Block-Supply Chain: A New Anti-Counterfeiting Supply Chain Using NFC and Blockchain

Naif Alzahrani
Portland State University
nalza2@pdx.edu

Nirupama Bulusu
Portland State University
nbulusu@pdx.edu

ABSTRACT

Current anti-counterfeiting supply chains rely on a centralized authority to combat counterfeit products. This architecture results in issues such as single point processing, storage, and failure. Blockchain technology has emerged to provide a promising solution for such issues. In this paper, we propose the block-supply chain, a new decentralized supply chain that detects counterfeiting attacks using blockchain and Near Field Communication (NFC) technologies. Block-supply chain replaces the centralized supply chain design and utilizes a new proposed consensus protocol that is, unlike existing protocols, fully decentralized and balances between efficiency and security. Our simulations show that the proposed protocol offers remarkable performance with a satisfactory level of security compared to the state of the art consensus protocol Tendermint.

CCS CONCEPTS

• **Computer systems organization** → *Peer-to-peer architectures*; *System on a chip*; • **Networks** → *Peer-to-peer protocols*; • **Applied computing** → *Supply chain management*;

KEYWORDS

Blockchain, supply chain, consensus protocol, counterfeiting

1 INTRODUCTION

The problem of counterfeit products, especially pharmaceutical products, has plagued the international community for decades [11]. The World Health Organization (WHO) estimates that globally, 10 percent of medicines are counterfeit, rising to 30 percent in developing countries [2]. The battle against counterfeiting remains a significant challenge. This is of significant concern, as counterfeit pharmaceutical products can cause critical illnesses and even death. In response to this situation, several product anti-counterfeiting approaches have been proposed. Nevertheless, most of the existing works are centralized and rely on a trusted server to coordinate and manage product authentication.

Most traditional centralized supply chains utilize technologies such as Radio-frequency identification (RFID) and NFC tags and an authentication server to fight counterfeiting attacks. There are three common counterfeiting attacks [10]: (1) the modification of a product's details on a tag, such as changing the expiration date, (2) the cloning of a genuine product's details to use on counterfeit product tags, and (3) the removal of a legitimate tag from a genuine product and its reapplication to a counterfeit product. The typical architecture of centralized supply chains creates several problems. First, there is a tremendous processing burden on the server, since significant numbers of products flood through the supply chain nodes. Second, substantial storage is required to store authentication records for all products. Third, as with centralized systems, traditional supply chains inherently have the problem of a single point of failure. To overcome such issues, blockchain technology stands out as a potential framework to establish a modernized, decentralized, trustworthy, accountable, transparent, and secured supply chain against counterfeiting attacks [11].

Over the last few years, blockchain technology has been an attractive solution for many different industries beyond cryptocurrency [6]. The reasoning behind this is the transparency, security, quality assurance, global peer-to-peer transactions, and decentralization that the blockchain provides [15]. Fundamentally, a blockchain is a public, distributed ledger that contains chained blocks, each of which is made up of several transactions. These blocks are validated globally and transparently to guarantee security (i.e., they are only comprised of valid and correct transactions). The blocks are shared and synchronized across nodes via a peer-to-peer, distributed, and decentralized structure [11]. Despite its potential to elevate security, work and storage distribution, and transparency of supply chains, there have been only a few projects that examine the integration of anti-counterfeiting supply chains and blockchain technology. This has motivated us to exploit this promising technology to combat counterfeit goods by transforming our previous centralized supply chain [2] to a decentralized block-supply chain.

The first problem that we address in this paper is the centralized anti-counterfeiting supply chains. Despite their potential to detect counterfeit products, they still experience single point processing, storage, and failure. Additionally, they do not offer transparency as they do not allow supply chain nodes to verify the authenticity of a product's data. To overcome these issues, we propose the block-supply chain, a decentralized supply chain that exploits blockchain technology. In this chain, each node maintains a blockchain for each product. This blockchain is comprised of chained blocks where each is an authentication event. A new block is proposed to the network by the node that currently has the product (i.e., the proposing node). This newly proposed block is then validated by a number of other nodes we call validators, to ensure that the block is valid. Upon successful validation, all nodes in the block-supply chain network add this block to their copies of the blockchain as will be explained in more detail in Section 3.

CryBlock'18, June 15, 2018, Munich, Germany

ACM ISBN 978-1-4503-5838-5/18/06…$15.00
https://doi.org/10.1145/3211933.3211939

The second problem that we tackle in this paper is regarding the number of blocks' validators and the way they are selected. A blockchain's validators are responsible for assuring its security and executing its consensus protocol. There are a considerable number of existing consensus protocols. Nonetheless, most of them do not take the number of validators or how to select them into consideration, as will be discussed in Section 2. The number of validators in a blockchain network influences its security substantially, especially in a fully decentralized blockchain, in which there are no special nodes; all nodes are trustless as in our block-supply chain use case. The optimal number of validators in such blockchains, which achieves the optimal security, would be all nodes in the network except the proposing node (i.e., $n-1$, where n is the number of nodes). However, this choice results in: (1) substantial validating work as of $O(n)$ for each validation event, (2) high communication overhead to reach consensus with $O(n^2)$ in protocols like PBFT [4] (Practical Byzantine Fault Tolerance) and Tendermint [3], (3) large block size due to including each validator's signature as an evidence of its validity. An alternative choice to the $n-1$ validators is to rely on a fixed static number of validators chosen at the genesis state (e.g., Tendermint and Hyperledger Fabric [5]). This choice, however, leads to: (1) partial centralization, (2) performance bottleneck, especially if the number of validators is relatively low, and (3) unfair selection between nodes, mainly if the nodes have equal voting and computing power.

To overcome the validators' selection problem, we propose a new protocol based on Tendermint, with the ability to select a different set of validators for each validation event (i,e., proposing a new block). Our protocol exploits two stages of randomness to protect against security threats and provide fairness (i.e., equal selection probability among nodes). Additionally, it guarantees the distribution of validation work and decentralization. Our protocol randomly employs *log n* validators, rather than the optimal choice of $n-1$ validators, and achieves a satisfactory level of security when compared to the optimal choice. The reasoning that has led to this specific value (i.e, *log n*) is to reduce the validation work from $O(n)$ to $O(log\ n)$, and the communication overhead from $O(n^2)$ to $O((log\ n).n)$. This reduction results in a considerable enhancement in performance, and it decreases the communication and the storage overhead. Nevertheless, we plan to make the number of validators dynamic and variable based on the hostility of the network.

The contribution of this paper is of twofold:

(1) It introduces a decentralized supply chain that exploits the blockchain technology (*block-supply chain*), and thus, overcoming the centralization issues previously mentioned. Block-supply chain can trace-and-track products without a centralized tracking server. Moreover, it detects the three counterfeiting attacks (modification, cloning, and tag reapplication) by involving the supply chain nodes transparently.
(2) It introduces a *new consensus protocol* that deals with the problem of selecting validators. This protocol has several advantages. First, it is very efficient and scalable, and it shows a considerable improvement in performance when compared to the optimal secure Tendermint with $n-1$ validators. Second, it offers a satisfactory level of security. For example, it was able to detect counterfeiting attacks with a detection rate of 98.4% in a blockchain network with 200 nodes, where 33% of them are malicious. Third, after the geniuses state, our protocol is fully centralized, that is it does not rely on fixed validators. Instead, the validators' set changes every time a new block is proposed. Forth, it provides fairness of selection among nodes with $\frac{1}{n}$ probability.

2 RELATED WORK

In this section, we will examine some very related existing works. The related literature falls into two general camps: (1) works that integrate blockchain and supply chain technologies, (2) the current widely used consensus protocols.

2.1 Block and Supply Chains.

Tian [16] proposed a conceptual framework of an agri-food supply chain using RFID and blockchain. This supply chain is designed to trace agri-food "from farm to fork."

Saveen et al. [1] discussed the potential benefits of using blockchain technology in manufacturing supply chains. Then, they proposed a framework for a manufacturing supply chain for cardboard boxes that involves blockchain as a platform to collect, store and manage product details of each product throughout its life cycle. This work introduced a general overview of replacing the centralized system into a decentralized one using blockchain.

In a study published by Hackius et al. [6], the authors conducted an online survey and asked professionals for their opinion on using blockchain in supply chain management. The authors found that most of the participants were positive about blockchain and the benefits that it can offer.

2.2 Consensus Protocols.

2.2.1 PBFT (Practical Byzantine Fault Tolerance). PBFT [4] is a replication algorithm that can tolerate Byzantine faults. PBFT progresses through a series of views. Each view has a primary node (i.e., proposer,) which is selected in round-robin order. The other nodes (replicas) in the view are called backups. A client sends a request to the primary. The primary multicasts a signed *pre-prepare* message for this request to the backups. The backups accept the *pre-prepare* message, and broadcast signed *prepare* messages. If the backups receive $2f$ *prepare* messages for the request (where f is the is the maximum number of replicas that may be faulty), they multicast signed *commit* messages. Backups execute a view change protocol, in case of a faulty primary. PBFT has the following issues [3]. First, changing the view is subtle and a bit complicated. Second, all previous client requests since the last commit are migrated to the new view.

2.2.2 Tendermint. Tendermint [3, 7] delivers *security* for replicating an application on multiple nodes as it can work even if up to one-third of nodes in the network fail in arbitrary ways. Tendermint is a consensus protocol that does not include proof-of-work mining, which overcomes the energy and resources consumption issues, and speeds up blocks' validations [7]. Tendermint is based on PBFT, and it involves three stages of voting to reach consensus (*propose*, *prevote*, and *precommit*). A proposer *propose*s a new block, then the validators *prevote* on the block and only proceed to *precommit* if they receive more than 2/3 of *prevote*s. Validators only

accept the block if more than 2/3 of *precommit*s are received. Tendermint requires a fixed known set of validators. Voting on a block proceeds in rounds, where each round has a new proposer. The validators vote on whether to commit the block or advance to the next round. Tendermint is notable for its simplicity, performance, and fork-accountability [8]. Though, the number of validators yields a powerful influence on Tendermint's performance. This is due to the communication overhead created by the two stages of voting (i.e., *prevote* and *precommit*). This creates a trade-off between performance and security, where more validators strengthen security. Our protocol is based on Tendermint and inherits all the features offered by Tendermint. However, it deals with the validators' selection issue by selecting a different random set of validators on each block proposal.

2.2.3 Hyperledger Fabric. Hyperledger Fabric employs PBFT as its consensus algorithm [5]. Thus, it can tolerate up to 1/3 byzantine nodes in a blockchain network. In Fabric v0.6, there exist a fixed number of *validation peers* responsible for executing the consensus protocol. A proposer can submit a transaction to any of them. Then, the chosen peer broadcasts this transaction to the other peers. One of the validation peers is selected as a *leader*. When generating a block, the leader broadcasts it to all peers. When a validation peer receives this block, it hashes it, broadcasts the hash to all other peers, and begins counting their responses. If two-thirds responses were received with the same hash, it commits the new block to its local ledger. Hyperledger Fabric, like Tendermint, suffers partial centralization since it employs a fixed known number of validation peers.

2.2.4 Stellar Consensus Protocol (SCP). SCP [12] is a consensus protocol that utilizes *quorums*, where a quorum is a set of nodes from a network sufficient to reach an agreement. SCP is based on Federated Byzantine Agreement (FBA), in which SCP exploits the concept of a *quorum slice*. A quorum slice is the subset of a quorum that can cause one particular node to reach an agreement. The key idea in FBA is that every node chooses its own quorum slices. A node accepts a vote or a transaction when a threshold (e.g., 2/3) of nodes in its quorum slice confirm it. However, SCP requires the quorum slices to overlap. The previously discussed protocols such as PBFT, Hyperledger Fabric, and Tendermint employ a fixed and globally known set of nodes to reach a consensus. In contrast, SCP gives each node a choice to select one or more quorum slices, each of which might have different nodes. Despite the beauty of this design, it might result in undermining the consensus as quorum slices might not overlap.

It is worth mentioning that we only considered the BFT (Byzantine Fault Tolerance) protocols in this section since they are more relevant to our proposed protocol. We did not have the space to include other mining consensus protocols like PoW (prof-of-Work) [13], PoS (prof-of-Stake) [17], or the non-BFT protocols like Paxos [9] and Raft [14].

3 PROPOSED BLOCK-SUPPLY CHAIN

This section describes our block-supply chain in detail. Block-supply authenticates each product and detects counterfeit goods without the need for a centralized authentication server. Instead, it involves the nodes in the authentication process by utilizing blockchain technology. The system has two phases, the initialization phase, and the verification phase. The products' manufacturer executes the initialization phase, and the supply chain nodes execute the verification phase. Each product is occupied by an NFC tag, which contains the product's details such as serial number, name, and expiration date.

3.1 Initialization Phase

This first phase is responsible for initializing the details of each product, securing them, and storing them on the product's NFC tag. Each NFC tag has a read-only unique tag ID (*TID*) and a *counter*. The read-only *counter* is increased automatically on each reading of the tag and keeps track of the number of times that the tag is read by the nodes.

In this phase, the manufacturer forms the product's data (*PData*), which includes the following: the unique product ID (*PID*), the product name (*PName*), the product expiration date (*PExpiryDate*), and a field called (*ToSignTID*), that is equal to the read-only (*TID*). Then, the manufacturer digitally signs *PData* using its private key to produce the product's data digital signature (*SignedPData*). After that, it writes the product data (*PData*) and its signature (*SignedPData*) to the product's tag.

Once the tag is prepared, the manufacturer creates a genesis block for the product. Figure 1 shows the block structure. A block contains three parts. First, the block header, which includes the following: the blockchain ID (to identify each product's blockchain), the block height (order) in the blockchain, the fee to be paid to the block's validators, and the hash of the previous block (H_{i-1}). Second, the Validation of the previous block to provide evidence of its validity. Third, the block data, which contains the product's data (*PData*), the shipping source node address (*Src*), the node the product is being shipped to address (*D*), and the current number of reads on the product's tag (*#Reads*) to track how many times the product's tag has been read (this field is for detecting the tag reapplication attack).

Finally, the manufacturer broadcasts the genesis block to all nodes in the supply chain, and ships the product to the supply chain. The manufacturer is the initiator of the product's blockchain and is no longer involved in authenticating the product.

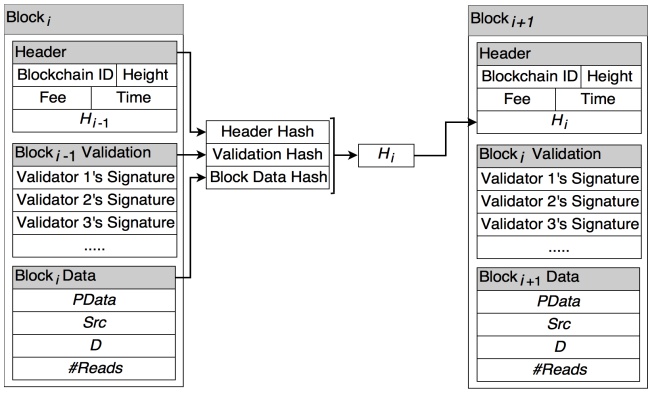

Figure 1: Blocks structure and chaining them.

3.2 Verification Phase

This phase is executed between the supply chain nodes using blockchain. As a product flows throughout the supply chain, its blockchain gets updated each time it leaves a node and moves to the next by adding new blocks to it. When a node (*D*) receives a product, two types of authentication are performed:

Local Authentication: the node (*D*) authenticates the product locally by, first, reading the product's tag and verifying the *Signed-PData*. If this verification succeeds, then the node checks if the *PData* on the tag is the same as the *PData* on the last block in the node's own copy of the product's blockchain. This step detects the "modification attack." Next, the node checks for "cloning attack" by checking the *ToSignTID* on the tag to the tag's read-only *TID*. Then, it compares the *ToSignTID* and *PID* on the tag to the corresponding ones on the last block in the blockchain. If a mismatch is detected, then the product is cloned. The last inspection is to check against the "tag reapplication attack." This is done by comparing the reads on the tag's *counter* (minus one to exclude the current node's read) to the number of reads stored on the last block (*#Reads*). If they are equal, then this indicates that no reads have been performed on the product's tag between *Src* and *D*.

Global Authentication: after successful *local authentication* , and before dispatching the product to the next node in the supply chain, the node becomes a *proposing node*, and proposes a new block. The proposed block contains the new source (*Src*), destination (*D*), the current number of reads of the tag's *counter* (i.e.,*#Reads*), the hash of the previous block, and the digital signatures of the previous block's validators. The proposing node, then, broadcasts this newly proposed block to all nodes in the blockchain network. The validators validate the block globally and vote on it.

Each validator executes the *global authentication* which includes the following steps. First, the validators ensure that the blocks are well-chained and have the appropriate order. Second, they check if *PData* in the previous blocks is the same as the one in the proposed block. Third, they trace-and-track the product by ensuring that the source (*Src*) in a block ($block_i$) is equal to the destination (*D*) in its previous block ($block_{i-1}$). If the three checks are successful, then the block is valid, and it is safe to include it in the blockchain. The validators guarantee that the blockchain always contains valid blocks so the next *proposing node* can safely rely on the blockchain when executing its *local authentication*.

Yet, we have only discussed the role of validators, but how we can select them and how can they communicate to reach an agreement on a block validity? Our proposed consensus protocol answers these two questions in the following section.

3.3 Consensus Protocol

In this section, we propose a new consensus protocol that achieves a remarkable balance between *security* and *scalability*. Unlike other protocols that rely on a fixed static set of validators responsible for validating all proposed blocks, our protocol randomly selects a different set of *log n* validators each time a new block is proposed. Thus, it improves the performance by distributing the validation works among nodes with equal selection probability. Additionally, there are numerous benefits: it provides fairness of selection among nodes, and it yields total centralization due to selecting different sets of validators.

Each node in the block-supply chain has a unique pair of keys (public and private) and is identified by its public key. There are four types of nodes. First, the "proposing" node, which currently has the product. It executes the *local authentication* algorithm, proposes a new block, and broadcasts it to all nodes in the network. Second, the "validator" node, which is responsible for validating the newly proposed block by executing the *global authentication* algorithm. Moreover, validators communicate their votes to reach consensus. Third, the "validation-leader" node, which is responsible for selecting the random *log n* validators for the proposing node. Each proposing node has a corresponding validation-leader assigned to it randomly as will be discussed later. Finally, the "idle" node, which does nothing except waiting for the decision to be made by validators on whether to accept or reject the block. All other nodes in the network are idle.

The validators' selection is performed in two stages:

Validation-leaders' selection: in this stage, each proposing node is mapped *randomly* to a validation-leader at the genesis state. The validation-leaders are regular nodes in the network. The blockchain initiator (i.e., the manufacturer) performs this stage. Then, it sends each validation-leader its corresponding proposing node's ID along with the genesis block. The random mapping guarantees that each proposing node is mapped to exactly one validation-leader and no node is mapped to itself.

Validators' selection: when a validation-leader receives a proposed block from its corresponding proposing node, it executes this second stage. The validation-leader *randomly* selects *log n* validators. Then, it instructs these validators to start validating the block by sending a "*validate*" message. The validators' selection algorithm ensures that the validation-leader a) does not select itself as a validator, and b) does not select a validator twice.

Thus far, we have covered how the validators are selected. However, these validators need a way reach consensus in the presence of Byzantine nodes. Our protocol is based on Tendermint and exploits its capability to overcome up to 1/3 Byzantine faults. The validators in our protocol *pre-vote* on the proposed block, and when they hear from more than 2/3 of *log n* other nodes, they *pre-commit* the block. The block is committed when more than 2/3 of *log n pre-commit*s are received. The consensus algorithm is illustrated in Figure 2, and it is summarized as follows:

(1) The proposing node *propose*s a new block and broadcasts it to all nodes.
(2) The validation-leader node responsible for this proposing node acts upon receiving the proposed block. It executes the "validators' selection" algorithm to randomly select *log n* validators. Then, sends *validate* message to these selected validators.
(3) The remaining nodes wait for a "*validator-time-out*" after receiving the proposed block. The *validator-time-out* is the time that every node waits to hear a *validate* message from the validation-leader node, or a *pre-vote* message from a validator. This time-out period protects the protocol's liveness from faulty validation-leaders. There are three possibilities a node might act in this step. First, if the node receives a

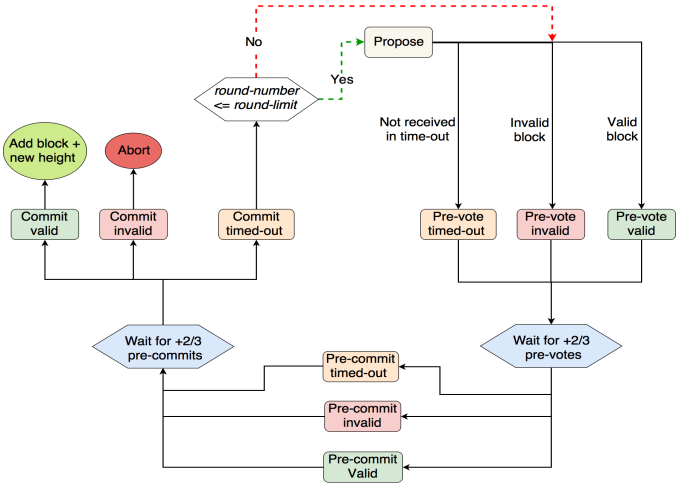

Figure 2: Our consensus protocol. The first step is "Propose".

validate message, then it acts as a validator and carries on the validation process. Second, if the node receives a *pre-vote* message, then it is an idle node, and it waits to hear the remaining 2/3 *pre-votes*. Third, if the node receives neither, then it acts as an "*all-validate*" validator and carries on the validation process. *All-validate* is an alternative mode to the *log n* mode that we described so far. This mode allows all nodes to participate in reaching a consensus. It is a special case that preserves the liveness of the protocol.

(4) The validators wait for a *proposer-time-out* after receiving the *validate* message. This time-out protects the protocol's liveness from faulty proposing nodes. The validators begin this step by initializing the voting's *round-number* to zero. Each validation event (i.e., executing the consensus protocol) is a round. The validators' votes depend on two factors: a) whether or not they receive the proposed block within the *proposer-time-out*, b) whether or not the proposed block is valid. If a validator receives the proposed block in the *proposer-time-out*, it validates the block, and *pre-votes* 'valid' if the block is valid or 'invalid' otherwise. However, if the *proposer-time-out* terminates, then the validator *pre-votes* 'timed-out'.

(5) When a validator receives more than 2/3 *log n pre-votes*, it *pre-commits* 'valid', 'invalid', or 'timed-out' according to the received *pre-votes* type.

(6) When a validator receives more than 2/3 *log n pre-commits*, it *commits* 'valid', 'invalid', or 'timed-out' according to the received *pre-commits* type. The remaining nodes *commit* when they receive more than 2/3 *log n pre-votes* followed by 2/3 *log n pre-commits*.

(7) There is a final subsequent step flows the *commit* step, and it is of three types. First, if a node *commits* 'valid,' then it adds the proposed block to the blockchain and extends it to a new height. Second, if the node *commits* 'invalid,' then it aborts the protocol. Third, if the *commit* is of type 'timed-out,' the idle and validation-leader nodes do nothing, they wait to hear again from the validators. On the other hand, the validators check the *round-number* against a rounds' counter we call *rounds-limit*. If the *round-number* is less than or equal to the *rounds-limit*, the validators: a) increase the *rounds-limit* by one, b) increase the *proposer-time-out* based on the network conditions, and c) start a new round giving a chance for the proposing node to re-propose. However, if the *round-number* is greater than the *rounds-limit*, the validators *pre-vote* 'invalid'.

It is worth mentioning that the proposed block and all types of messages (i.e., *validate*, *pre-vote*, and *pre-commit*) are digitally signed by the sender using its private key and verified by the receiver using the sender's public key.

4 EXPERIMENTS AND EVALUATION

In this section, we evaluate the performance and security of our proposed block-supply chain protocol. One of our most important design goals is to balance between performance and security. We chose Tendermint as a reference protocol due to its noteworthy performance, ability to maintain liveness and safety in the presence of Byzantine nodes, and most importantly its similarity to our blockchain use case (i.e., mining-less). Our protocol achieves remarkable performance and at the same time maintains a reasonable level of robustness in a fully decentralized and distributed manner. The high performance and scalability are accomplished by decreasing the number of validators and distributing the validation work among the blockchain nodes on every block proposal, instead of relying on the same set of validators. The security is achieved by the random leaders-proposers' mapping and validators' selection.

4.1 Performance

We have conducted several experiments to examine our protocol performance with different numbers of nodes. Our reference protocol was Tendermint employing $n-1$ nodes as validators. We chose $n-1$ validators for Tendermint because the only way that Tendermint can guarantee the total centralization, fairness of validators' selection, and optimal security is by involving $n-1$ nodes in the validation process.

We simulated our block-supply chain application to examine and compare the performance of Tendermint and our proposed protocol using Omnet++ as our simulation platform. We chose networks of sizes 100, 125, 150, 175, 200 nodes respectively to investigate the scalability of the protocols when the number of nodes increases. Figure 3 illustrates our findings. When block-supply chain uses Tendermint with $n-1$ validators, the consensus latency (i.e., the time taken to validate products) is very high compared to our protocol. This latency increases dramatically as the number of the nodes increases when Tendermint is used. In contrast, our protocol yields a gradual increase in latency, which demonstrates excellent scalability.

4.2 Security

To evaluate security, we conducted several experiments. For each experiment, we randomly select the 0.33% random malicious nodes for each network of the five networks (i.e., the 100, 125, 150, 175, or 200 nodes networks). The 0.33% threshold is chosen because Tendermint and ,hence, our protocol can only tolerate 1/3 of Byzantine

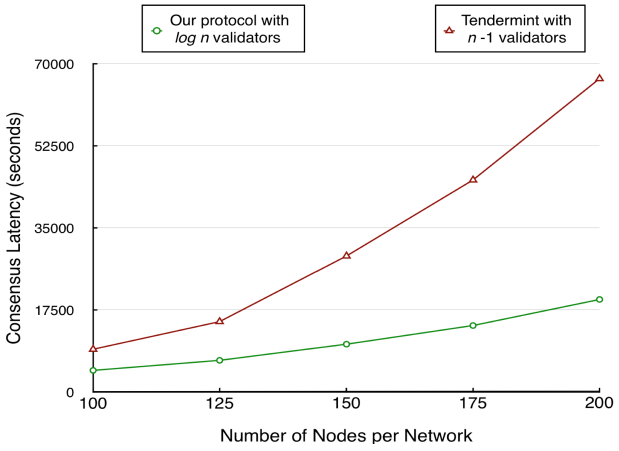

Figure 3: Consensus Latency for our protocol and Tendermint with $n-1$ validators.

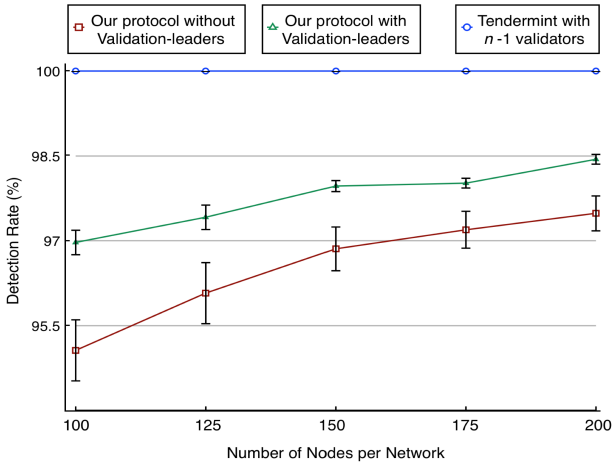

Figure 4: Detection Rate with 95% Confidence Intervals.

nodes. To further increase the risk we assumed, without loss of generality, that these malicious nodes know each other. Therefore, they cooperate with each other to compromise the blockchain. We chose the "cloning attack" as a security threat. Our security metric was Detection Rate (DR). To illustrate the importance of introducing *validation-leaders* in our protocol, we included the detection rate of our protocol with and without them. Figure 4 shows the DR with 95% Confidence Intervals (IC) for our protocol without and with validation-leaders compared to Tendermint with $n-1$ validators.

5 CONCLUSIONS AND FUTURE WORK

We have proposed a new decentralized supply chain (block-supply) utilizing blockchain and NFC technologies. The block-supply chain was able to track-and-trace products and detect modification, cloning, and tag reapplication attacks. For this chain, we introduced a new scalable and secure consensus protocol. Our simulations show that our new protocol is very efficient for large networks, which makes it a suitable choice for large blockchains that require total centralization.

Nerveless, this is an ongoing work, and we plan to make it more efficient and robust. For example, the always-validation (i.e., validators always validate even if chances of attacks are low) is a performance shortcoming particularly in blockchains with low hostility. Future work includes the application of a game theoretical model, so instead of always-validate, validators validate with some probability that is probational to the blockchain hostility.

Additionally, although the validators' set changes dynamically every time a new block is proposed, the number of validators is static (i.e., *log n*), which is a piece of knowledge that can be exploited by an adversary. A future solution is to make the number of validators dynamic and changeable based on a blockchain's hostility factor. The blockchain nodes learn this factor periodically and update their beliefs about the network. As a result, the number of validators utilized by our protocol changes (in a random way) proportionally to the hostility factor.

Finally, one limitation to consider is that it is possible for a validation-leader node to produce an assignment that is not truly random. This may result in a biased validators' selection and may compromise the safety of the network in case if the validation-leader is malicious and cooperates with other malicious nodes. A possible solution is to involve the validation-leaders in the game theoretical model mentioned above so that a punishment payoff is applied in case of such attacks. Another solution is to require the validation-leaders to provide some sort of proof of work.

REFERENCES

[1] Saveen A Abeyratne and Radmehr P Monfared. 2016. Blockchain ready manufacturing supply chain using distributed ledger. (2016).
[2] Naif Alzahrani and Nirupama Bulusu. 2016. Securing Pharmaceutical and High-Value Products against Tag Reapplication Attacks Using NFC Tags. In *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE.
[3] Ethan Buchman. 2016. *Tendermint: Byzantine fault tolerance in the age of blockchains*. Ph.D. Dissertation.
[4] Miguel Castro, Barbara Liskov, and others. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99.
[5] Hyperledger Community. 2018. *hyperledger/fabric*. https://github.com/hyperledger/fabric/tree/v0.6.
[6] Niels Hackius and Moritz Petersen. 2017. Blockchain in logistics and supply chain: trick or treat?. In *Proceedings of the Hamburg International Conference of Logistics (HICL)*. epubli.
[7] Jae Kwon. 2014. Tendermint: Consensus without mining. *Retrieved May* 18 (2014).
[8] Jae Kwon and E Buchman. 2016. Cosmos: A network of distributed ledgers. (2016).
[9] Leslie Lamport and others. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001).
[10] Mikko Lehtonen, Thorsten Staake, and Florian Michahelles. 2008. From identification to authentication–a review of RFID product authentication techniques. In *Networked RFID Systems and Lightweight Cryptography*. Springer.
[11] Tim K Mackey and Gaurvika Nayyar. 2017. A review of existing and emerging digital technologies to combat the global trade in fake medicines. *Expert opinion on drug safety* 16, 5 (2017).
[12] David Mazieres. 2015. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation* (2015).
[13] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[14] Diego Ongaro and John K Ousterhout. 2014. In search of an understandable consensus algorithm.. In *USENIX Annual Technical Conference*.
[15] Marc Pilkington. 2016. 11 Blockchain technology: principles and applications. *Research handbook on digital transformations* (2016).
[16] Feng Tian. 2016. An agri-food supply chain traceability system for China based on RFID & blockchain technology. In *Service Systems and Service Management (ICSSSM), 2016 13th International Conference on*. IEEE.
[17] Pavel Vasin. 2014. Blackcoin's proof-of-stake protocol v2. *URL: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper. pdf* (2014).