted to create the same event again, with the same expiration date and description. The value of this output is the user's event creation fee. The output can then be spent by including the event in a CreateMarket transaction (see below).

These inputs and outputs are combined into a Bitcoin-style transaction:[6]

```
{
    "type": "CreateEvent",
    "vin": [
        {
            "n": 0,
            "value": 0.01000000,
            "units": "bitcoin",
            "scriptSig": "<Joe's signature>
                          <Joe's public key>"
        }
    ],
    "vout": [
        {
```

[4] See Nubits for an example.

[5] SHA256 followed by RIPEMD160, usually referred to simply as 'hash-160'.

[6] Like Bitcoin, Augur's transactions are broadcast and stored in serialized form. A deserialized, rawtransaction-like JSON format is shown here for readability. For brevity, fields such as `locktime` and `sequence` are omitted from our examples when they are not directly relevant. Also, for Script fields (such as `scriptSig` and `scriptPubKey`) only the `asm` subfield – i.e., the input to `CScript()` – is shown.

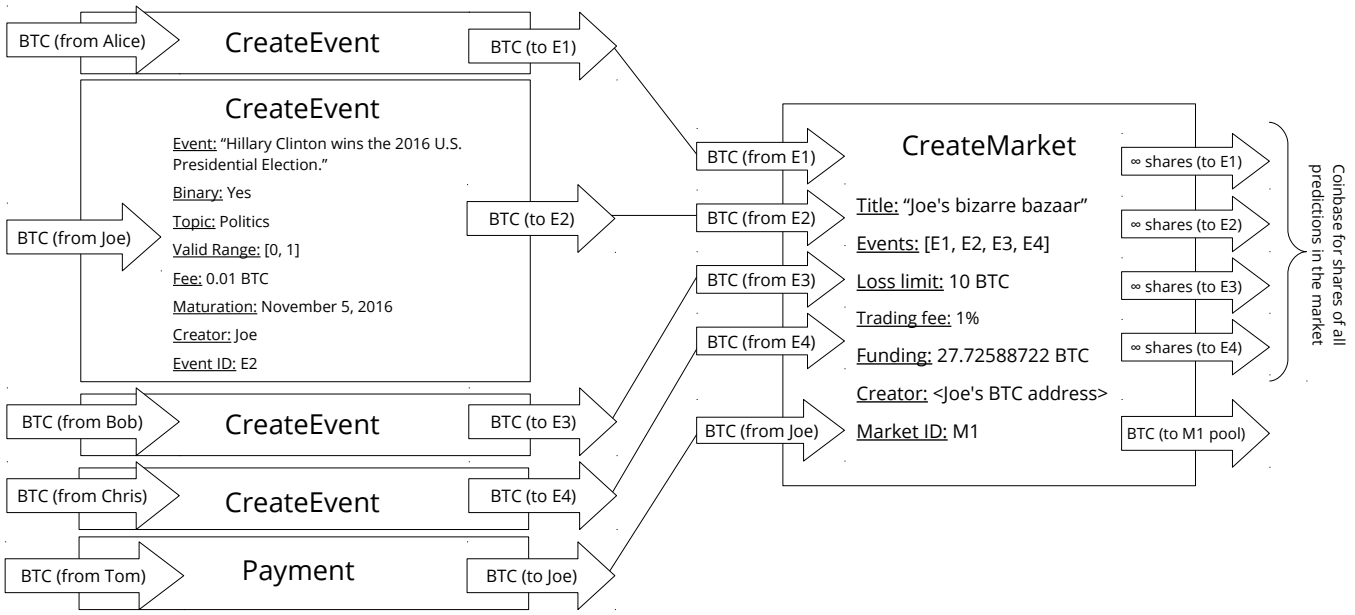

FIG. 2. CreateEvent transactions are payments from the user's address to a newly-generated event address. A CreateMarket transaction is a coinbase for shares of all the events contained in the market. Any user who has created one or more Events can incorporate their event(s) into a new market. In this simplified diagram, arrows represent inputs and outputs, and lines represent spending an unspent output. Note that events and markets have been given simplified IDs (E_1, E_2, etc.) here to make the figure easier to read; actual event and market IDs are 160-bit hashes of the event or market data fields, respectively.

```
            "n": 0,
            "value" : 0.01000000,
            "units": "bitcoin",
            "event": {
                "id": "<event hash>",
                "description": "Hillary Clinton
                    wins the 2016 U.S. Presidential
                     Election.",
                "branch": "politics",
                "is_binary": True,
                "valid_range": [0, 1],
                "expiration": 1478329200,
                "creator": "<Joe's address>"
            },
            "address": "<base-58 event ID>",
            "script": "OP_DUP
                       OP_HASH160
                       <event hash>
                       OP_EQUALVERIFY
                       OP_MARKETCHECK"
        }
    ]
}
```

The CreateEvent output's Script is keyed to the Event's ID:[7]

```
OP_DUP
OP_HASH160
<event hash>
OP_EQUALVERIFY
OP_MARKETCHECK
```

Since the event ID is the hash-160 of the Event's data, the CreateMarket transaction's matching input Script is:

```
<market hash>
<market data>
<event data>
```

The raw event data is supplied by the user to be included in a market, then is pushed onto the stack during the subsequent CreateMarket transaction. `OP_MARKETCHECK` first calculates the hash-160 of the market's fields (excluding the market ID), then verifies that this hash matches the actual market ID. The CreateEvent transaction's output is automatically sent to the event's *address*, which is a base-58 string derived from the event ID in the same way that Bitcoin addresses are derived from the public key's hash-160.

B. Market Creation

Once the event is created, he then clicks on 'Create Prediction Market', where he can include his new Event in a prediction market. Here he enters other information about the market:

- Title: label/brief description of the market.
- Events: list event IDs to be included in the market.

[7] Transaction Scripts are conventionally written left-to-right. However, we use a top-to-bottom format here for the sake of readability.