Lightweight Blockchain Consensus Protocols for Vehicular Social Networks

Zehui Zheng, Jianping Pan, *Senior Member, IEEE*, and Lin Cai, *Fellow, IEEE*

***Abstract*—This paper presents lightweight blockchain consensus protocols that are specifically designed for vehicular social networks, or other networks that resemble them. They are lightweight in the sense that no puzzles are required to be solved and no cryptocurrency systems are required to be built beforehand. Various scenarios are considered, including whether the communication is reliable and whether the vehicle grouping is static. Although we break our solution of the consensus problem into private chain and public chain, we provide a scheme to smoothly bridge between them. Theoretical analysis is provided to guarantee the algorithms to succeed with high probability, before experiments are conducted to verify the algorithms and compare their theoretical performance versus simulation performance, namely worst-case performance versus average performance.**

***Index Terms*—Blockchain, vehicular social networks, probabilistic analysis.**

NOTATIONS

β	The ratio of the number of bad nodes (adversaries) over the total number of nodes in the group.
b_i	Beacon/message with index i, where i can be timestamp or nonce.
c_{b_i,v_i}	The counter for vote v_i of b_i. If votes are binary (0 or 1), there will be two counters, one for each possible vote.
G	The number of upload trials from non-malicious nodes in the local group.
H	A class of universal hash functions.
k	The number of nodes that are picked to upload consensus to the servers.
n	Number of nodes in the group.
p	Probability that a message is successfully received by any given node.
r	The number of rounds needed.
s	The ratio of deviation from the expectation of a random variable, used in the Chernoff bound formula.
t	Number of rounds needed for sending messages. Same for t_1, t_2, etc.
T	The empirical time for a node to pass by any RSU with high probability.
v_i	Vote with regard to message b_i.
$\|\|$	Concatenation of two strings/variables.
θ	Threshold for making a decision.
δ	Tolerable failure probability.
ϵ	Any arbitrarily small positive number.

Manuscript received September 16, 2019; revised December 5, 2019; accepted January 29, 2020. Date of publication February 14, 2020; date of current version June 18, 2020. This work was supported in part by Huawei. The review of this article was coordinated by Prof. H. Li. *(Corresponding author: Zehui Zheng.)*

Zehui Zheng and Jianping Pan are with the Department of Computer Science, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: zehnzhe@gmail.com; pan@uvic.ca).

Lin Cai is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: cai@ece.uvic.ca).

Digital Object Identifier 10.1109/TVT.2020.2974005

I. INTRODUCTION

BITCOIN blockchain [1] was an ingenious way for distributed systems to reach consensus with high probability. The basic idea is that in each time period (about 10 minutes), it is most likely to have one node who solved a puzzle in the ideal situation, although forks might arise temporarily mainly due to network delay. The puzzle is based on a one-way function and requires the output to have an adjustable length of leading 0's. This process resembles random sampling from the population. The chances of being selected as the next leader are proportional to the computational power possessed. Therefore, as long as the population of good nodes has more than 50% of the total computational power, over the long term, the valid hash chain will outpace invalid chains with high probability.

Bitcoin has also attracted many research works in the area of distributed consensus, which results in a number of variants of the bitcoin blockchain. For example, instead of proof of work (PoW), which burns a considerable amount of electricity, Ethereum [2], [3] proposes a chain-based proof of stake (PoS). With this method, the next leader will be selected according to the size of stake, although this may seem unfair for nodes that are not wealthy or newly joined the network. Also, this may require a reliable cryptocurrency system to be built beforehand.

Among various versions of blockchains, the characteristics that they have in common are:

- by bringing trust to a distributed system, blockchains are able to allow the system to self-maintain and self-adjust;
- more robust and secure, due to the lack of a single point of failure.

It becomes natural to apply the technology of blockchains into solving specific issues arising in distributed systems. To this paper's concern, we are focusing on designing blockchain consensus protocols for vehicular social networks (VSNs). The reason why blockchain fits into VSNs also has to do with VSNs' characteristics. First of all, large amount of data generated by nodes in VSNs make it extremely challenging for any centralized entity to process them, while ensuring that the network is scalable. Secondly, with VSNs getting more and more widely deployed, the ability for the system to handle

security issues will also be an indispensable part for it to grow robustly.

In short, the main problem that we are addressing is how to reach consensus in distributed vehicular social networks. In this work, due to the considerable variants of blockchain, we use "blockchain" as a general term referring to distributed consensus mechanisms, which does not necessarily use a hash-chain structure.

Considering nodes in VSNs mostly have resource constraints, such as storage and computation constraints, our protocols are required to be lightweight. Thus, PoW is not in our consideration, because of its computational intensity. Plus, we aim for good nodes to reach consensus in a faster and more reliable manner, both of which are not satisfactory in PoW. Because with PoW, the generation of the next block takes time much larger than the maximum network broadcast delay, while bad nodes still have chances proportional to their computational power to propose the next block. Similarly with PoS, it would be ideal to reach consensus in a simpler manner, without requiring a secure cryptocurrency system to be already built.

By a simpler manner, we mean that we solely require the nodes in VSNs to exchange messages for reaching consensus, which is the typical way for solving the Byzantine Generals Problem [4]. We also call this type of problems as reaching the Byzantine Agreement (BA). For some nodes in the vehicular social networks, they may exhibit Byzantine behaviours that arbitrarily deviate from valid behaviours. Nodes that are not Byzantine are good nodes. Specifically, in our scenarios, we define **good** nodes as the nodes that are functional and never tend to behave invalidly. Even for some nodes that do not deliberately exhibit invalid behaviours but due to vehicle malfunctions (e.g., sensor failure), they are still considered as Byzantine nodes. However, our target would be for all nodes that do not deliberately exhibit any invalid behaviours to reach consensus, even for those that are Byzantine. Thus, our objective differs from the traditional Byzantine Agreement from this aspect.

Another feature of VSNs that makes our objective easier to reach than Byzantine Agreement is that every good node in the local group has a sensing range that can cover the whole local group and thus observe the same event. With this assumption, we claim that every good node will have the same vote and they vote truthfully. However, each good node still needs to collect votes from other nodes, because the data that lead to the consensus are not kept, due to the storage constraints of most of the nodes. Only relevant group consensuses are kept with nodes and node signatures are as proofs.

We propose our solution as a hybrid chain design. In other words, we integrate private chain with public chain. On the private chain, nodes in the local group exchange votes and reach consensus locally. Local consensus will be uploaded to servers as the public chain. This is done by some nodes when they are passing by roadside units (RSUs).

Since we have introduced the problem that we are working on from a high level perspective, it would be helpful to actually list scenarios that our system may be applied into:

- Suppose that there is an accident happening in the middle of the road, where it is not in the coverage of any RSU. Only the nodes in that local group are able to sense all the duration of the accident and reach consensus of whom to be responsible. Witness nodes do not have to stay at the same place to report to the police, who came shortly after the accident. When passing by the next RSU, some witness nodes will upload the consensus to the RSU, for police or insurance investigations.
- Similarly, when there is congestion happening on the road, nodes in the local group are able to sense and reach consensus about whether the road is congested. Even when some bad nodes fake the message that there is congestion while there is not, they will not be able to collect enough signatures, because good nodes can sense that it is not the case. Good nodes may even reach consensus that some message senders are Byzantine and actions need to be taken by the servers. When passing by the next RSUs, consensuses will be uploaded by some nodes.

The contributions of this paper include:

- We propose blockchain consensus protocols that are extremely lightweight, specifically for reaching distributed consensus in VSNs.
- We break the solution into private chain and public chain, while providing a scheme to bridge between these two parts.
- We propose tailored versions of our protocols for various scenarios. Scenarios include when the communication link is reliable, when the communication link may be unreliable, when the vehicle grouping is static and when the grouping can be dynamic.
- We conduct extensive experiments to evaluate our algorithm's average performance, in comparison with the theoretical guarantees.

The rest of the paper is organized as follows. Section II lists research works that are related to blockchains, Byzantine Agreement and vehicular networks. Section III states the preliminaries that are useful for the design of protocols, including concentration bounds, family of universal hash functions, notations with their meanings, key definitions, and general assumptions and requirements that are applied throughout the entire paper. We illustrate the design of consensus algorithms for the private chain (local group) in Section IV, including the scenarios when the communication link is reliable and unreliable. Following the private chain, Section V focuses on the public chain, including the cases when grouping is static and dynamic. In these two sections, we also illustrate the structures for the private chain and public chain, respectively. Simulation results for both the private chain and public chain are shown in Section VI. We draw the conclusion in Section VII.

II. Related Work

Bitcoin [1] uses proof of work that requires nodes to be constantly solving computational puzzles. This causes a large amount of electricity consumption. Plus, it is slow to reach consensus, because the difficulty of computational puzzles is adjusted to ensure the puzzle solving time is much larger than the network broadcast delay, to avoid forking on the hash chain.

Ethereum [2] enables the blockchain to be Turing-complete and supports smart contracts, which allow nodes to be more autonomously organized. Also, Ethereum tends to apply proof of stake for selecting the next block [3]. However, this might require an existing cryptocurrency system.

There are also variants of the above two well-known blockchains. The idea of PHANTOM, GHOSTDAG [5] is to shorten the interval of block creations, which in bitcoin blockchain will create more forks and separate the computational power of good nodes into working on separate sub-chains. This will render the system toward a more vulnerable position, since the power of good nodes is diverged. Thus, instead of finding the longest chain, PHANTOM recognizes the maximum well-connected k-cluster. However, due to the fact that finding the maximum k-cluster is an NP-hard problem, GHOSTDAG is proposed as a greedy algorithm to solve it. However, a formal proof is still under construction for this paper. Lightning network [6] provides a second-layer solution, where a large number of micro-payment transactions can happen off-chain and only the ending-balance will be updated onto the blockchain. Since this solution is based on the bitcoin blockchain, it also inherits the shortcomings of the bitcoin blockchain. In most of the blockchains, there are two roles of nodes in the network: nodes that initiate transactions and nodes that approve them. In IOTA [7], these two roles are combined as one, aiming to support more micro-transactions by getting rid of miners and blocks. Each transaction needs to approve two transactions selected by the Markov Chain Monte Carlo (MCMC) method. The key feature of Hashgraph [8] is *gossip about gossip*, where no votes need to be sent because every node will eventually know what other nodes have learned about. However, message size could be a concern for this variant. In Snowflake to Avalanche [9], each node does random sampling to learn about other nodes' votes and adjust its own vote. Eventually, *weak termination* is achieved in the sense that all good nodes decide on one value or none of them decides (decides not to decide). But only transactions from bad nodes may not be approved.

Traditional research works for Byzantine Agreement mostly assume that communication links are reliable, which is typically the case for the wired Internet. [10] achieves reliable broadcast, which can tolerate less than one third of the nodes in the network to be Byzantine. Reliable broadcast can be used as a primitive for reaching Byzantine Agreement. A practical state machine replication algorithm is presented in [11]. As for wireless networks, [12] achieves reliable broadcast with the presence of Byzantine nodes, with high probability. However, it is not specific for VSNs, in which case the algorithm can be further simplified.

Most of the existing research works related to blockchain on vehicular networks are directly built on top of existing blockchains, especially Ethereum (smart contract) and bitcoin blockchains. [13] basically adopts the bitcoin blockchain, while replacing transactions as event messages to reach consensus about events happened in the network. CUBE [14] is composed of hybrid blockchains that use public and private blockchains together. Only for those that require a very high level of trust in critically important elements, such as firmware upgrades by automakers, will be on a public blockchain, e.g., Ethereum. [15]–[17] proposed a self-managed and decentralized VANET, based on Ethereum's Smart Contract. A decentralized trust management system [18] for vehicular networks by jointly employing PoW and PoS. Similar as PoW, [19] proposes a system for data credibility assessment. Few research works for blockchain on vehicular networks focus on reaching consensus by solely exchanging messages (beacons). Lightweight protocols would be necessary due to the time, resource, and scale constraints on vehicles, which is what this paper is focusing on.

III. Preliminaries

A. Concentration Bounds

According to the **law of large numbers**, we know that the average of a large number of independent and identically distributed (i.i.d.) random variables will be very close to the expected value. More specifically, we can use Chernoff Bounds to bound the probability that the deviation from expectation exceeds a certain ratio.

In this paper, we apply a form of Chernoff Bounds [20] for independent Poisson trials, the summation of which is denoted as $X = \sum_i X_i$, with expectation $\mu = E[X]$. Thus, for any deviation ratio $0 < s < 1$:

$$\Pr\left[X \leq (1-s)\mu\right] \leq e^{-\mu s^2/2} \tag{1}$$

This is not the tightest form of Chernoff Bounds, but it is simple and one of the most commonly used.

B. Family of Universal Hash Functions

Let H be a class of functions from set A to set B, where $|A| > |B|$, we say H is **universal** if no pairs of distinct keys collide under more than $(1/|B|)$th of the functions [21]. That is, when h is picked uniformly at random from H, then

$$\forall x, y \in A, x \neq y : \Pr\left[h(x) = h(y)\right] \leq 1/|B| \tag{2}$$

From [21], we can also have the construction of such a universal hash function family. Let $A = \{0, 1, \ldots, a-1\}$, $B = \{0, 1, \ldots, b-1\}$ and p be a prime with $p \geq a$. For $m, n \in Z_p$ and $m \neq 0$, we define function

$$f_{m,n}(x) = ((mx + n) \mod p) \mod b \tag{3}$$

Thus, we have a family of universal hash functions $H = \{f_{m,n} | m, n \in Z_p \text{ and } m \neq 0\}$. At the beginning of our system, we may have to predetermine a number (e.g., k) of uniformly random universal hash functions. Therefore, every node will agree on which hash functions that everyone is using. Set A is the input, which can be encodings for some agreed-on messages (consensuses). Set B can be the set of integers from 0 up to the number of nodes in the local group minus one. Assumed that we sort nodes' ID from the smallest to the largest, up to k random nodes can be chosen by applying such k universal hash functions.

C. Key Definitions

In this subsection, we give definitions for different types of nodes. By nodes, we refer to vehicles.

- **Good** nodes: nodes that are functional and never tend to behave invalidly.
- **Bad (Byzantine)** nodes: nodes that exhibit behaviours that arbitrarily deviate from valid behaviours, including nodes that does not deliberately exhibit invalid behaviours but due to vehicle malfunctions (e.g., sensor failure).
- **Non-malicious** nodes: nodes that never tend to behave invalidly deliberately. The set of non-malicious nodes is a **superset** of the set of good nodes.
- **Malicious** nodes: nodes that may behave invalidly on purpose.

D. Assumptions & Key Requirements

The following assumptions are applied throughout the whole paper, although in some sections, further assumptions might be added to clarify the scenarios that we are addressing.

1) Each vehicle's sensing range can cover the whole local group and therefore can observe the same environment.
2) Digital signatures cannot be forged in reasonable time.
3) The number of bad nodes is less than half of the nodes in any group ($\beta < 0.5$).
4) All good nodes will vote faithfully.

We also list some key requirements that are necessary throughout the whole paper. Further requirements may be added specifically for some sections.

1) Every node has the full knowledge of membership in the local group.
2) Message is sent out by broadcast, which also means that a node cannot send different versions of the message, in one broadcast.
3) Every broadcast message is protected by a digital signature.
4) Messages that are broadcast by any given node can only be received in the same order of them being sent out.
5) If a sent message is received, it is received by the end of the same round.
6) Decisions are binary (0 or 1). Even if real-world decisions are not binary, they can be converted into binary ones through a binary decision tree.
7) Each vehicle is equipped with GPS, and thus has time synchronized [22].
8) RSUs are deployed in most of the main road intersections and are connected with servers through wired links or other reliable communication links. This means that our system has the hierarchy of vehicles, RSUs in main intersections, and central servers connected to and controlling these RSUs directly. Further simulation will consider the ratio of RSU deployment.

IV. Algorithms for Private Chain

The objective for this section is that all **non-malicious** nodes in the local group (private chain) should make the same decision. The challenges of achieving that include:

Algorithm 1: Consensus Algorithm (When $p = 1$).

```
Function sendMessage (b_i)
    broadcast(b_i);
Function receiveMessage (b_i)
    % broadcast vote for b_i;
    broadcast(b_i || v_i);
    decision = null;
    c_{b_i,v_i}++;
    % time out period will be the delay upper bound
      for one round;
    while not time out do
        % receive (b_i, v_i) for the first time;
        if receiveEffectiveVote(b_i, v_i) then
            c_{b_i,v_i}++;
            if c_{b_i,v_i} > θn then
                decision = v_i;
                break;
            end
        end
    end
    if decision == null then
        decision = NotToDecide;
    end
```

- some non-malicious nodes may have malfunctioned sensors and may make a different decision unless they learn about the majority decision;
- malicious nodes may try to fool good nodes into doubting the correctness of good nodes' sensor data, by sending arbitrary votes or not sending votes at all.

Two lightweight algorithms (Algorithm 1 and Algorithm 2) are proposed in the following. In the first one, we will solely consider $p = 1$ for all nodes, where p is the probability that a broadcast message is successfully received by any given node. In the second one, we explore what we can do when p is arbitrarily in the range of 0 and 1. Note that for now we assume n, the number of nodes in the group, is fixed.

Formally, we will prove the following theorem:

Theorem 1: For any arbitrary p value in (0, 1], all non-malicious nodes of a group will make the same decision, unless with negligible probability.

Proof: Combining the following two lemmas (Lemma 2 and Lemma 3) in this section is sufficient to prove this theorem. ■

A. $p = 1$

The algorithm for $p = 1$ is shown as Algorithm 1. During the process, each node will keep a record on which nodes have already voted, so that we would not double count the votes. We set the threshold (θ) to be 0.5 so that only one decision can be made by non-malicious nodes. Under these settings, we have the following lemma.

Lemma 2: Given $p = 1$, all non-malicious nodes of a group will make the same decision, after one round of communication.

Proof: To prove only one decision can reach the threshold, we observe that each node can only have one effective vote, it is not possible for votes v_0 and v_1 to be both above the threshold, otherwise the total number of votes is more than the total number

Algorithm 2: Consensus Algorithm (When $p \leq 1$).

```
Function sendMessage (b_i)
    broadcast(b_i);
Function receiveMessage (b_i)
    % broadcast vote for b_i;
    decision = null;
    r = 1;
    c_{b_i,v_i} = 1;
    while r ≤ r_max do
        broadcast(b_i||v_i);
        % time out period will be the delay upper
          bound for one round;
        while not time out and decision == null do
            % receive (b_i, v_i) for the first time;
            if receiveEffectiveVote(b_i, v_i) then
                c_{b_i,v_i}++;
                if c_{b_i,v_i} > θn then
                    decision = v_i;
                    break;
                end
                % if no decision counter can possibly
                  reach threshold counts;
                if n − Σ_{v_i} c_{b_i,v_i} < nθ − c_{b_i,v_i} for all
                  c_{b_i,v_i} then
                    decision = NotToDecide;
                    return;
                end
            end
        end
        r++;
    end
    if decision == null then
        decision = NotToDecide;
    end
```

of nodes, which is a contradiction with the fact that each node can only have one effective vote.

Now we prove that the only decision will be made after one round of execution of the algorithm. Due to the fact that good nodes are majority, good nodes will make the same decision as their votes, while non-malicious nodes that are malfunctioning will soon realize their votes are not consistent with the majority decision and will make the same decision as the majority (good nodes). By taking the union set of the set of good nodes and the set of non-malicious nodes that are malfunctioning, we reach the conclusion that the set of non-malicious nodes will make the same decision. ■

By this algorithm, we reach consensus in local groups, where communication is reliable, with message complexity $O(n^2)$.

B. $p \leq 1$

In this subsection, we will consider a more practical case when p may not be equal to 1, but in the range (0,1]. Because of the unreliability of communication links, we have to relax our objective to be that all non-malicious nodes in the local group should make the same decision, **with high probability**. In other words, we design such an algorithm that bounds the probability that we fail to achieve our original objective by δ, which is defined as tolerable failure probability.

In this scenario, the simplest strategy would be for each node to re-broadcast its own vote for $r_{\max}$ rounds. The value of $r_{\max}$ depends on how bad p is and how small is the tolerable probability of failure (δ) is. Under this strategy, the following lemma will be proved.

Lemma 3: Given $0 < p \leq 1$ and tolerable failure probability δ, after $r_{\max}$ rounds, all non-malicious nodes of a group will make the same decision, unless with negligible probability (δ), where $r_{\max}$ satisfies the inequality $1 - \left(\sum_{i=n\theta}^{n(1-\beta)} \binom{n(1-\beta)}{i} P^i (1-P)^{n(1-\beta)-i}\right)^n < \delta$ and $P = 1 - (1-p)^{r_{\max}}$.

Proof: In this proof, we consider the worst p value in the group, which makes the following analysis as the worst-case analysis.

For each node, it has probability at least p to receive a vote from another given node and probability at most $1-p$ to not receive that vote. The probability for the node to miss all votes from a specific node after r rounds is at most $(1-p)^r$, or $1-(1-p)^r$ to receive at least one vote from that node regardless duplicates. For convenience, we denote P as $P = 1-(1-p)^r$.

In the worst case, βn bad nodes will not vote at all or cast totally arbitrary votes. So in the worst case, the probability that a given node receives enough ($\geq \theta n$) votes from good nodes after r rounds is $\sum_{i=n\theta}^{n(1-\beta)} \binom{n(1-\beta)}{i} P^i (1-P)^{n(1-\beta)-i}$. Ideally, we should then analyze the probability of all non-malicious nodes collect enough good votes. However, due to the fact that it is **indistinguishable** between behaviours of non-malicious but malfunctioning nodes and nodes that deliberately behave badly, we instead analyze the probability for **all** nodes to receive enough good votes. The probability that all n nodes receive enough ($\geq \theta n$) votes from good nodes after r rounds is $\left(\sum_{i=n\theta}^{n(1-\beta)} \binom{n(1-\beta)}{i} P^i (1-P)^{n(1-\beta)-i}\right)^n$.

Therefore, we will determine $r_{\max}$ for $P = 1-(1-p)^{r_{\max}}$ by satisfying the following equation:

$$1 - \left(\sum_{i=n\theta}^{n(1-\beta)} \binom{n(1-\beta)}{i} P^i (1-P)^{n(1-\beta)-i}\right)^n < \delta \quad (4)$$

Thus, we prove the lemma. ■

In the case when $\theta = 0.5$ and $\beta = 0.5 - \epsilon$, for any ϵ that is very small, we can simplify Eq. (4) to be:

$$1 - P^{n^2\theta} < \delta \quad (5)$$

For example, if $\beta = 0.5 - \epsilon$, $p = 0.9$, $n = 100$, and suppose 2^{-20} (roughly 10^{-7}) is a probability of failure that we can tolerate, then we need 10 rounds to guarantee the failure probability is tolerable. If n goes lower or p goes higher, we will need fewer rounds.

Given the fact that the number of good nodes exceeds θn, if a decision can be made, we can guarantee that after $r_{\max}$ rounds, all the non-malicious nodes will make the same decision, with high probability (w.h.p.). In the exception case where no decision can be made, due to time out or the fact that there is no single fact to agree on, nodes may decide *NotToDecide*.

TABLE I
LOCAL CONSENSUS FORMAT

Group ID
Event
Nonce or timestamp
Decision
Signature 1
Signature 2
Signature ...

One example of no decision can be determined is that some non-malicious nodes cannot collect enough votes to make a decision. However, we could not possibly make the threshold lower because then there could easily be multiple decisions.

Also note that it is still possible (with negligible probability) that some non-malicious nodes have made a decision while the other non-malicious nodes have not. However, it is guaranteed that after $r_{\max}$ rounds, with high probability, all good nodes will make the same decision. Note that we also have early stop detections for this algorithm to reduce energy consumptions for vehicles. For example, when a node has learned about the majority decision, all it will do is only broadcast with no need for processing other nodes' votes anymore. Also, in rare cases when there is no single fact to agree on (line 20), nodes may stop early and decide *NotToDecide*.

Our algorithm for $p \leq 1$ is shown as Algorithm 2. With the above probabilistic analysis, we conclude that we achieve our objective, with failure probability bounded by δ.

By this algorithm, we reach consensus in local groups, where communication may be unreliable, with message complexity $O(n^2 r)$ and r satisfies Eq. (4).

C. Private Chain Structure

For each voting event, the consensus will be in the format shown in Table I. *GroupID* can be either assigned by servers or by hashing a string of ordered IDs of group members. The probability of hash collision can be very low when the output space of hash function is large.

If there are multiple voting events, the consensuses can be organized in chronological order according to the time when the event was initialized. Each event can be easily organized in a chain structure with each element on the chain as a consensus in the format of Table I. Different nodes may have different versions of the chain, in the sense that signatures collected are not exactly the same. However, the consensuses of decisions are the same.

V. ALGORITHMS FOR PUBLIC CHAIN

In this section, we discuss how to go from the private chain to the public chain, so that data can be accessed for the whole system, while relaxing the storage constraints on vehicles. Also, this hybrid scheme reduces the computational overhead for servers. Because without reaching consensuses first in the local groups, servers need to process sensor data collected from all nodes, which is not likely to be scalable as the network grows. In this section, we do not require communication links to be reliable, namely $p \leq 1$.

The objective of this section is to ensure that, given a time length reasonable enough for nodes to upload results, if there is no conflict, then the uploaded consensus is valid, while if there is a conflict, the system should be able to recognize the consensus made by the majority.

Challenges for the process of going from private chain to public chain include:

- Communication link may be unreliable while there is only a limited period of time when vehicles pass by RSUs.
- Servers may not know exactly who are in the local group, so that it is possible for Byzantine nodes to collude and directly share private keys for signatures. Thus, Byzantine nodes may be able to use signatures from colluded nodes that are not actually in the same local group. Specifically, Byzantine nodes can try to fool servers by uploading a fake result (e.g., fake group size and/or use others' signatures) before non-malicious nodes upload the actual consensus.

We break our proposed solution into two cases. Firstly, we handle the scenario when vehicle groupings are fixed all the time. Then we relax this constraint and allow all groups to be dynamic. Formally, we will prove the following theorem in this section.

Theorem 4: Assuming that good nodes will not leave the group during the decision making process, given the empirical time T for a node to pass by any RSU with high probability, it takes time up to mT for a valid result from private chain to appear and stay on public chain except with negligible probability, where m satisfies $(1-p)^m < \delta$.

Proof: Lemma 5 and Lemma 6 in the following two subsections suffice to prove this theorem. Note that regardless of group sizes, given the same p and δ, m (in Lemma 5) and G (in Lemma 6) have the same value. However, they denote different meanings and thus have different notations. ■

A. Static Groups

In this subsection, we require an assumption that groupings are fixed from the very beginning. Every node in the network has full information about the groupings of the whole network, namely, which node ID is in which group.

After an agreement is reached on a local chain, the result will be sent in one message, whose format is shown in Table I, to the RSUs and servers. Servers will verify whether this result is valid, by checking whether the number of signatures exceeds the threshold (θ) and also whether all signatures are from the members under the corresponding group ID.

Since it does not require much effort to verify a decision, any node can send the result to servers. E.g., the node with the smallest ID in the group is expected to send the result to servers. If servers did not receive a valid message within a period of time, servers may randomly request another node in that group to send the result. However, this would not happen too many times, because the servers will know that, with high probability, the node with the smallest ID is Byzantine that does not actively upload consensus.

Algorithm for uploading consensus to servers is shown in Algorithm 3. This pseudocode is from the perspective of vehicle

Algorithm 3: Upload Decision to Public Chain (Static Groups).

```
if ownID == smallestID or
   ownID == requestedID then
    toPublicChainStatic (b_i);
end
Function toPublicChainStatic (b_i)
    if decision! = null then
        wait(RSUsInRange);
        if RSUsInRange() then
            broadcast(groupID||b_i||decision||signatures);
        end
    end
```

nodes. With the algorithm, we show the following lemma to be true.

Lemma 5: Given groupings that are static from the beginning of the system, after time interval mT, where m satisfies $(1-p)^m < \delta$, the valid consensus from private chain will appear and stay on public chain, except with negligible probability.

Proof: We first argue that after time interval mT, valid consensus will be uploaded onto public chain. We then explain why Byzantine nodes will not be able to upload invalid results onto public chain.

Because group members' information is available for static groups, if the node with the smallest ID fails to upload consensuses due to the unreliable communication links, edge serves will randomly request another node in the same group to do so. The probability that servers keep failing to receive the results decreases exponentially as time goes on. Specifically, after m trials, the probability the consensus message is still not uploaded successfully is $(1-p)^m$. We specify an m value according to p and δ to satisfy the following equation so that the probability of failure is only negligible:

$$(1-p)^m < \delta \tag{6}$$

Plus, since the groupings are static, Byzantine nodes are not able to use signatures of nodes that are not actually in the same group, assuming digital signatures are not forgeable. If Byzantine nodes upload results by faking groups size or using others' signatures, those results will be rejected immediately by simply checking the membership information. Thus, there will not be results on the public chain that conflict with the actual consensuses.

We therefore prove the lemma. ■

By this algorithm, we upload data from the private chain of static groups to the public chain, with message complexity $O(-\log\delta)$. This is because the number of trials (m) until a successful upload satisfies $(1-p)^m < \delta$.

B. Dynamic/Static Groups

In this subsection, we relax the restriction that groups have to be static. What could go wrong under this case is that a bad node may simply send a fake message to the servers with signatures from bad nodes, who may be from any other groups. To alleviate this situation, we require each node to periodically handshake with RSUs. By doing this, servers are able to evaluate how likely a node belongs to a specific group. Thus, bad nodes would not be able to use signatures belonging to bad nodes that are far away.

To upload the local consensus to the public chain, we require only some nodes in the group to do so. Although the naive solution is to have all the non-malicious nodes trying to upload consensus, it wastes communication resources. Thus, we only randomly select some nodes from the local group to upload consensus, via k universal hash functions. Assume that we have chosen k such functions from the family of universal hash functions uniformly at random, from the beginning. To know which node each hash function specifies, each node orders the members in the same group, according to IDs, and gives them *ranks*. The output space of each hash function is the same as the space of group members' *ranks*. Also, since we assume each member knows the membership information of its group, they will agree on the *ranks* of group members as well. We stress that the key is for non-malicious nodes to learn about how many trials they should attempt.

However, it is possible that bad nodes will fake the group size and collect only signatures from bad nodes to fake a majority decision. To solve this issue, we rely on some non-malicious nodes to upload the proof for the actual consensus, which conflicts with the fake decision. We have to assume that there are not too many bad nodes who might be in any group between any two RSUs. In other words, since bad nodes could not obtain good nodes' signatures to support invalid consensus and number of bad nodes' signatures is outnumbered by the number of good signatures for the actual consensus, the servers will trust the consensus with more signatures, when conflict happens, and punishments might be imposed to nodes who are believed to send out fake messages.

We state our further assumptions as follows:

- The number of bad nodes who might be in any group between any two RSUs is less than half the smallest size of groups between those two RSUs. This is a stronger assumption than $\beta < 0.5$.
- Good nodes in the local group would not leave the group during the decision making process (this assumption can easily be satisfied when group members reach consensus relatively fast enough).

The system also requires that:

- Each node has to periodically handshake with RSUs, which means that each node should try to handshake with the RSUs whenever it passes by them.
- For any integer k up to the possible largest local group size, k universal hash functions are predetermined uniformly at random from the universal hash function family, at the beginning of the system.
- Each node in the group orders all members in the same local group, according to their IDs. After ordering, each node corresponds to a *rank*. The node with the smallest ID gets rank 0, while the highest gets rank as the group size minus one.

Under these settings, as well as assumptions and system requirements, we prove the following lemma.

Lemma 6: Given time interval up to GT, where G satisfies $(1-p)^G < \delta$, the valid consensus from private chain will be uploaded onto public chain and be recognized as the only valid consensus, except with negligible probability. During this time interval, only k trials from the local group are required to upload consensus, where k satisfies $e^{-(1-2\,G/k)^2\,k/4} \leq \delta$.

Proof: To prove this lemma, we first argue that with G trials from non-malicious nodes, the valid consensus will be uploaded successfully, with high probability. Then we show that by requiring k random trials from group members, at least G out of k trials are made from non-malicious nodes. As the last step, we illustrate the reasoning for why only valid consensuses will stay on public chain.

Assume that we have G trials for uploading consensus made from non-malicious nodes. The probability that all those non-malicious trials fail is $(1-p)^G$. We say this probability can be negligible when G satisfies the following equation:

$$(1-p)^G < \delta \tag{7}$$

To give a theoretical guarantee regarding how good the probability that G nodes out of k are non-malicious is, we apply the Chernoff bound as Eq. (1) illustrates. For applying the Chernoff bound, we still need the expectation of a random node being non-malicious. However, it is non-trivial to know the ratio of non-malicious nodes because non-malicious but malfunctioned nodes behave indistinguishably from malicious nodes. But we know that the number of non-malicious nodes is lower bounded by the number of good nodes. Thus, we use the latter for plugging in the Chernoff bound.

By applying the Chernoff bound, we know that we can find such a k that with high probability, we have at least G non-malicious nodes. Here we consider the worst case when the ratio of bad nodes is $1/2-\epsilon$. Also, assume X as the random variable of the number of non-malicious nodes among k uniformly random nodes. We apply the Chernoff bound where $s = 1-2\,G/k$, $k > 2\,G$ and $\mu = E[X] \geq k/2$:

$$\begin{aligned}\Pr[X \leq G] &= \Pr\left[X \leq (1-(1-2\,G/k))k/2\right] \\ &= \Pr\left[X \leq (1-(1-2\,G/k))\,\mu\right] \\ &\leq e^{-(1-2\,G/k)^2\,k/4} \\ &\leq \delta \end{aligned} \tag{8}$$

Now we prove that only the valid consensuses will stay on public chain, and thus the scheme for bridging between private chain and public chain is secure. The reasoning is as follow:

- by choosing k random nodes to upload the consensus onto the public chain, with high probability, at least one non-malicious node will successfully upload. Therefore, if there is no conflict (malicious nodes do not upload fake results), the uploaded one is the valid one;
- if malicious nodes upload a fake result onto the public chain, then with high probability, non-malicious nodes will upload the true consensus to conflict with the fake one, either before or after the fake one was uploaded. Since the true consensus includes signatures that outnumber the signatures for the fake decision, servers will be able to recognize the true consensus.

We thus prove the lemma. ■

Algorithm 4: Upload Decision to Public Chain (Dynamic/Static Groups).

```
Function ownRank (ownID)
    % sort in ascending order;
    sortedList=membershipList.sort();
    return sortedList.index(ownID);
Function toPublicChainDyn (b_i)
    if decision! = null then
        % hashFnArray stores an array of k hash
          functions and hashFamily stores the
          predetermined IDs of hash functions;
        hashFnArray = hashFamily[k];
        hashMatched = 0;
        hashChecked = 0;
        while hashChecked < k do
            if
              hashFnArray[hashChecked](b_i||decision) ==
              ownRank(ownID) then
                hashMatched++;
            end
            hashChecked++;
        end
        uploadTrials = 0;
        while uploadTrials < hashMatched do
            wait(RSUsInRange);
            broadcast(b_i||decision||signatures);
            uploadTrials++;
        end
    end
```

Pseudocode for this algorithm is shown as Algorithm 4. Specifically, the k value can be calculated from the above inequalities (Eq. (7) and (8)).

With large k, $(1-2\,G/k)^2$ is close to 1. Thus, $\Pr[X \leq G] \leq e^{-\Omega(k)}$, which means that with k increasing, the probability that there are not enough good nodes decreases exponentially. For instance, suppose $p = 0.9$ and $\delta = 2^{-20}$, then $G \geq 7$. Therefore, $k \geq 82$.

Note that these k universal hash functions pick a multiset of nodes to upload consensuses. Specifically, when a node is picked multiple times, it uploads its consensuses to servers for the same number of times.

By this algorithm, we upload data from the private chain of dynamic/static groups to the public chain, with message complexity $O(k)$ and failure probability $e^{-\Omega(k)}$.

C. Chain Structure and Properties

Similar as the blockchain and its variants, we also organize data on the public chain into blocks, each of which contains local consensuses happened in a period of time. We specify this period as the upper bound of the time needed for any local consensus to be reached and uploaded onto the public chain. By doing so, we are able to maintain a relative ordering of events in different blocks. In other words, we know that events in the previous block happened before the events in the next block. For example, we

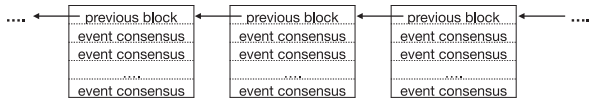


Fig. 1. Public chain structure.

may be able to know that some groups detected that there is a traffic jam happening on a road and later other groups reported a traffic accident happened on the same road. We may be able to conclude that it is the traffic jam that caused the accident rather than the accident caused the traffic jam. However, we may not be able to determine the order for each event.

The structure for public chain is shown as Fig. 1, where each *event consensus* is in the format of Table I. The attribute "previous block" can be the hash value of the previous block, or simply the pointer pointing to the previous block.

We analyze the properties of this chain structure by referring to [23]–[25] for the fundamental properties defined as follows:

- φ-chain consistency: blockchains of any two honest miners at any point in time during the mining execution can differ only in the last φ blocks.
- Chain growth: The blockchain length should be increased steadily.

As our scheme does not have mining, we substitute miners by RSU(s). Since our public chain is mainly maintained by RSU(s), which are connected through a reliable wired network, we claim that we have $\varphi(\tau)$-chain consistency, where $\varphi(\tau)$ is a function with dependence on wired network delay τ. As our public chain is ordered and grows by time, it is straightforward to possess the property of chain growth.

VI. Simulation & Numerical Results

In this section, we show experiments that we have conducted to verify our algorithms. We will show some simulation results and visualization of theoretical bounds, plus the gap between them, for both private chain and public chain, respectively. Note that theoretical results can be interpreted as the worst case scenario, in the sense that the theoretical bound would not be exceeded with high probability, while the simulated results will show the average cases, which can be exceeded roughly with probability one half.

A. Simulation Settings

As for our application scenario, it would be ideal to be on highway roads, where velocities of vehicles are relatively similar and distance between any vehicles is relatively stable. However, we should also consider the cases where the distance between vehicles is less stable, especially on city roads. Thus, we consider that latter case and use the vehicular mobility dataset made by the TAPASCologne project [26], which shows the vehicle trace data in the city of Cologne, Germany. Specifically, we set the coordinates to be $(50°57'06.8''N, 6°51'00.6''E)$ with radius 1km and conclude that it is reasonable to assume for a group of vehicles, 43 good nodes can stay in the group for 13 seconds. These coordinates are set to be roughly the middle of two main road intersections, where it is likely to be outside the coverage

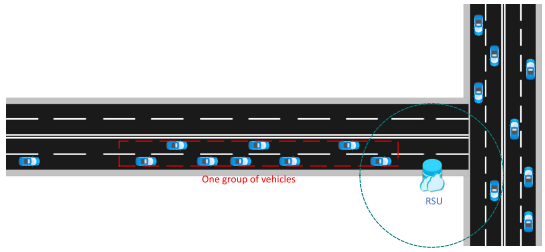


Fig. 2. A simulation scenario.

of RSUs and it is likely for a group of vehicles to remain in the group for a while. However, vehicles in the group may still head onto different roads after the consensus process. As mentioned, we assumed that RSUs are deployed in most of the main road intersections, so that even if vehicles are heading onto different roads, they all have chances to pass by an RSU within time T with high probability. We will further conduct simulations on how the deployment of RSUs may affect the value of T.

In our simulations, the size of local groups can vary from 10 up to 100, and the minimum probability (p) of successful transmission can be in the range $[0.3, 0.99]$. Although empirically a vehicle group size is usually small to avoid high rate of collisions, when multiple RSUs are nearby, a group size can be larger by utilizing RSUs for relaying to more remote vehicles without jeopardizing the p value too much. Specifically, with RSUs, we have more than just vehicle-to-vehicle (V2V) communication, but also vehicle-to-infrastructure (V2I) and infrastructure-to-vehicle (I2V) communication.

As for communications among vehicles and RSUs, we adopt dedicated short-range communications (DSRC) standards in the United States [27]. As defined in IEEE 1609.4, time is segmented into "sync periods," each of which is 100 ms by default. In other words, regardless of group size, 100 ms will be sufficient for each node to broadcast once. However, with larger group size, p will decrease accordingly. In this paper, we do not define the relationship between group size and p, since the relationship is highly related with MAC protocol design (e.g., resource reservation mechanism, access mechanism, modulation mechanism, and so on), for the sake of wide applicability of our work. After broadcast, if the receiver can receive that broadcast, it will be received within 100 ms after the broadcast. Because the local group range is so small that the propagation delay can be negligible. Thus, we conclude that one round is 100 ms.

Fig. 2 illustrates a scenario of our settings. To display the scenario clearly, we only show one example of vehicle groups. Other vehicles will be grouped according to the road segments they are on, heading direction and velocity.

Lastly, we have simulation regarding the deployment of RSUs, on a ten-by-ten grid (Fig. 3), where we assume vehicles will not go out of the grid. On each intersection, each vehicle turns left, right, forward or backward, uniformly at random. We further assume each road segment has the same length and we are interested on how many road segments a vehicle needs to travel before encountering an RSU. We simulate 1000 vehicles on this ten-by-ten grid. The locations of RSUs and vehicles are random in the simulation. An example of RSUs deployment

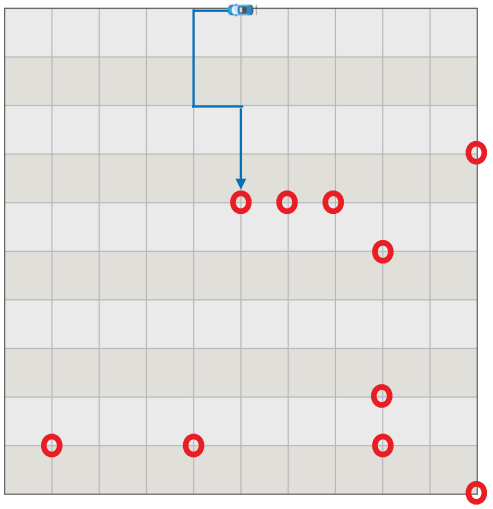


Fig. 3. A ten-by-ten grid.

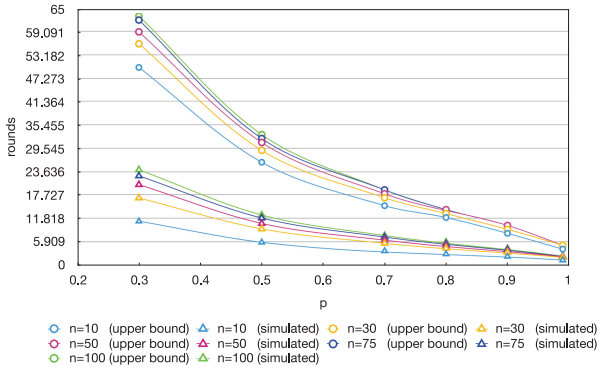


Fig. 4. Number of rounds needed to reach local consensus.

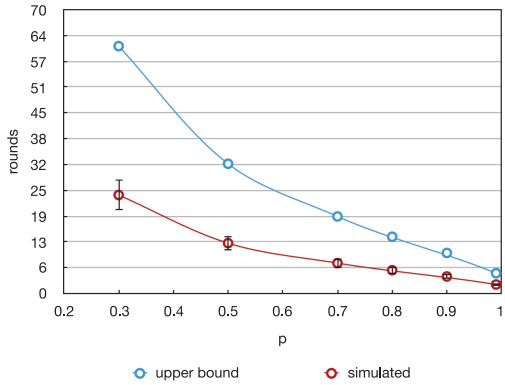


Fig. 5. Standard deviation of the number of rounds to reach local consensus ($n = 100$).

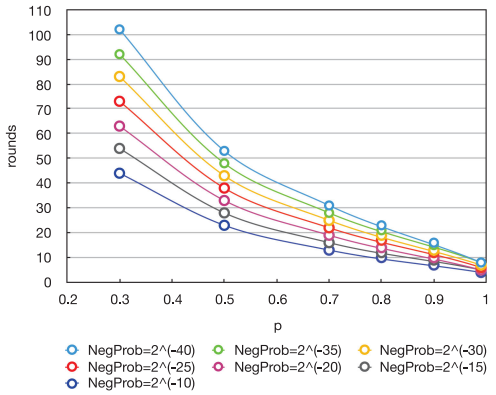


Fig. 6. How the tolerable failure probability changes the theoretically required number of rounds to reach local consensus ($n = 100$).

(10 intersections) is shown as red circles in Fig. 3, in which an example of the trajectory of a vehicle is shown in blue lines. For simplicity, we assume each road segment is 5 km and vehicles travel at 50 km/h, which means that the typical time it takes to travel one road segment is 0.1 h.

B. Simulation for Private Chain

Simulation is done by using **Python** implementing Algorithm 2. We repeat each scenario for 500 repetitions and the simulation results show the average rounds needed for all good nodes to make a decision. We take 2^{-20} as tolerable failure probability, which can be adjusted according to actual application scenarios. Unless β is a variable in the simulation, we take $\beta = 0.5 - \epsilon$.

1) How p and n Affect the Number of Rounds to Reach Local Consensus: In Fig. 4, x-axis is the probability p and y-axis is the rounds we need either theoretically or experimentally. We can see that as p increases, the number of rounds needed decreases, while the more number of nodes in the group, the more rounds needed. Although it is up to the physical implementation regarding what p value is achievable given a certain group size n, we confirmed that all parameter combinations in Fig. 4 can be achieved, by simulating a MAC protocol with linear network coding on vehicle groups with sizes up to 100. We set group diameter to be 300m, where vehicle density is 0.2 per meter. Note that we allow multiple lanes in the simulation scenario. Specifically, we find 6 lanes is a reasonable number in our settings. Under this scenario, p value can be above 0.99.

We may also notice that there is a gap between theoretical value and average value. In practice, the number of rounds needed will not exceed theoretical guarantee with high probability. Note that the gap will be smaller if the system tolerates a higher probability of failure.

Take $p = 0.7$ as an example, in this case, it takes no more than about 20 rounds to reach consensus locally, with high probability. Note that since each round is 100 ms, it takes about 2 s to reach consensus locally. This may also justifies our assumption that *good nodes in the local group would not leave the group during the decision making process*, since it is reasonable to believe good nodes will not get too far away within two seconds.

In Fig. 5, we pick a scenario, where $n = 100$ and show the standard deviation (as vertical lines) of those experimental data. The result shows that the standard deviation is in the range of $14\% - 19\%$. The upper bound will not likely be reached, although also depending on the negligible probability.

2) How the Tolerable Failure Probability Affects the Number of Rounds to Reach Local Consensus: In Fig. 6, we can see that

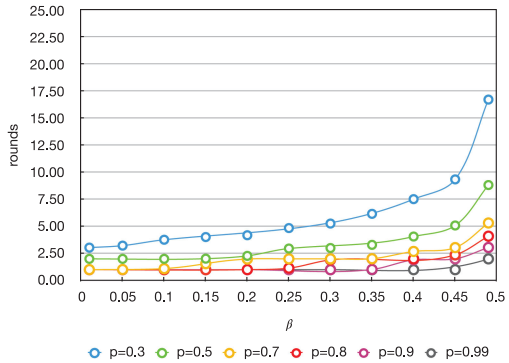


Fig. 7. How the ratio of bad nodes changes the average number of rounds to reach local consensus ($n = 100$).

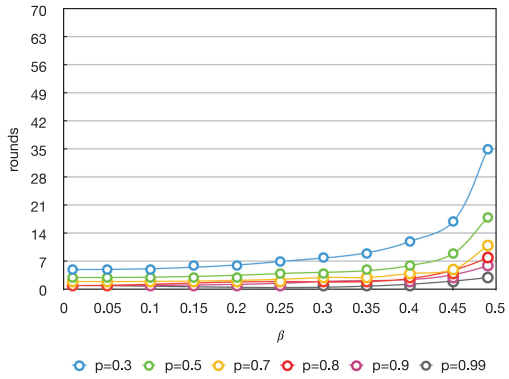


Fig. 8. How the ratio of bad nodes changes the theoretically required number of rounds to reach local consensus ($n = 100$).

if we can tolerate a larger value of failure probability, then the number of rounds needed can be less.

3) How the Ratio of Bad Nodes Affects the Number of Rounds to Reach Local Consensus: Fig. 7 shows that experimentally when β increases, we will need more rounds for all good nodes to reach local consensus.

Fig. 8 indicates that theoretically when β increases, we will need more rounds for all good nodes to reach local consensus, which is consistent with experimental results. In the next figure (Fig. 9), it indicates that all theoretical upper bounds are not violated.

Since in Fig. 9, lines are overlapping with each other when β is small, we take log (base 2) on the y-axis and thus we have Fig. 10.

C. Simulation for Public Chain

Simulation for public chain comprises of mainly two parts. First of all, we evaluate the relationship between the number of times (G) needed for non-malicious nodes to upload consensus and p, under various tolerable failure probability δ. Then, we conduct experiments to verify that among k trials, at least G trials are from the non-malicious nodes. In Eq. (3), we assume that message x has nonce. Thus, the hash result will also be random. In the second part of the experiment, we will simulate $f_{m,n}(x)$ by using uniform random number generator in **Python** and then

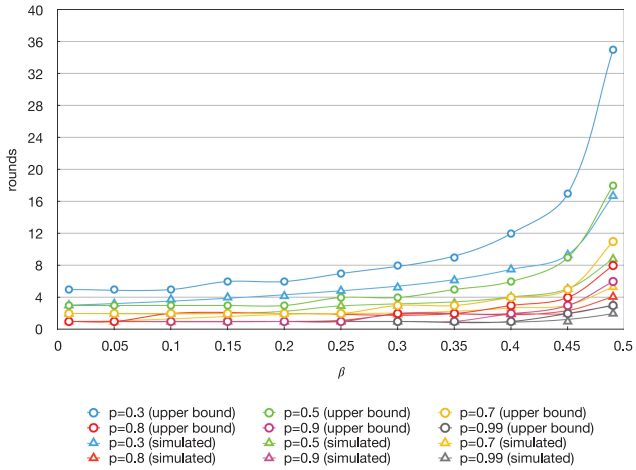


Fig. 9. Average versus theoretical guarantee ($n = 100$).

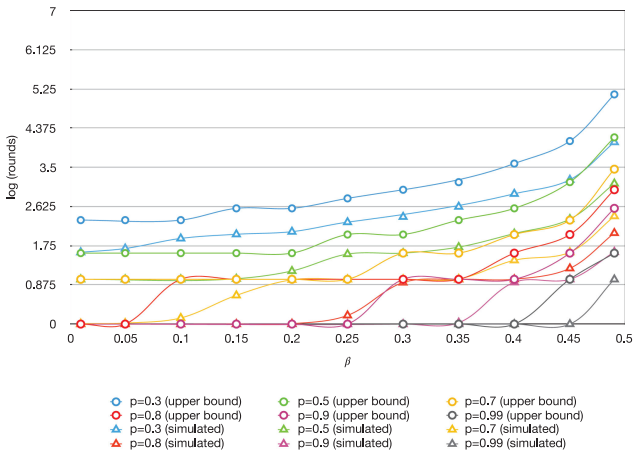


Fig. 10. Average versus theoretical guarantee (in log scale and $n = 100$).

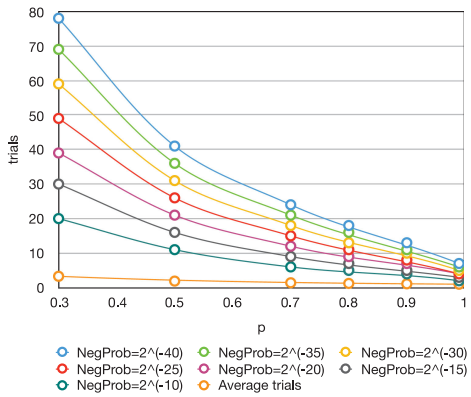


Fig. 11. Number of trials needed to upload to public chain.

we take that number modulo the number of nodes in the group. All experiments will be repeated for 500 times, before taking the average values for visualization, while theoretical upper bounds are directly taken from the equation calculations.

From Fig. 11, we know that as the probability p increases, the number of non-malicious trials needed to upload consensuses decreases. Also, as we relax the tolerable failure probability, the number of required non-malicious trials decreases. Moreover,

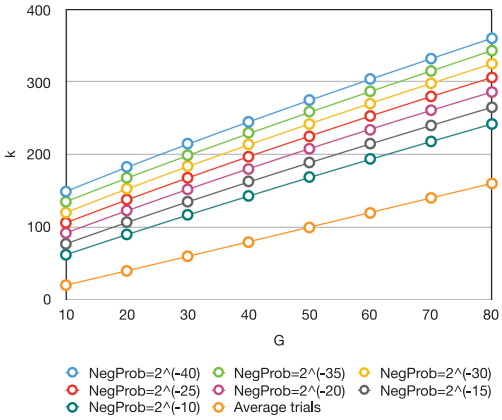


Fig. 12. Number of trials needed to ensure enough good trials.

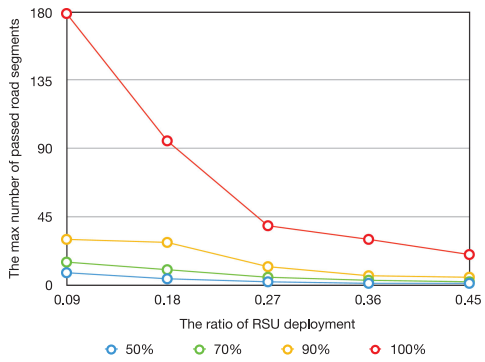


Fig. 13. The maximum number of road segments passed before encountering an RSU.

note that the average performance (shown as the line at the bottom of the chart) indicates that it is very promising, in comparison with the theoretical consideration. In other words, in practice, it takes much fewer trials, than the theoretical guarantees with regard to the tolerable failure probabilities we defined.

Similarly, we evaluate how large k should be to ensure that there are at least G non-malicious trials out of k trials. Here, we only consider the worst case when the ratio of bad nodes is $0.5 - \epsilon$. Evaluation results are shown in Fig. 12. As we require more non-malicious trials (G), the number of required trials (k) should also increase. Still, in average cases, it takes much less trials to collect enough non-malicious trials, in comparison with the theoretical requirements with regard to the negligible probability we defined.

The deployment of RSUs plays a significant role in the process of uploading data from private chain to public chain, which directly affects the value of T. We conduct simulations on how the deployment of RSUs will affect the value of T, by converting the value of T to equivalently the number of road segments a vehicle needs to pass before encountering an RSU, assuming road segments have the same length. We plot in Fig. 13 the maximum number of road segments needs to be passed for half of the vehicles (i.e., half a thousand), 70% of the vehicles, 90% of the vehicles, and all the vehicles. It is clear that the increasing ratio of RSU deployment will significantly decrease the value of T. Specifically, when the ratio of RSU deployment is 0.36, then 90% of the vehicles will pass by an RSU within 6 road segments. From our setting, we know that each road segment takes 0.1 h. Thus, equivalently, we say 90% of the vehicles will pass by an RSU within 0.6 h, which is reasonable for the global consensus in VSNs

VII. Conclusion

In this paper, we have provided a highly lightweight solution for blockchain on vehicular social networks. We broke the problem into private chain and public chain. On private chain, we considered two cases: when communication is reliable and when it may be unreliable. Following private chain, a scheme was designed for transferring data onto public chain, depending on whether the groupings are static or dynamic. We also visualized our experiment results from various aspects and showed that on average, the performance is quite promising, in comparison with the theoretical guarantees derived with regard to given negligible failure probabilities. In the future work, we would like to bring down the message complexity, while keeping the simplicity of current protocols.

Acknowledgment

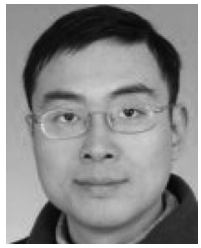
The authors would like to thank Dr. Yue Li and Hamed Mosavat-Jahromi for useful discussions regarding communication issues in vehicular networks. We also would like to thank Peiyuan Guan for processing the dataset made by the TAPAS-Cologne project. We further thank Rui Liu for helping with data visualization. We appreciate the helpful feedbacks provided by Lei Lu and others.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.
[3] V. Buterin and V. Griffith, "Casper the friendly finality gadget," 2017, *arXiv:1710.09437*.
[4] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.
[5] Y. Sompolinsky and A. Zohar, "PHANTOM, GHOSTDAG: Two scalable blockDAG protocols," IACR, Lyon, France, Tech. Rep. 1004, 2018.
[6] J. Poon and T. Dryja, "The Bitcoin lightning network: Scalable off-chain instant payments," 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf
[7] S. Popov, "The tangle," 2016. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf
[8] L. Baird, "The Swirlds hashgraph consensus algorithm: Fair, fast, Byzantine fault tolerance," Swirlds Tech. Rep. SWIRLDS-TR-2016-01, 2016.
[9] Team Rocket, "Snowflake to Avalanche: A novel metastable consensus protocol family for cryptocurrencies," 2018. [Online]. Available: https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV
[10] G. Bracha, "Asynchronous Byzantine agreement protocols," *Inf. Comput.*, vol. 75, no. 2, pp. 130–143, 1987.
[11] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Des. Implementation*, 1999, vol. 99, pp. 173–186.
[12] V. Bhandari and N. H. Vaidya, "Reliable local broadcast in a wireless network prone to Byzantine failures," in *Proc. DIALM-POMC Joint Workshop Foundations Mobile Comput.*, 2007.
[13] R. Shrestha, R. Bajracharya, A. P. Shrestha, and S. Y. Nam, "A new-type of blockchain for secure message exchange in VANET," *Digital Commun. Netw.*, vol. 2019, pp. 1–14, Apr. 2019.

[14] Autonomous car network security platform based on blockchain. [Online]. Available: https://cryptorating.eu/whitepapers/CUBE/CUBEWhite_Paper-V1.3.pdf, Accessed on: Aug. 22, 2019.
[15] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-managed and blockchain-based vehicular ad-hoc networks," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.: Adjunct*, 2016, pp. 137–140.
[16] A. Bochem, B. Leiding, and D. Hogrefe, "Unchained identities: Putting a price on sybil nodes in mobile ad hoc networks," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2018, pp. 358–374.
[17] B. Leiding and W. V. Vorobev, "Enabling the vehicle economy using a blockchain-based value transaction layer protocol for vehicular ad-hoc networks," 2018. [Online]. Available: https://uploads-ssl.webflow.com/5a4ea18a81f55a00010bdf45/5af66af095f858b1e0ed3fa8_Chorus-Technology-Research-Paper-1.0.pdf
[18] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1495–1505, Apr. 2019.
[19] Z. Yang, K. Zheng, K. Yang, and V. C. M. Leung, "A blockchain-based reputation system for data credibility assessment in vehicular networks," in *Proc. IEEE 28th Annu. Int. Symp. Personal, Indoor, Mobile Radio Commun.*, 2017, pp. 1–5.
[20] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[21] J. L. Carter and M. N. Wegman, "Universal classes of hash functions," *J. Comput. Syst. Sci.*, vol. 18, no. 2, pp. 143–154, 1979.
[22] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.
[23] Y. Zhang, C. Xu, X. Lin, and X. S. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2908400.
[24] Y. Zhang, C. Xu, J. Ni, H. Li, and X. S. Shen, "Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2923222.
[25] Y. Zhang, C. Xu, N. Cheng, H. Li, H. Yang, and X. S. Shen, "Chronos+: An accurate blockchain-based time-stamping scheme for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 216–229, Mar.–Apr. 2020.
[26] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Trans. Mobile Comput.*, vol. 13, no. 5, pp. 1061–1075, May 2014.
[27] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

Zehui Zheng received the B.E. degree in electronic information engineering and computer science from Shenzhen University, Shenzhen, China, in 2017. He is currently working toward the master student of computer science with University of Victoria, Victoria, BC, Canada. His research interests include distributed consensus algorithms, specifically blockchain algorithms, and protocols.



Jianping Pan (Senior Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from Southeast University, Nanjing, China, and he did the Postdoctoral Research with the University of Waterloo, Waterloo, ON, Canada. He is currently a Professor of computer science with the University of Victoria, Victoria, BC, Canada. He also worked with Fujitsu Labs and NTT Labs. His area of specialization is computer networks and distributed systems, and his current research interests include protocols for advanced networking, performance analysis of networked systems, and applied network security. He received the IEICE Best Paper Award in 2009, the Telecommunications Advancement Foundation's Telesys Award in 2010, the WCSP 2011 Best Paper Award, the IEEE Globecom 2011 Best Paper Award, the JSPS Invitation Fellowship in 2012, the IEEE ICC 2013 Best Paper Award, and the NSERC DAS Award in 2016, and has been serving on the Technical Program Committees of major computer communications and networking conferences including IEEE INFOCOM, ICC, Globecom, WCNC, and CCNC. He was the Ad-hoc and Sensor Networking Symposium Co-Chair of IEEE Globecom 2012 and an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He is a senior member of the ACM.

Lin Cai (Fellow, IEEE) received the M.A.Sc. and Ph.D. degrees (awarded Outstanding Achievement in Graduate Studies) in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2002 and 2005, respectively. Since 2005, she has been with the Department of Electrical and Computer Engineering, University of Victoria, and she is currently a Professor. Her research interests include communications and networking, with a focus on network protocol and architecture design supporting multimedia traffic and the Internet of Things. She was the recipient of the NSERC E.W.R. Steacie Memorial Fellowships in 2019, the NSERC Discovery Accelerator Supplement (DAS) Grants in 2010 and 2015, respectively, and the best paper awards of IEEE ICC 2008 and IEEE WCNC 2011. She has founded and Chaired IEEE Victoria Section Vehicular Technology and Communications Joint Societies Chapter. She has been elected to serve the IEEE Vehicular Technology Society Board of Governors, 2019–2021. She has served as an Area Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, a member of the Steering Committee of the IEEE TRANSACTIONS ON BIG DATA and IEEE TRANSACTIONS ON CLOUD COMPUTING, an Associate Editor of the IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON COMMUNICATIONS, *EURASIP Journal on Wireless Communications and Networking*, *International Journal of Sensor Networks*, and*Journal of Communications and Networks*, and as the Distinguished Lecturer of the IEEE VTS Society. She has served as a TPC Co-Chair for IEEE VTC2020-Fall, and a TPC Symposium Co-Chair for IEEE Globecom'10 and Globecom'13. She is a registered professional engineer of British Columbia, Canada.