A Secure Distributed Ledger Through Fair Witness Selection

Jetse Brouwer[1]

***Abstract*— the Fair Witness Selection Protocol is a novel witness selection scheme that can be used to create a truly scalable distributed ledger. The proposed scheme offers an upper-bound of 0.001% change for fraudulent while only requiring 20 witnesses (if $< 1/5^{th}$ of the network is malicious) while also guaranteeing liveliness. Because the number of witnesses required remains constant with respect to the network size, it opens the door for truly scalable DLT. It does so by relying on widely available and well understood cryptographic primitives. This paper provides a mathematical proof on the correctness and liveliness of the proposed scheme, and suggests values for the security parameters under different network assumptions.**

I. INTRODUCTION

This document is the result of a broader research on establishing secure distributed ledger technology (DLT) without global consensus. The reasoning for such a distributed ledger has to do with scalabilty: global consensus requires agreement by a majority of all nodes. This not only creates the demand to know about all the transactions, but also requires total order. By optimizing the network and tweaking certain parameters high throughput can be achieved, but the question is how far this tweaking will get us with respect to the size of the network.

This eventually led to the question: "How to achieve a secure distributed ledger without global consensus". This papers proposes Fair Witness Selection Protocol (FWSP) which places a configurable limit on the probability of the success of a fraudulent transaction.

in section II we discuss the Trustchain data structure and how this removes the need for total order, related work, formalization of terminology, and then define a system model. In section III-A the Fair witness selection protocol is described. in section IV-C the correctness, liveliness, and results of the propose protocol are described.

II. DEFINITIONS, FORMALIZATION AND PREVIOUS WORK

A. Trustchain

Trustchain is a distributed pair-wise ledger developed at Delft university of Technology. Trustchain distinguishes itself from traditional DLTs by not having a single ledger containing all transactions, but having a single ledger for every unique user. Every block is created by multiple parties, and the parties involved in the transaction will always be among the parties creating the block. This jointly created block will then be appended to the chain of all involved parties through a hash. Every block irrevocably entangles ledgers of all involved parties.

Every block points to a previous block which means that each block effectively is a record of an happened-before relation as described by Lamport in [1]. Due to these pointers and the transitivity of the happened-before relationship each ledger is self-contained and consistent with respect to the happened before relation. Since every party involved with the transaction is required to co-create the block, no transactions that affect a party that are not recorded on that party's ledger can exist.

In figure 1 an example of the happened-before relation is shown. When thinking of the arrows as beneficial monetary transaction in the direction of the arrow, it can be easily reasoned that causal ordering of transactions are already sufficient for correct execution of these transactions. To execute transaction $j \rightarrow d$ only the order $i \rightarrow j$, $f \rightarrow i$, $e \rightarrow f$, $b \rightarrow e$, and $a \rightarrow b$ is relevant to asses the validity. The exact timing with respects to c and e, f, g, i, and j is irrelevant, knowing that $c \rightarrow d$ and $b \rightarrow c$ is enough to assess the current state of P_1's ledger.

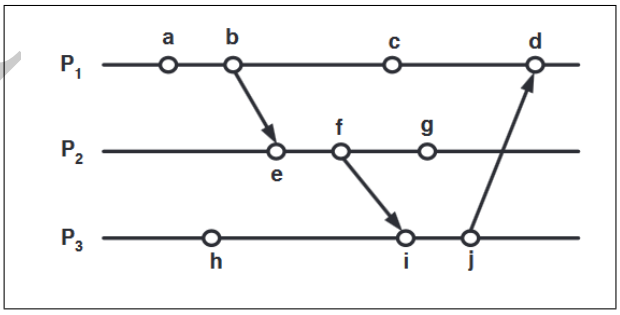

Fig. 1. An example of the happened-before relation. The horizontal lines represent ledgers, and the The events a, c, and h do not involve P1, P2, or P3 as counter-party so the arrows are omitted for clarity.

Since the Trustchain data structure already is inherently causally ordered, and a single ledger already describes the total current state of the corresponding party, the remaining reason for global consensus is to prevent invalid (potentially malicious) transactions from happening. When global consensus can be safely dropped as a requirement, the requirement for a single data structure and a majority vote per transaction can be dropped. As a result multiple parties (that do not depend on, or interact with each other) can be transact concurrently with each other, enabling a highly scalable DLT.

In remaining part of this work we will focus on prevention of malicious transactions through the Fair Witness Selection

[1]J. Brouwer is with Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands

Special thanks to Alexander Stannat, master student in mathematics, for his expertise and guidance regarding the mathematical portions of this document

Protocol.

B. Related work

Significant work has been done to increase the throughput by utilizing and improving existing consensus algorithms. Tendermint[2], HoneyBadger[3], and Ripple[4] use Byzantine fault-tolerant consensus algorithms to create global consensus, a problem with Byzantine fault tolerance (BFT) protocols is that they require a fixed set of servers to be determined ahead of time; Undermining the decentralized nature of distributed ledgers and opening the window for attacks targeting to that specific set of server. Furthermore BFT protocols do not scale very well.

Algorand [5] uses verifiable random functions to select a committee which will be responsible for the creation of the next block. Algorand requires a committee size of 2000 to achieve a probability of 5×10^{-9} that a invalid transaction is passed (under the assumtion that at most 20% of the network is malicious) where as FWSP achieves the same security level with only 27 witnesses under the same assumption.

Another branch of research focuses on incrasing throughput by chainging the underlying data structure. GHOST [7], SPECTRE [8], and Meshcash [6] are recent proposals for increasing Bitcoins throughput moving from a chain-structured ledger towards a tree or directed acyclic graph (DAG) structures, and resolving conflicts in the forks of these data structures.

A structural difference between Trustchain and the above mentioned proposals is the global data structures on which all the algorithms work. This means that only a single transaction, or set of transactions can be dealt with at the time and that they still require global consensus. At the time of writing no pair-wise ledgers and accompanying consensus algorithms were known to the authors.

C. Formalization

We begin the description of the systems by defining the following terms:

- **Node** A node is a single entity running the Trustchain network. Every node within the network must be able to aid in the ratification of transactions (I.e. be a witness). A node is allowed to engage in a transaction with another node and propose this to the network.
- **Transaction** An agreement between two ore more parties, including a publicly verifiable signature from every party and pointers to the last blocks of the transactors unique chain,.
- **Witnesses** A node that is not involved in the transaction, but is chosen to oversee the the transaction and approve it in case of a valid transaction.
- **Block** A block is a data structure containing the transaction, the witness' IDs and the corresponding signatures, finalized by a hash.
- **Chain** A chain is an ordered set of blocks, where each block contains a pointer to a previous block (except for the genesis block, which represents the start of a chain). when talking about the chain of a specific party, the ordered set of all blocks co-created by that party is meant.

We use the term *nonfaulty* to refer to nodes in the network that behave honestly and without error. A *malicious* nodes may deviate from the algorithm (Byzantine errors) randomly send or refuse to send messages.

D. System model

FWSP makes certain assumptions on the system for it to work. We assume that all the nodes have a private and authenticated channel for communication (for example through the use of public key cryptography). A weak sense of synchronization is implied trough to use of messaging time-outs, this is done to overcome the termination impossibility in asynchronous systems as described by Fischer, Lynch, and Paterson[9]. We assume that every node node has contact information, earlier blocks, and the signature verification key for every node (or means to acquire this); be it through a predetermined look-up table, or a byzantine-fault tolerant DHT[10]. As proven in [11] there are no solutions for consensus algorithms with more than $\lceil \frac{n}{3} \rceil - 1$ malicious nodes, therefor we assume this as the upper-bound. It is assumed that the unique identifiers are in the same domain as the H().

It is assumed that:

- nonfaulty nodes only propose valid transactions, will respond timely (i.e. before message time-out) and will sign every valid transaction.
- All malicious nodes collude, only propose invalid transactions, and will only sign transactions from malicious nodes.

Refusing to sign any valid transaction simulates a denial-of-service attack on honest nodes. It assumed that transactions are agreed uppon and signed by the involved parties before initiating this protocol.

III. Fair Witness Selection Protocol

The core concept of the fair witness selection protocol is every block will be witnesses by a number of nodes. Only with enough witness signatures a block will be considered valid.

Now the question arises, how to pick witnesses in a random but verifiable way. Random because no party should be able to manufacture a transaction in such a way he has control over the selected witnesses. Verifiable as anyone in the network should be able to verify that the witnesses were selected according to the algorithm.

The design goals are as following:

- Every valid transaction must be accepted in finite time.
- An invalid transaction will be accepted with a negligible probability.

A. Definition

Let N be the set of all nodes, w the set of selected witnesses, M the set of all malicious nodes, H the set of all honest nodes, and k the minimum required number of

signatures. such that $|N| = |H| + |M|$, $|M| \leq \lceil \frac{|N|}{3} \rceil - 1$, and $k = \lfloor \frac{2|W|}{3} \rfloor + 1$. We further more specify s to be the minimum size for the witnesses set, this number vastly impacts security (see section IV-C for suggested values). Let $V(t) \rightarrow \{true,\ false\}$ be a deterministic function that takes transaction t and outputs $true$ if and only if t is valid. Let $H(\{0,1\}^r) \rightarrow \{0,1\}^n$ be a cryptographically secure, uniformly distributed hash that takes a message of any lengths with an output of length n.

Let $Sign(\{0,1\}^r, sk) \rightarrow \{0,1\}^n$ be a secure signing function, and $Ver(\{0,1\}^n, pk) \rightarrow \{true,\ false\}$ the corresponding verification function.

The core strength of FWSP lies in the witness selection. The i^{th} witness is selected through:

$$w_i = \sup\{z \in N : z \leq H(t||i)\} \tag{1}$$

Due to the uniform distribution of the hash function, node are effectively selected at random. Furthermore, the pre-image Resistance of the hash ensures that its difficult to find a transaction that will result in a given ID. There is a non-zero possibility that the selected node is malicious, however the probability that this and every successive witness is malicious decreases exponentially. A larger minimal witnesses set result in a higher security parameter. In section IV-A a mathematical proof of this security parameter, along with a the minimum witness set for a variety of assumptions and requirements are given.

A trivial attack against this selection process would be to keep increasing i until enough malicious witnesses are found. To combat this also the i used for determining the witness is posted, along with the the ID and a valid signature. Since k, the number of required signatures, grows linearly with the witness the adversary decreases his probability of succeeding exponentially.

An honest node can create a valid block by selecting his witness using equation 1 and request those witnesses to sign the transaction. Again, there is the non-zero possibility that less than k honest nodes are among the witnesses. however, since by definitions $|H|$ is at least twice $|M|$ the probability of an honest majority exponentially increases with every increment in the witness set.

Block creation: The creation of the block starts by verifying the validity of the block. if the block is valid the node calculates the hash for the block with a 0 append, and selects the node who's ID is the closest but smaller or equal to the hash and sends him the chosen ID, transactions and 0. The node repeats this process now with 1, 2 ... up to s, (the minimum number of required witnesses) instead of 0.

Every time a node receives a $\perp$ or a request times-out it extends the witness group by increasing i by one, and selecting the witness according to equation 1 once again. This process continues until the node has acquired enough (more than k) valid signatures.

From here on the transaction is concatenated with the signatures, and than finalized by concatenating the hash of the transaction and signatures to create the final block.

Block signing: On receiving a signature request, the receiving node first verifies that the transaction is valid and he indeed is the required witness. If this is not the case, the node will reply with $\perp$, otherwise the node will append the given i to the transaction and sign it. By appending the i the node prevents any malicious actor to make false claims about the number of nodes selected as witnesses by modifying i after having received the signatures. The node's ID concatenated with signatures and i are then send back to the receiver.

Block verification: Block verification is a non-interactive process. The verification is straight forward, and is started by checking the hash of the block is correct, and the number of valid signatures is equal to or greater than the minimum required. Next the largest witness set is such that $|validSigantures| \geq \lfloor \frac{2|W|}{3} \rfloor + 1$ still holds, is calculated to ensure no i larger than maximum allowed witness set. For each witness it is verified that they were selected according to protocol, and that the signature is valid. If all of these tests pass the block is valid, otherwise the block is considered invalid.

Algorithm 1: createBlock(transaction, s)

```
t ← transaction, validSignatures ← ∅
if V(t) then
    for i ← 0 to s do
        w ← sup{z ∈ N : z ≤ H(t||i)}
        requestSignature(w, t, i)
        W ← W ∪ w
    while |validSignatures| < k do
        if requestSignature() time-out then
            i ← i + 1
            w ← sup{z ∈ N : z ≤ H(t||i)}
            W ← W ∪ w
            send requestSignature(w, t, i)
    block ← t
    foreach s ∈ validSignatures do
        block ← block||s
    return block||H(block)
```

Algorithm 2: on receive requestSignature(t, i)

```
t ← transaction, p ← nodeID
w ← sup{z ∈ N : z ≤ H(t||i)}
if V(t) and w = p then
    s ← Sign(t||p||i, sk)
    reply p||s||i
else
    reply ⊥
```

Algorithm 3: verifySignatures(block, s)

$t, signatures, hash \leftarrow interpret(block)$
if $|signatures| < s \vee hash \neq H(block)$ **then**
  return $\perp$
$w_{max} \leftarrow \lceil |signatures| \times \frac{3}{2} \rceil - 1$
if $\{\exists s \in signatures : i > w_{max}\}$ **then**
  return $\perp$
foreach $e \in signatures$ **do**
  $w, Sign, i \leftarrow interpret(e)$
  $w_{expected} \leftarrow sup\{z \in N : z \leq H(t||i)\}$
  if *not* $(w = W_{expected}$ *and* $Ver(Sign, w_{pk})$ **then**
    return $\perp$

IV. Correctness, Liveliness and Security

A. Correctness

1) Every valid transaction must be accepted in finite time: If an honest node proposes a block containing a valid transaction, but less than $\lfloor \frac{2|W|}{3} \rfloor + 1$ nodes reply with a valid signature (due to malicious behavior or due to faulty behavior) the node will increase the witness set. It will do so until it has collected enough valid signatures or until $W = N$.

Since $|N| = |H| + |M|$ and $|M| \leq \lceil \frac{|N|}{3} \rceil - 1$ it holds that:

$$|H| \geq \lfloor \frac{2|N|}{3} \rfloor + 1$$

A transaction is valid if:

$$v \geq \lfloor \frac{2|W|}{3} \rfloor + 1$$

Where v is the number of valid signatures. By definition honest nodes always sign valid transactions, which gives:

$$v = |H|$$

Therefor a block is valid if

$$\lfloor \frac{2|W|}{3} \rfloor + 1 \geq \lfloor \frac{2|N|}{3} \rfloor + 1$$

When $W = N$ the equation always holds, therefor the block will always be valid. □

2) An invalid transaction will only be accepted with a negligible probability: When randomly selecting a node from the network, the probability of selecting a malicious node is:

$$\Theta = \frac{|M|}{|N|}$$

Let X be a random variable corresponding to the probability of selecting a malicious node from the witness set, which is Bernoulli distributed.

$$X = \begin{cases} 1, & \text{if } n \in M. \\ 0, & \text{if } n \in H. \end{cases} \tag{2}$$

We say that $X \sim Ber(\Theta)$ and that X is a Bernoulli distribution with parameter Θ. If $X_1, ..., X_n$ are i.i.d $Ber(\Theta)$-random variables and $n = |W|$, then $Y = \sum_{i=1}^{n} X_i \sim Bin(n, \Theta)$ models the number of malicious nodes among the witnesses.

k is the number of required valid signatures before a block is considered valid.

$$k = \lfloor \frac{2n}{3} \rfloor + 1$$

For a malicious transaction to succeed, at least k malicious nodes are required in the witness set. Therefore the probability of succeeding is the probability of having k malicious nodes in the witness set, which gives:

$$P(y = k) = \binom{n}{k} \Theta^k (1 - \Theta)^{n-k}$$

Not only k signatures will pass an invalid transaction, but any number greater than k will also succeed. The probability of a malicious transaction succeed for a given witness set becomes:

$$P(y \geq k) = \sum_{i=0}^{n-k} \binom{n}{k+i} \Theta^{k+i} (1 - \Theta)^{n-k+i}$$

Since a node is allowed to increase the witness set if a transaction did not succeed, the probability becomes:

$$\sum_{i=0}^{|N|-n} P(y = k||N| = n + i)$$

Since the network size, and therefore the possible sizes of witness sets increase linearly, but the probability of succeeding starts at a value smaller than 1 and decreases exponentially, it resembles a geometric distribution. Therefor it is assumed that the number of required witnesses for a given security converges to finite number when the network size approaches infinity. Through experiments this convergence is confirmed, the results can be seen in figure 2.

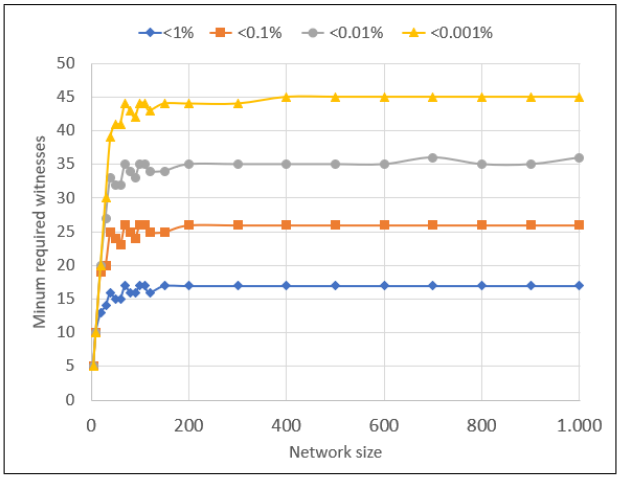

Fig. 2. Minimum number of witnesses as a function of the network size. The malicious nodes make up 1/3 of the total network. The number of required witnesses are given for $< 1\%$, $< 0.1\%$, $< 0.01\%$, and $< 0.001\%$ chance of malicious transactions succeeding.

The oscillation seen in figure 2 is a result of the floor and ceill functions when calculating the number of required signatures.

B. Liveliness

Since every valid transaction eventually will succeed, every party involved in a valid transaction will eventually be able to proceed. Since signing and creating transactions are two separate procedures that have no dependencies on each other, transactions can be signed during a transactions, transactions can be started while signing, and two signature requests can be fulfilled concurrently.

C. Security

The security of the protocol relies heavily on the minimum required number of witnesses. If a single witness would be allowed the probability of a malicious transaction succeeding is equal to Θ. A larger witness group results in a lower probability, the exact number of witnesses depends on the allowable probability of malicious transactions and the number of malicious nodes in the network.

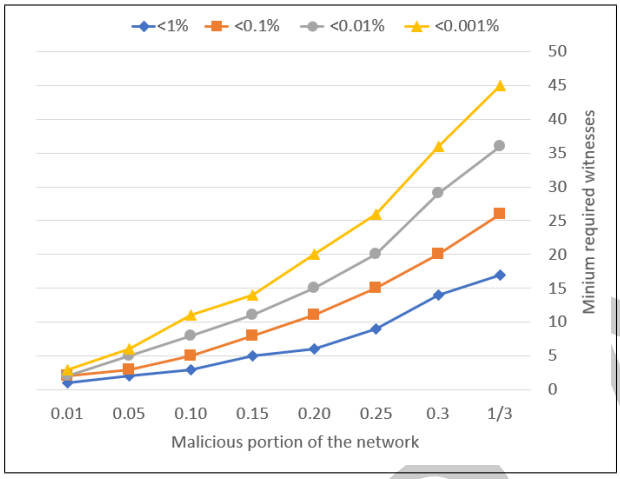

Fig. 3. Minimum number of witnesses as a function of the malicious portion of the total network. The number of required witnesses are given for $< 1\%$, $< 0.1\%$, $< 0.01\%$, and $< 0.001\%$ chance of malicious transactions succeeding.

In figure 3 the minimum number of required witnesses is given, where X-axis is the portion of the malicious portion of the network and the y-axis is the minimum number of required witnesses. In case the required number of witnesses is larger than the network size, the network size should be chosen, as this will result in probability of 0 (See section IV-A for proof).

When assuming a the number of malicious nodes to be $1/5^{th}$ of the complete network (Similar to Algorand [5]) the network only requires 20 witnesses to guarantee $< 0.001\%$ change of malicious transactions. To guarantee a probability of1×10^{-9} or less 27 witnesses are required, whereas Algorand requires 2.000 to achieve the same level of security (for a network of 10.000 nodes).

The security can be further increased when witnesses are not only required to verify the transaction, but also the last n blocks. For a node to be able to execute valid transactions after a malicious one, it has to create at least n other valid-looking blocks before honest witnesses are willing to cooperate with it again. Since every block has an independently selected group of witnesses, the probability of having $n+1$ consecutive malicious blocks is λ^n. If λ is 0.00001 ($< 0.001\%$) and n is chosen to be 15, the probability becomes 0.00001^{16}. If an adversary possesses the power to brute-force such transactions, he might as well use his power to break virtually any encryption used since:

$$0.00001^{16} < \frac{1}{2^{256}}$$

Where $\frac{1}{2^{256}}$ is the probability of guessing a private key for a 256 bits encryption scheme (used in TLS and for financial and government communication).

V. Future work

FWSP can possibly be used as a means to select nodes for block replication. By using the same underlying principle, the same guarantees can be gives while preventing block-hiding and forking attacks.

References

[1] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. Comm. of the ACM,21:558565,1978

[2] E. Buchman, J. K. and Z. Milosevic. The latest gossip on BFT consensus. "https://arxiv.org/abs/1807.04938", September, 2018.

[3] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The Honey Badger of BFT protocols. In Proceedings of the23rd ACM Conference on Computer and Communications Security (CCS), pages 3142, Vienna, Austria, Oct. 2016

[4] D. Schwartz, N. Youngs, A. Britto. The Ripple Protocol Consensus Algorithm. "https://ripple.com/files/ripple_consensus_whitepaper.pdf, 2014.

[5] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich. Algorand: Scaling Byzantine Agreementsfor Cryptocurrencies, MIT CSAIL

[6] I. Bentov, P. Hubek, T. Moran, and A. Nadler. Tortoise and hares consensus: the Meshcash framework for incentive-compatible, scalable cryptocurrencies. Cryptology ePrint Archive, Report 2017/300, Apr. 2017

[7] Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in Bitcoin. In Proceedings of the 2015Financial Cryptography and Data Security Conference, 2015. http://eprint.iacr.org/

[8] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. SPECTRE:A fast and scalable cryptocurrency protocol. Cryptol-ogy ePrint Archive, Report 2016/1159, 2016. http://eprint.iacr.org/

[9] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. Journal of the ACM (JACM) 32.2 (1985): 374-382.

[10] M. Young, A. Kate; I. Goldberg, M. Karsten. Practical Robust Communication in DHTs Tolerating a Byzantine Adversary. Proceedings - International Conference on Distributed Computing Systems January 2010

[11] L. Lamport, R. Shostak, M. Pease. The Byzantine Generals Problem. 1982