Using the Bitcoin Blockchain for secure, independently verifiable, electronic votes.

Pierre Noizat - July 2014

The problem with proprietary voting systems

Existing electronic voting systems all suffer from a serious design flaw: they are proprietary, i.e centralized by design, meaning there is a single supplier that controls the code base, the database, the system outputs and supplies the monitoring tools at the same time.

The lack of an open source, independently verifiable output makes it difficult for such centralized systems to acquire the trustworthyness required by voters and election organizers.

This design flaw is therefore limiting electronic vote applications at a time when growing computer literacy and usage should in fact foster their widespread adoption.

Without an easy, free access to effective, secure electronic voting technology, political participation is limited to on site elections that are few and far between, given the high setup cost of election preparation, supervision and post-election operations.

Electronic vote is not meant to replace traditionnal elections but can provide a much needed complementary voting method.

Open source, free software electronic transaction and voting systems

Since the invention of the Bitcoin protocol [1], the Bitcoin network has grown to become today the most powerful distributed computing network on earth, providing a very secure, free software, database infrastructure to store electronic transaction data of any kind.

The complete Bitcoin transaction database is referred to as the Blockchain because bitcoin transaction records are paginated in blocks appended to the database every ten minutes on average.

An electronic vote is essentially an electronic transaction whereby a voter, given some voting credits, will spend them in favor of one or more candidate recipients.
Candidate recipients can be people like in a presidential election or options to chose from like in a referendum election.

This paper describes how to leverage the availability of the Bitcoin blockchain as a secure transaction database, to log votes and audit vote results. The proposed method involves Merkle Trees [2] for voters list verification and block explorers for vote count check.
The solution protects the privacy of the vote by using a simple form of Bitcoin Pay-to-Script-Hash scripts [3] [4] with 2-of-3 multi-signature addresses.
I have developped a demo voting application based on my proposal to assess usability and performance but testing has only begun and is far from over: I am sharing here early results to get the discussion going. The application is deployed on the website unchain.voting.

Here's how the whole process unfolds in 5 steps.

1) Pre-election, know the list of candidates :

Each candidate (C) assigns a public key **KeyC** to the voter and publishes everywhere a Bitcoin address like 1Martin.., that is a vanity address [5] prepared by the organizers for candidate Martin.
For the secrecy of each ballot, each KeyC is different for each voter, so the candidate must generate N public keys KeyC for N voters.

Each candidate should be required by the organizers to publish the Merkle Root of a hash tree comprising all of his public keys KeyC.
Alternatively, the voting application can generate the keys KeyC on behalf of the candidate although this option could be avoided to reduce the need to trust the organizers .

Using the voting application, each voter should be able to check her branch of the hash tree, linking the key keyC to the root.

2) Check the list of voters

Each voter (B) is also assigned by the election organizers (e.g an association A) a public key **KeyA** and a nonce value that can be used by the voter as a secret access key to log in to her account.
These access keys will be sent securely by the organizers to each voter

The voters list can be represented in a Merkle Tree (Fig. 1): the application gives each voter the list of interior hashes linking her public key KeyA to the Merkle Root.

With the Merkle Tree published by the application and with the standard computation formulas below, the voters list can be independently verified by all parties because every voter can check the path from her leaf node to the Merkle Root and make sure that the number of leaf nodes is equal to the number of voters.

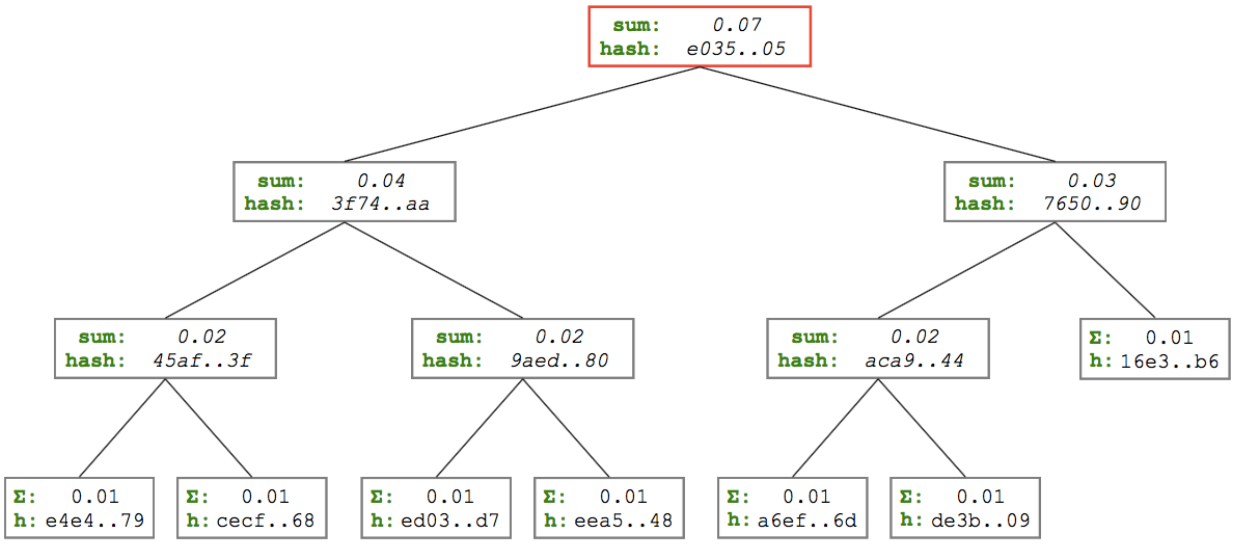

Fig. 1 - Merkle Tree of a 7-voter list

This is how interior hashes are calculated as 64-character string, using Ruby scripting:

sha256_base64 "#{sum_of_upper_left_and_right_nodes}|#{upper_left_hash}|
#{upper_right_hash}"

Likewise, leaf node hashes are calculated as follows:

sha256_base64 "#{voter_name}|#{voter_credit_balance}|#{voter_nonce}"

The voter credit balance can be intialized to one in the simple case of a presidential election or to n if the voter can chose n of m candidates like in the election of n board members. Voter credit is updated when the voter cast her vote, spending all or part of her voting credit.

sha256_base64 is the standard method for generating the 64-character hash of a string:

sha256_base64(string) = OpenSSL::Digest::SHA256.new.digest(string).unpack('H*').first

The path to the root for one of the voter in the above example tree would look like this:

Voter name: stephan

Voter credit balance: 0.01 ฿

Account Nonce (voter access key): 958a78b68c8f53f89c3433c1fde98152

Account Hash (leaf node hash):

eea58cf19ed83072a9ea650e1c71a100f8f22129196cf52d1d91c4ecf68bdc48

Interior Hash #1:

9aed30f6447d11e852873d4896d43a13152773b7ba2dae65b1eb342854c9de80

Interior Hash #2:

3f74cef73f11bbc2eca69e70b985f7a3e6a69d292f4dc0fb9d0fbcdd5453dfaa

Interior Hash #3 (Merkle Root):

e035cfb7615443b4e49bc1a0bd4b16bdb28ec79ad8c409f5bbf55dea9cbf7f05

As a precautionary option, the public key KeyA should be random and should not be on the Bitcoin elliptic curve bitcoin, for instance using a sidechain, to prevent the organizers (A) from producing a valid signature.

This precaution is also independently verifiable and reduces the need to trust the organizers. However, the complexity of using a sidechain instead of the blockchain should not be underestimated.

Additionally, the application assigns a unique public key **KeyB** to the voter.

As an option, to further reduce the need to trust the organizers, KeyB can be generated independently by the voter herself. However, opening this option would entail some voter education, support efforts and less usability.

As usual, the choice for an optimal convenience/security trade-off should be guided by common sense and knowledge of the voters demographics, including computer literacy. Generally speaking, organizers should not not expect that people will be able to handle private keys themselves in a secure manner.

3) Prepare the ballot

Before election day, with the A, B and C public keys recorded by the application , the application can create a **2-of-3 multi-signature address** and a transaction sending to it a bitcoin micro-payment (about the price of a postage stamp) upon the voter confirmation of her vote.

The voter’s ballot now has a cryptographically secure representation in this transaction which is broadcast to the bitcoin network nodes by the application.

At this point, it should be noted that the voter can check independently and immediately that her vote has been submitted to the bitcoin network by looking up the multi-signature address with any block explorer such as blockchain.info or blockr.io: although unconfirmed yet, the transaction is showing 2-3 seconds at most after she confirmed her vote.

After a few hours, the ballot is securely logged in the blockchain with an increasing number of network confirmations.

The output script (destination) of her micro-payment transaction looks like this:

OP_HASH160 Hash OP_EQUAL

where **Hash** is the hash calculated by her bitcoin wallet from the following (**SerializedScript**) script:

OP_0 OP_2 KeyA KeyB KeyC OP_3 OP_CHECKMULTISIG

This transaction is compliant with BIP-16.

Each key is a concatenation of the x and y coordinates of a point. If the point is on the bitcoin Koblitz curve, then it is a Bitcoin public key and there is a corresponding private key. Coins sent to the multisignature address can be spent only if at least 2 of the keys are Bitcoin public keys. A key is represented as a 128 hexadecimal character string, each coordinate being represented as 32 bytes. Here is an example :

KeyA:

7d56d13563243998e4934e189574db4eca3b1ac1e0f20f8e136ec047a202151da60a38a561bc4debd2

8c0433c51b2f21f1df58c53d4e69c828351dfff4ce1eca

KeyB:

d3363437960978fef861678d9ae0e18544cf999892c0375e7b2856a5258d740f0dfd31d3b1547b0847c1e65f4d7228a2e7223cc18e42b4271bf7421a54f03d48

KeyC:

2fdede700db4d7c5e40cbe90d61961240e6fab869270c21d024614b83803adfc84e001d313a24910317ec3a7e73c6dcdf828acb639d8048cc4efed95b77c8eb9

Multi-signature address whose funding represents the vote of B in favor of C:

3NSxmhDPpYak4HH5BYJdsNDq5s1bE121oH

Because this address was funded, it can be found in a transaction output of the blockchain by anyone using a block explorer. More precisely, the address is derived from the 20-byte hash of the redeem script of the transaction recorded in the blockchain. The redeem script itself, a simple concatenation of the three public keys, will not be visible until the output is spent in a subsequent transaction, after election day.

The application displays a list of all the funded multi-signature addresses as an indicator of the vote participation rate.

There is no way to guess neither the voter (B) nor the candidate (C) from a multi-signature address without knowing all three public keys (KeyA, KeyB, KeyC) and knowing to whom they belong.

In particular, the candidates have no way of knowing their rankings at this point.

A voter can check that the multi-signature address representing her vote appears in the list.

In the demo application, organizers generate and fund a separate address for each voter in the pre-election period. This address (Address B) is derived from KeyB to keep things simple.
On election day, the unspent output of Address B can be used by the application to fund the multi-signature address corresponding to the voter's choice.
If multiple choices are offered in the election like in a board member election, several multi-signature addresses can be linked to the voter's account to reflect her selection, each address representing a selected candidate or chosen option.

Since we want the multi-signature address funding transactions to show up in the blockchain, preferrably before the vote count, organizers must consider the settings of transaction fees. If transaction fees were set too low, a funding transaction could end up being left in limbo, stuck in the memory pool of the miners and ultimately dropped.

According to the Bitcoin Wiki [6], a transaction may be sent without fees if these conditions are met:

- *It is smaller than 1,000 bytes.*
- *All outputs are 0.01 BTC or larger.*
- *Its priority is large enough, where*

priority = sum(input_value_in_base_units * input_age)/size_in_bytes

The typical size of a transaction from one standard address to one multi-signature address is around 223 bytes.

As of core client version 0.3.21, « *transactions needed to have a priority above 57,600,000 to avoid the enforced limit. This threshold is written in the core client code as COIN * 144 / 250, suggesting that the threshold represents a one day old, 1 btc coin (144 is the expected number of blocks per day) and a transaction size of 250 bytes.*
So, for example, a transaction that has 2 inputs, one of 5 btc with 10 confirmations, and one of 2 btc with 3 confirmations, and has a size of 500bytes, will have a priority of
*(500000000 * 10 + 200000000 * 3) / 500 = 11,200,000* »

Therefore, the recommended setup process to avoid paying fees to the miners is to use 0.01 BTC as the transaction amount for each voter and to fund Address B at least one day before election day. The challenge is about paying minimal mining fees while ensuring that the transactions will be confirmed.
It should be anticipated that, in the future, miners fees will evolve from rule-driven fees to market driven fees, turning the above fee-avoiding recommandation into a fee-minimizing strategy.
Because the time period when the coins are needed equals roughly two days, the amount of the transaction should be scaled according to the number of voters: as the amount is driven lower by a large number of voters, mining fees will start to kick in. As a result, organizers may be able to find an optimal trade-off combining financial costs and mining fees.
In fact, as noted in the description of step 3, there is no strong requirement for a fast confirmation time as long as the voter can see that her vote transaction has been broadcast to the network nodes. Confirmations are needed only to etch the transaction in the blockchain forever and to make verifications possible at any time in the future.
If large scale operations like a national election are on the agenda, the organizers should consider mining themselves to ensure that the vote transactions will eventually make their way to the block-

chain: the more hashing power is harnessed by the organizers, the lower the statistical cap on the maximum confirmation time.

4) Counting votes on election day

At the end of election day, the application can simply display the compiled results from its database. Compared to a traditionnal, proprietary system, the added feature is the ability to display a funded multi-signature address for each vote record.
The blockchain does not say much about this address yet until its coins are spent to another address. The election results will be made independently verifiable in the next and final step.

5) Post-election: showing independently verifiable results

Now is the time to show the world a lasting, easily verifiable proof that the vote count was not rigged in any way.

Lets say one of the candidate is Martin: Martin has advertised a special bitcoin vanity address 1Martin.. on social networks during the election campaign.
For each voter, the application uses the 2 private keys corresponding to KeyB and KeyC respectively, to sign a transaction spending the coins from the 2-of-3 multi-signature address to the address 1Martin.., unequivocally linking the vote to the candidate.

The transaction from the multi-signature address to 1Martin.. includes the following input:

SignatureB SignatureC SerializedScript

where SerializedScript designates the redeem script converted from its binary form to an hexadecimal string.

Looking up the multi-signature address in the blockchain using any block explorer, it's easy to find the redeem script in the spending transaction, in other words the serialized input script of the transaction sending coins from 3NSxmh… , revealing the three public keys.
An observer can use the Merkle Tree published by the candidate before the election to check the validity of the interior hashes connecting KeyC to the Merkle Root. Similar verifications applied to KeyA and KeyB will allow the observer to assert the validity of the multi-signature address.

A discrepancy between the vote count provided by the application on election day and the blockchain transactions could be spotted easily with any block explorer software.

However, it is impossible for an observer to link a multi-signature address to a voter insofar as KeyB cannot be linked to the voter identity by the observer.

Incidentally, this system uses an advantage of a digital signature compared to a manual signature: a digital signature identifies its author only with her consent , if and when she reveals her identity.
The votes, while preserving the secrecy of the ballots, are perfectly verifiable by candidates and voters , independently of any organization.

To complete the post-election operations, the organizers collect the funds from address 1Martin.. and from the other candidates vanity addresses that were prepared in the pre-election period.

Would it be simpler to just create a new block chain and wallet for each election cycle?

It's in fact simpler to use an existing blockchain.

Besides, no other blockchain comes close to the Bitcoin blockchain in terms of protection against a 51% attack led by one of the candidates.

Creating a new blockchain dedicated to a particular vote amounts to requesting the voters to trust more the organizers and all the candidates: the organizers or some candidate could take control of the new blockchain, mine some votes, drop some other votes. This would seriously disrupt the verification of the vote count.

Conclusion

Before the invention of Bitcoin , electronic voting solutions were not satisfactory because they were not easily auditable and not sufficiently transparent neither for candidates nor for voters. In addition, they require a costly, labor intensive set up.

With the Bitcoin Blockchain, any community can organize a free, secure electronic voting.

Initially, communities, associations or listed corporations should consider using this technology for local elections, board elections or general assemby voting of their members or shareholders.

Scaling up to national elections might require direct involvement of the organizers in mining operations or some level of cooperation with miners.

The proposed system does not solve all the issues associated with electronic voting but it does provide a valuable alternative to current, proprietary electronic voting systems with the following benefits:

- free, open source peer-reviewed software,
- ubiquitous,
- secure
- protecting the secrecy of the ballots
- allowing free, independent audits of the results
- minimizing the trust level required from the organizers

References

[1] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", November 2008.

[2] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133, April 1980.

[3] Gavin Andresen, BIP-0016 (Bitcoin Improvement Proposal), January 2012.

[4] https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki.

[5] https://en.bitcoin.it/wiki/Vanitygen

[6] https://en.bitcoin.it/wiki/Transaction_fees.