electronics

Article

autoCoin: Secure Content Sharing Based on Blockchain for Vehicular Cloud

Wooseong Kim * and Kyungho Ryu

Department of Computer Engineering, Gachon University, Seongnam 13120, Korea; rudgh1368@gachon.ac.kr
* Correspondence: wooseong@gachon.ac.kr

Abstract: A future smart car will be more than a means of transportation, as it will not only move people to a destination without requiring them to drive but will enable people to work or enjoy a trip with entertainment. For this, smart vehicles need to deal with various types of data for safety and infotainment, such as real-time traffic, multi-media contents, documents and weather information. Recently, a fleet of vehicles connected to other vehicles and infrastructure (i.e., road side units) using a legacy or 5G mmWave spectrum has been considered as a platform to cooperate for those new tasks, known as the vehicular cloud or fog. Within the vehicular cloud, data management should consider security, high availability and interoperability between vehicles. However, these are not easily achievable without a centralized service provider; it is difficult for an autonomous P2P system to guarantee data integrity, and it cannot compensate drivers that actively participate in the vehicular cloud. Fortunately, the many successes achieved in the field of crypto-currency raise the possibility of defining incentives that are necessary for a sustainable digital economy. In this paper, we propose *autoCoin*—an approach that aims to encourage smart vehicles to cooperate to create and exchange infotainment data securely under the assumption of rationality. We introduce a scalable blockchain architecture for *autoCoin* and a smart contract to exchange contents without third-parties using an off-chain technique.

Keywords: blockchain; crypto-currency; intelligent transportation; vehicular cloud; vehicular fog

Citation: Kim, W.; Ryu, K. autoCoin: Secure Content Sharing Based on Blockchain for Vehicular Cloud. *Electronics* **2021**, *10*, 1477. https://doi.org/10.3390/electronics10121477

Academic Editor: Andres Marin

Received: 13 May 2021
Accepted: 11 June 2021
Published: 19 June 2021

1. Introduction

A vehicular cloud is spontaneously established on roads, as shown in Figure 1. The vehicular cloud consists of vehicles and roadside units (RSUs), which are inter-connected by vehicle-to-infrastructure (V2I) connections with the RSU (e.g., LTE or 5G base stations) or vehicle-to-vehicle (V2V) connections using dedicated short range communication (DSRC), Wi-Fi or mmWave-based device-to-device (D2D) connections in licensed bands [1]. V2V communication through the D2D achieves low latency because of its proximity, while it can achieve more reliability and availability through the cellular infrastructure.

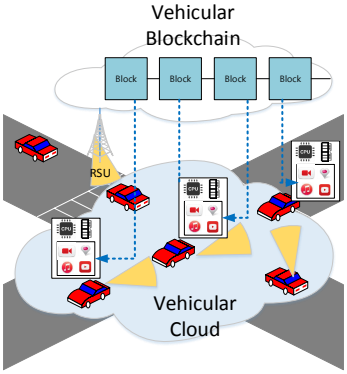


Figure 1. Vehicular cloud and blockchain.

In the vehicular cloud, vehicles collect safety and infotainment data from the environment with a few thousand sensors and cameras and share them in publicly accessible storage units. However, security threats exist in this autonomous P2P system model. For example, a malicious vehicle may attempt to fabricate traffic information to avoid congestion, or a driver who has violated traffic signals may attempt to delete the video captured by other vehicles. Therefore, introducing a centralized service model can mitigate the security threats, but the increased communication costs are not trivial.

In such a trust-less environment, a blockchain system, which is now widely used for cryptocurrencies such as Bitcoin [2] and Ethereum [3], provides many useful traits such as non-repudiation, non-alterable storage and incentives defined in a pure digital economy. Considering there is no trust governor in charge of the security of the vehicle data, a decentralized blockchain model is advantageous. Additionally, an incentive mechanism for the blockchain can encourage the active participation of vehicles, as the usefulness of data often depends on the proximity of a vehicle's location.

The vehicular blockchain is created by vehicles and RSUs to record and maintain critical vehicle data securely, as shown in Figure 1. While the vehicular blockchain grows with new additional blocks of transactions of infotainment data, security and integrity are guaranteed by agreement regarding the new, common view of the blockchain among the vehicles or RSUs. Thus, the blockchain state is updated by the majority of vehicles following a consensus algorithm.

Although the blockchain has already proved its feasibility with the many successes of crypto-currencies and applications, it still faces challenges in terms of throughput and latency for the vehicular cloud. For instance, the block interval of Bitcoin is 10 min on average due to the consensus process, which is not applicable for the dissemination of real-time information such as vehicle traffic and accidents. We enumerate the possible challenges that need to be addressed and the required properties for the vehicular cloud as follows.

- Throughput and scalability: large amounts of data generated over geographical regions need to be recorded or queried simultaneously;
- Latency: a wide range of time limits; safety or traffic data should be exchanged almost in real-time, while the normal road view is not as urgent;
- Reliability: records should be immutable, and actual data should be retrievable;
- Auditability: data to be written in a new block need to be audited by other nodes, not only in terms of inspecting hashes for block serialization but regarding the credibility of the data itself.

Recently, several studies about the vehicular blockchain have been introduced in which the blockchain for the security and privacy protection of safety and traffic data, crowd-sourcing for the error correction of sensor data, or deep learning have been used. Initially, Sharma et al. [4] proposed a vehicular network architecture based on the blockchain system for the intelligent transport systems of a smart city. Using the consensus algorithms of the blockchain, Yang et al. [5] introduced a trust management system for traffic safety and efficiency, and Su et al. [6] proposed a credit-based algorithm for rescue collaboration in disaster regions. Zhang et al. [7] provided a mechanism for a decentralized, secure and reliable database using the vehicular blockchain, and Kang et al. [8] exploited mobile edge computing (MEC) for the same purpose. Furthermore, Li and Feng et al. [9,10] tried to solve a privacy preserving problem for vehicular networking. Regarding crowd-sourcing, Li et al. [11] adopted the vehicular blockchain in order to correct GPS error. Fu and Dai et al. [12,13] introduced a permissioned blockchain architecture for deep reinforcement learning in vehicular edge computing. Most of the aforementioned works focused on the security and privacy protection of vehicular data in the vehicular blockchain.

In this paper, we improve the scalability and reliability of the vehicular blockchain for content sharing using distributed content caching and off-chain trading in the vehicular market place. Our model introduces different roles of users for the blockchain-based market place, listed as follows.

- Content publisher: every vehicle can be a publisher, creating data and publishing them through a blockchain;
- Content consumer: not only can vehicles but anyone who needs vehicular data act as the consumer. For example, the NHTSA (National Highway Traffic Safety Administration) or DoT (Department of Transportation) may have interest in traffic information and videos of the road view. A consumer would search for contents in a blockchain and pay for them;
- Miner: not only can vehicles and RSUs of the vehicular cloud but dedicated blockchain nodes in the Internet act as a "miner", sustaining a public ledger for vehicular data with a specific consensus algorithm. Fixed and wired nodes such as RSUs are more reliable to manage a blockchain rather than vehicles.

In the content market place, each vehicle maintains content data securely with its local storage and shares the data for an incentive, which promotes trust in the contents. This incentive system prevents misbehavior, as this leads to reduced chances of earning an incentive from the cooperation, and encourages users to provide content actively.

The rest of this paper is organized as follows. First, the background of the vehicular cloud and blockchain is briefly reviewed in Section 2. We introduce the *autoCoin*—a blockchain for vehicular cloud—especially regarding the protocol design of *autoCoin* and its implementation in a P2P module, as well as smart contracts in Section 3. In Section 4, we evaluate the performance of *autoCoin* on the real testbed and discuss experimental outcomes. We conclude this paper in Section 5.

2. Background

2.1. Vehicular Cloud

Gerla et al. introduced the concept of the vehicular cloud in [14], which allows vehicles to share not only computing resources such as processors, storages and sensors, but also infotainment data (navigation data with traffic status, dash-cam video, music, movies, etc.) using single or multiple-hop ad-hoc wireless communications.

The vehicular cloud is useful to control vehicles for safety purposes. For example, following vehicles in a fleet can recognize the road curvature in advance if they have access to high-resolution video (e.g., UHD 4K) from the dash-cams of leading vehicles. Recently, a deep neural network for self-driving and safety applications has been trained in the vehicular cloud.

Previously, several research works were conducted regarding resource allocation and energy saving [15–17]. Vehicular information centric networking (VICN) for data or resource query and allocation within the vehicular cloud has been intensively explored in [18–22]. In the VICN, a vehicle broadcasts relevant packets or requests a content name resolution service for host location, and then receives data through a direct or indirect connection between the server and client vehicle. Bouk et al. [23] introduced a vehicular content centric network (VCCN) based on content centric networking (CCN) [24], and Zhang et al. [21] showed that the caching technique improves throughput and increases content availability.

2.2. Blockchain Overview

With the emergence of Bitcoin in 2009, the blockchain—as its underlying core technology—has attracted a great deal of attention from academia and the FinTech industry, and it has shown promise in applications such as digital currency, smart contracts, the Internet of Things (IoT) and privacy services. As its name suggests, it is a chain of blocks which can be open and distributed across the world as a public ledger, allowing individuals and organizations to commit transactions. The chain continuously grows with new blocks of transactions.

The key difference between the blockchain and a conventional database is that the blockchain system can be decentralized and anonymous. The underlying network is created and maintained by distributed nodes, which are not necessarily owned and controlled by a

certain authority (CA). To maintain data integrity in such a decentralized system, a new block including a hash value of the parent block must be verified by all participants. Any tampering attempts can be detected easily, because modifying a past block invalidates all the subsequent blocks due to the hash chain. This consensus algorithm also forces users to select the longest chain among different copies that are created by forking or attack, as tampering is thus made more difficult by the imposed cost of block creation.

The blockchain consists of multiple technologies such as peer-to-peer (P2P) networking, database, synchronization, cryptography, etc. With regard to networking, gossip protocols for managing nodes and block dissemination have been developed for the blockchain module. Furthermore, the blockchain module contains consensus mechanisms such as Byzantine agreement (i.e., a three-phase handshake for agreement among nodes) to synchronize a distributed ledger and provides a programmable interface such as a smart contract for applications.

Bitcoin, as one of the most successful applications of the blockchain, has contributed to the design of the basic architecture of the blockchain beyond crypto-currencies. In the Bitcoin system, participants as computing nodes first check the hash values of several of the previous blocks of the longest chain and verify the authenticity of transactions. Then, a new block with the selected transactions is serialized to the longest chain and propagated to all nodes. This update procedure is driven by the Nakamoto consensus protocol [2]. Details of the consensus algorithms are discussed below.

2.3. Blockchain Consensus Algorithms

For the blockchain, many consensus algorithms have been introduced. First, the Nakamoto consensus protocol for Bitcoin [2] works efficiently for a permissionless public blockchain. The Nakamoto consensus algorithm (i) validates chains—i.e., checking the Proof-of-Work (PoW) on previous several blocks and crashes between transactions—(ii) selects the longest chain among global and local views and (iii) solves a cryptographic puzzle for a new block. This puzzle is very simple to solve using hash query repetition but exhaustive for the CPU. Accordingly, a chain of puzzle blocks is almost immutable because a blockchain participant must investigate a large number of previous blocks before adding a new block; an enormous amount of computing power is required to generate six puzzle blocks for manipulation. However, a guild of miners recently threatened the security of the blockchain, and the tolerated computing power from an adversary is known to be less than 25% [25].

The Nakamoto consensus based on PoW prevents Sybil attacks, but the protocol is costly from an economic point of view and tends to waste too much energy for hash computations. In 2013, Peercoin [26] first implemented a PoS with the concept of coin age, and Blackcoin [27] uses a raw stake instead of the coin age. Later, several PoS algorithms have implemented random algorithms for miner selection; e.g., the follow-the-Satoshi procedure [28,29]. Proof-of-Stake (PoS) regards the possession of coins or blocks as the computing power of PoW, because committing more computing power can allow clients to earn more coins as rewards. However, mining that costs nothing can lead to intentional chain forks and allow double spending attacks; i.e., the nothing-at-stake problem. Delegated Proof-of-Stake (DPoS) adopts a concept of delegates who are elected based on their reputation and are in charge of adding blocks [30]. To impose a cost for being a leader, the destruction of coins is required in the Proof-of-Burn approach [31], and a security deposit is required against dishonest node activity in the Proof-of-Validation approach [32]. Proof-of-Elapsed time has been proposed by Intel for the Sawtooth Lake platform [33], which selects the node with the shortest waiting time, which is determined by a special device: Intel Software Guard Extensions. Proof-of-Capacity (e.g., storage space) is proposed to save energy and encourage contributions to a chain instead of coin stake [34,35]. However, the nothing-at-stake problem still exists.

In contrast to all of the aforementioned algorithms, several algorithms have been proposed for a permissioned private blockchain, which are mostly inherited from Byzan-

tine Fault-Tolerant (BFT) protocols such as Practical BFT (PBFT) [36] for Hyperledger Fabric [37,38], Symbiont [39], R3 Corda [40], Ripple [41], etc. These protocols to exchange messages among nodes verify the transactions and chaining based on a majority of votes. Some of these vote-based protocols use quorum slices to accelerate transaction validation among nodes; e.g., Stellar [42] and Raft [43]. Similarly, Proof-of-Authority (PoA) is now applied for Parity Aura and Ethereum Clique [44] for the permissioned blockchain.

2.4. Blockchain Scalability

Blockchain now faces the significant hurdle of scalability and performance as coin transactions increase exponentially. Bitcoin can achieve only a maximum if 7 transactions per second (TPS), compared to legacy systems such as VISA that can tolerate 2000 TPS. There have been many efforts to scale up blockchains using several approaches. For Bitcoin, Bitcoin-NG [45] has been proposed, which allows a miner of a normal key block to add micro transaction blocks freely before the next key block. Basically, performance and scalability are limited by a delay to synchronize a global view of a blockchain among nodes. However, the delay from the packet delivery over global networks and processing of countless transactions is bounded by the network and sever capacity.

Most approaches consider processing transactions without interrupting the main blockchain. One approach is to build private blockchains for a certain organization or application, for which the transaction verification and management are achieved by permissioned nodes using faster and more efficient consensus algorithms, as shown in [36–38,42,43]. To improve the credibility of the private chains, some states of blocks should be recorded at the main chain periodically. Another approach is to use off-line chains to process transactions in parallel with the main chain. The off-chain technique allows nodes to record only the final transaction at the main chain instead of all intermediate transactions between peer nodes; e.g., Raiden [46] and Lightning networks [47,48] for Ethereum and Bitcoin, respectively. Both L2 protocols over crypto-currency networks support micro and instant payments through a direct payment channel between peer nodes.

Plasma [49] is scalable using hierarchical off-chains, resembling a tree structure. Such an off-chain technique is still being explored and faces some challenges [50]. On the other hand, multiple on-chains are used simultaneously to increase TPS, which is known as sharding (a term typically used in database systems). This sharding-based blockchain allows multiple committees of nodes to process transactions in parallel. Thus, the total number of transactions processed in each consensus round can be multiplied by the number of committees. Since Luu et al. [51] proposed the first sharding-based consensus protocol for public blockchains, several sharding-based blockchains have been introduced: RScoin [52], omniLedger [53] and rapidChain [54].

3. autoCoin: Blockchain for Vehicular Cloud

In this section, we address the architecture of our vehicular blockchain and the mechanism of *autoCoin* for trading contents among vehicles based on the blockchain. The *autoCoin* system enables vehicles to create an eco-system with cooperation in the cloud and to manage infotainment data securely.

3.1. Vehicular Data Management Using Blockchain

The *autoCoin* blockchain is used to store data generated by the vehicular cloud securely, including not only small-sized safety data (e.g., several tens of bytes) but large multi-media data, such as a video or high quality image. In particular, the vehicular cloud can manage these data in three different ways:

- Regional cloud: the video data are stored within a vehicular cloud, where vehicles can access the data using a Geographic Hash Table (GHT) [55]. However, it is complicated to manage the videos transferred over the regional cloud as vehicles enter and exist the

cloud frequently; for instance, vehicles have to copy videos to all incoming vehicles by broadcasting;

- Central cloud: the video data are stored at a central cloud. Accordingly, each vehicle should upload its own data regardless of demands from other vehicles, which demands large amounts of storage in the Internet and expensive cellular connections, even though all parties can easily look up the data;
- Local cache with location registration: VICN allows each vehicle to hold its own data in its local storage, while it registers the data location in the same manner as a Uniform Resource Identifier (URI) (e.g., /blockchain-account/mike/v/1.mp4 (accessed on 18 June 2021)) on a central server as meta-data information for data query. Content data can be queried and delivered by content routing [14,22].

For *autoCoin*, the third approach is most efficient; floating data in the distributed regional cloud cause serious traffic overheads in a VANET, and the centralized cloud can suffer from resource wastage due to redundant or useless content and data. Additionally, large vehicular data (e.g., popular dash-cam "video" data) are not appropriate to record within the blockchain for security purposes, which badly affects the performance of the blockchain. Therefore, our *autoCoin* blockchain system manages only the URI and meta-data (e.g., video size, format, time, geo-location, price, etc) of videos without caching the video data.

Figure 2 illustrates an example block of the *autoCoin* blockchain, where a vehicle with an account address *A* deploys a transaction to the public (e.g., a smart contract address) including the meta-data of a video captured at date 2018-9-1,GPS location (37.38, 126.37) and expiry date. Because of storage limitations in each vehicle, a vehicle should set a lifetime for video data; furthermore, frequent transactions for dead video files can cause overheads in the blockchain.

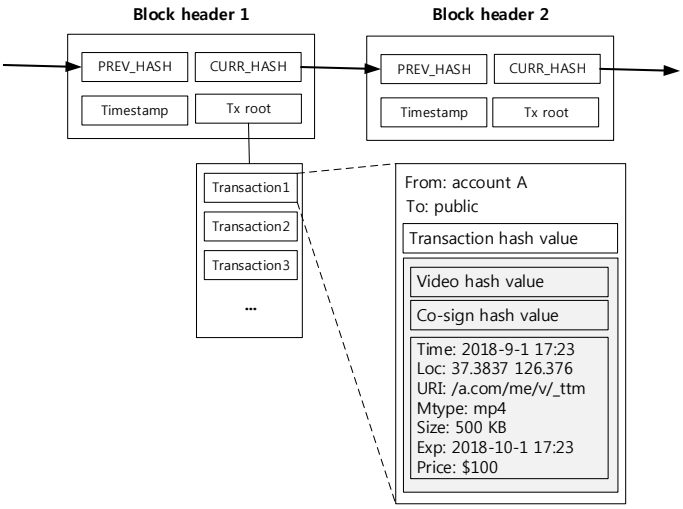


Figure 2. Block structure in a vehicular blockchain.

3.2. Hierarchical Vehicular Blockchain

For the public blockchain, the block interval is controlled by the difficulty of mining, which increases the stability and security of the transactions; e.g., 10 min for Bitcoin and 15 s for Ethereum on average. Unfortunately, it is not simple to increase only the block rate or size to increase the TPS since a high block rate and large blocks can cause frequent forking due to the long propagation delay in blockchain networks [56]. For this, the blockchain can be exposed to potential threats of double spending, even though those forked transactions can be handled by some designated rules and re-chaining based on a consensus algorithm; i.e., GHOST [57]. The performance of the block generation has recently been a critical hurdle in applications demanding tremendous numbers of transactions and real-time payment.

The *autoCoin* system demands a higher capacity and lower latency compared to Bitcoin, as *autoCoin* transactions of vehicular data are generated often and urgently. In brief, if 100 K vehicles (50% of vehicles registered in Seoul stay on the roads for about 1.5 h) generate publication transactions with $\lambda = 0.001$ (e.g., an event every about 15 min) and the transaction size is 500 bytes, then about 50 KB should be recorded every second. Simultaneously, consumers deploy more transactions than publishers, with an additional 750 KB by consumers $\lambda = 0.015$. With this block size, this block dissemination delay over a P2P overlay network including the processing delay of message validation and saving a block in a storage [2,58] varies from several seconds to more than a minute. The block interval should be long enough to propagate transactions to all nodes in order to create a block with choronologically ordered transactions, although several consensus algorithms allow transactions with a limited number of forks of blocks, as used in Ethereum. Furthermore, real-time applications of the vehicular cloud motivate a reduction of the block interval.

To reduce this overhead and improve the TPS, the vehicular blockchain can be split into regional chains since vehicular data are probably meaningful only in a certain region; for example, a video of a car accident in Seoul is not very useful for someone in Los Angeles (LA). Therefore, it is inefficient to manage a globally unique view of this vehicular blockchain, in contrast to conventional digital currencies, which are used around the world. Various sizes of the vehicular blockchain can be considered according to different regional sizes:

- Macro vehicular blockchain: a blockchain deals with vehicular data generated in a certain country, state or big city. Many vehicles and roads can generate tremendous numbers of transactions and also increase the forking probability. However, a single chain is tamper-resilient due to the difficulty of forming a majority for attack and allows vehicles to exchange contents with a single coin;
- Micro vehicular blockchain: a blockchain can be set for a small city or district. The small size of a group allows nodes to reach agreement quickly and significantly reduces the effort to perform synchronization globally, although it could be vulnerable to a majority attack; a small number of nodes for attacks on the chain can be easily gathered. In addition, there could be more than several hundred chains for a country. Therefore, additional management and consensus algorithms for security and multiple regional chains are necessary.

To scale up the capacity of *autoCoin*, transactions are distributed to geographical blockchains that form a hierarchical structure, as shown in Figure 3. For instance, micro vehicular blockchains belong to a macro vehicular blockchain; an LA blockchain would be a child of the California blockchain, and the California blockchain would be a child of the US blockchain. Accordingly, a root vehicular blockchain would cover all macro vehicular blockchains around the world. Therefore, the root or macro chain is managed reliably by fixed nodes such as RSUs and mining servers rather than vehicles, in contrast to the micro-chains.

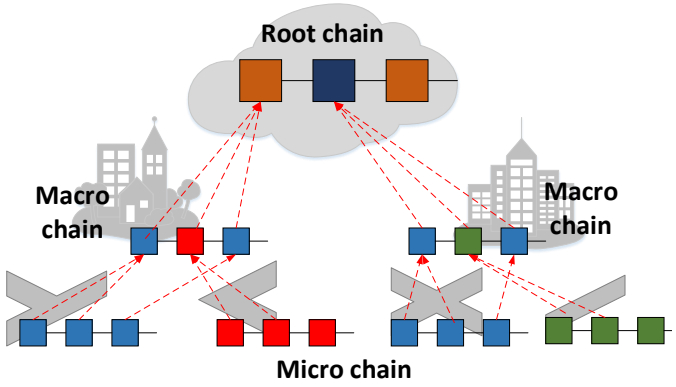


Figure 3. Example of a hierarchical *autoCoin* blockchain that consists of multi-layers such as root, macro and micro blockchains. The macro chain covers a city area and the micro-chain deals with small districts or several roads.

To improve the credibility of the sub-chains, the sub-chains periodically record evidence of their transactions at the super-chain block [49]; i.e., a child blockchain deploys a transaction containing a list of hash values of blocks that have been generated after the last connection with the parent blockchain. The frequency of this logging on the super-chain determines the degree of security of sub-chains.

In a hierarchical vehicular blockchain, local transactions are handled by the regional micro-blockchain. For instance, the transaction of a dash-cam video captured in LA is sealed in a block of an LA-based blockchain. This architecture leads the vehicle to search for contents faster by reducing the search scope and balances the transaction load over distributed micro-chains.

3.3. Bootstrap for Joining a Regional Chain

Since the micro-blockchain for *autoCoin* is created over the vehicular cloud spatially and temporally, block data are loaded or flushed at each vehicle according to a join or leave operation. In other words, vehicles join the vehicular cloud by downloading the blocks of the micro-chain from neighboring vehicles or RSUs as they handover from one cloud to another.

In conventional P2P networks, a central sever manages a list of host information, such as a tracker for bitTorrent [59] or a Distributed Hash Table (DHT) [60]. The former approach is fast, and it is easy to find peers, but it can be unreliable due to a single point of failure, while the latter is robust but it is complicated to implement.

Blockchains also adopt the above two methods to discover peer nodes; they use bootstrapping nodes containing a list of peer locations or the DHT. However, it is challenging to maintain an anchor node that is reliable and always available for bootstrapping in the vehicular cloud. We can consider the static RSU as a candidate for the key peer node that broadcasts the chain and transactions periodically or on-demand.

On the other hand, vehicles can broadcast or multicast messages of peer discovery and block query and acquisition directly toward adjacent vehicles that are probably members of the chain without the bootstrapping information. In consequence, vehicles can obtain the copy of the chain and transactions for a new block.

3.4. Content Market Place for Vehicles

Cooperation among vehicles for the vehicular cloud and blockchain is not easily achieved as it incurs expenses such as storage, wireless network connection (Wi-Fi or cellular subscription), energy, etc. For instance, some vehicles that are uninterested in the contents of the data would not be willing to share their local storage or wireless connections. For this, a digital currency such as Bitcoin can contribute to an incentive mechanism to encourage cooperation.

In this paper, we introduce *autoCoin*, which implements an incentive-compatible algorithm based on digital currency for secure data sharing in the vehicular cloud. The *autoCoin* system creates a coin ecosystem to encourage vehicles to monitor or sense the environment and share that information with others, in addition to sustaining a public ledger for data integrity. Accordingly, vehicles as information providers receive rewards for publishing contents.

The *autoCoin* system consists of vehicles or RSU nodes that are in charge of holding data, maintaining a blockchain as a public ledger and publishing contents. Vehicles are supposed to publish or consume contents, but blockchain mining can be optional according to vehicle capability; for instance, a heavy node with sufficient storage and processing power would play the role of the miner, while a light node would participate only in the content market. Although the number of the mining vehicles in the regional micro-chain is small for secure consensus, super-chains linked in the hierarchy of the vehicular blockchains can validate the updates to the sub-chain.

There are several challenges for data security and trading in the vehicular content market. For example, in the case that vehicles exchange dash-cam videos of a vehicle accident, the integrity of the video needs to be verified; a malicious user may try to delete part or all of the video, or replace it with another video. In this section, we introduce a fair-trading mechanism to solve the following threats on data trading in the vehicular cloud.

- Data integrity: a consumer purchases content based on the meta-data information in the blockchain. Thus, there must be a proof that the received content from a publisher is the same as that noted in the blockchain;
- Data trustworthiness: a publisher can publish content without any verified evidence. For example, a publisher can deploy a video captured at A location with meta-data of location B;
- Trading concurrency: trading without a third party always incurs a risk. Both a publisher and consumer can cheat each other at any step of trading. For instance, a consumer could refuse to pay even though it received data.

Table 1 describes several symbols used in the following sub-sections.

Table 1. Security symbols in the vehicular content market.

Symbol	Description	Information
E	Encryption, ciphering	AES 256
D	Decryption, deciphering	AES 256
H	Hash	SHA 256
K_p	Public key	ECDSA
K_s	Private key	ECDSA
K_m	Message/session key	Temporal ciphering key
K^p	Publisher key	Private or public key
K^c	Consumer key	Private or public key
M	Message/data	Video
X	Ciphered or hashed message/data	Video

3.4.1. Data Integrity

For data integrity, a hash value of video data, $H\{M, K_p^p\}$, using the public key of a publisher, K_p^p, can be used. Figure 2 shows an example block that includes meta-data of the video and its hash value to check data integrity in a blockchain. A consumer requests the video data from the publisher using the video URI. Checking that the video is uploaded by the publisher is performed by comparing the recreated hash value and that available from the block.

3.4.2. Data Trustworthiness

Although a received video satisfies data integrity with the same hash value, the factual relationship of the video with the meta-data noted by the publisher cannot be ensured. A publisher can register a hash of an unrelated video in the meta-data, which would lead a consumer to acquire incorrect information. This problem is almost intractable for certain types of data. Thus, we introduce an exemplary solution only for a dash-cam video in this section.

The most important information in the meta-data, as shown in Figure 2, is time and location. In order to guarantee data trustworthiness, witness vehicles can be cited regarding the meta-data. For example, if several vehicles observe the same accident, they may have similar videos at the same time and location. For this, a publisher can receive co-signs on the video from nearby vehicles using direct communications (e.g., DSRC, WLAN, D2D, etc); a neighbor vehicle would verify a content hash given by the publisher using its private key and then return it back to the publisher with its own address (i.e., account). Finally, a consumer can verify a video hash that is co-signed by neighboring vehicles. Moreover, the consumer can compare videos captured by the publisher and co-signed neighbors to investigate authenticity if the co-signed neighbors also have published videos at the same time. More co-signs enhance the credibility of the published video content; a consumer would definitely prefer content with more witnesses. Similar research has been conducted in [61].

3.4.3. Trading Concurrency and Completeness

For this problem, we propose a solution based on a crypto-currency and smart contract. If a publisher and consumer act rationally, they can betray a peer node for their own benefit. Possible misbehaviors by a publisher and consumer are described below:

- Reject to pay: a consumer does not pay although it receives data;
- Reject to provide content: a publisher does not send content though it obtains payment first;
- Purchasing capability: a consumer who does not have enough money requests contents;
- Contract destruction: one party leaves in the middle of data transfer (some video data can be more than 1 GB in size).

For the first two challenges of trading concurrency, the data should be encrypted so that it cannot be used at the consumer side before payment. The payment should be processed simultaneously with the receipt of a decryption key; however, this concurrency cannot be achieved directly by two peers. Instead, we propose a smart contract to solve this problem, which plays the role of a third party broker between the two peers.

For the proof of purchasing capability, a smart contract can force a consumer to deposit an amount corresponding to the content value written in the blockchain. During the trading procedure, contract destruction can occur intentionally or because of any problem in underlying networks. Thus, if the trading procedure stops in the middle, either the publisher and consumer could initiate contract destruction after a certain grace period. Typically, a publisher does not want to break the contract in the middle because it cannot receive payment in advance, while a consumer can change its mind for several reasons (e.g., downloading latency), meaning that the publisher meaninglessly expends wireless network resources and time. To prevent this, a penalty will be charged to the consumer when the grace period is over; the amount of the penalty could be a percentage of the original value. The consumer cannot withdraw their deposit money during the grace period, which should be configured so that the consumer can restart a transaction after a break.

3.5. Off-Chain Content Trading Using Smart Contract

In *autoCoin*, content information is recorded by the publishers in a Merkle tree of transactions, as shown in the example in Figure 2, and is queried and shown in the distributed application of a consumer. Here, a smart contract allows the publisher and

consumer to execute a transaction script (e.g., the Solidity codes of Ethereum [62]) to deploy or to purchase content.

For the delivery of a video from the publisher to consumer, they need to set up a socket connection. Otherwise, a third-party miner can relay the video through a smart contract, which would cause a severe overhead for the blockchain. Moreover, miners can deny service because of latency that may lead them to lose a mining opportunity for the next blocks; the latency could be more than several minutes for a large file.

Figure 4 shows a procedure of URI resolution using a blockchain. *autoCoin* adopts an ICN framework of name resolution such as data-oriented (and beyond) network architecture (DONA) [63], rather than CCN [24]. Our previous study [22] regarding the ICN architecture in LTE networks provides more information regarding content query and delivery using V2I (RSU) connections. As the vehicle *A* updates its IP address with its own account, the vehicle *B* queries the location of the vehicle *A* through blockchain nodes.

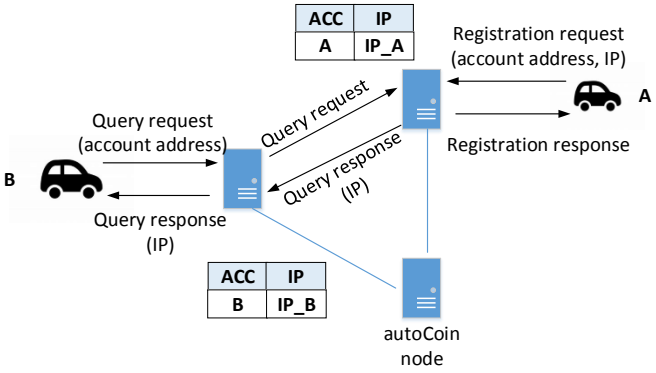


Figure 4. URI resolution service by blockchain nodes.

The principal (i.e., the real-world identifier of a publisher) of the URI can be an account address of a vehicle rather than a typical domain (e.g., /a.com/) because the account address is unique and manageable in the blockchain. In addition, to balance information, each account maintains its current IP address, which is updated periodically or due to a sub-network change event; thus, blockchain nodes build an account resolution table, as shown in Figure 4. Actually, the IP address update occurs infrequently because a VANET consists of a single sub-network and a cellular network typically consists of only a few sub-networks to avoid network mobility. The account address is a long hash value derived by the private key of the owner, which is not easy to use. However, the content query usually achieved inside a distributed application does not ask a user for their account directly; instead, the application abstracts the account address from a URI in a block.

For direct trading between a publisher and consumer through a socket connection, an off-chain trading concept is adopted with a coin deposit and balance proof. This off-chain trading supports not only the direct trading without a third party but also off-loading from the main blockchain. Figure 5 depicts the procedure of content trading through the off-chain. First, the publisher *A* deploys a transaction of content using the smart contract, and then the consumer *B* also executes the smart contract to purchase the content, transferring the cost of the content to the account address of the smart contract as a deposit. The coin deposit prevents double spending from the consumer *B* because the content delivery takes much more time than mining speed (for example, a 100 MB video would take 160 s at 5 Mbps). Accordingly, this deposit cannot be withdrawn until the trading completes or the grace period is over.

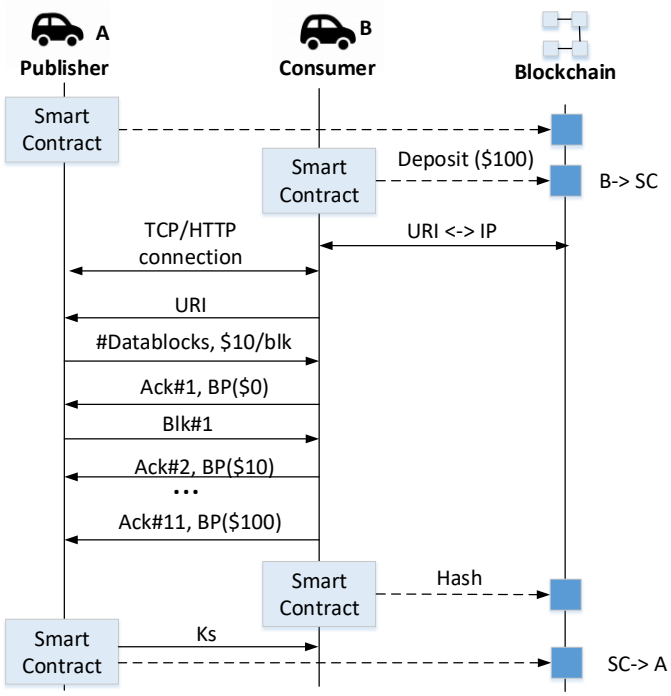


Figure 5. Vehicular content trading procedure based on a smart contract.

Once the consumer B obtains the URI of the content, it opens a TCP/HTTP connection with the publisher A. As the consumer B requests data transmission with the URI, the publisher A notes the total number of data blocks and the price of each block. The consumer B has to send an expected block sequence number and a Balance Proof (BP) noting the accumulated payment for the received data blocks with their own digital signature. In the figure, the content consists of 10 segment blocks and each segment costs $10. Thus, the consumer B increases the BP by 10 for each acknowledgement.

Once transmission finishes, the consumer B verifies the received video data by comparing a generated hash of the received video with a hash value written in the blockchain. In a stricter approach, hash values of segment blocks can be recorded in a structured way (e.g., a Merkle tree) in the publisher's block to verify every received segment block. Once the consumer receives all chunks of data, it executes a smart contract with its own signature, the hash value of the received data and the public key of the publisher, $X = E\{H\{M,\ K_p^p\},\ K_s^c\}$. The smart contract deploys a transaction to write the value of X on the blockchain. After this, the publisher can finalize the contract sometime later or after receiving notification from the consumer through a direct connection. The publisher A conducts the smart contract to reimburse the BP with an encryption key, K_m^p, of the content. Before leaving the encryption key, K_m^p, on the blockchain, the smart contract checks the validity of the consumer's signature—whether the deciphered hash, $X' = D\{X,\ K_p^c\}$ is same as the hash value in the blockchain or not—and then conducts a coin transaction to the publisher A.

The smart contract for the off-chain content trading is shown in Appendix A (functions for content registration, token deposit and reimbursement for a payment channel, and opening and closing a channel), which is written in the Solidity script language and executed over the Ethereum Virtual Machine (EVM).

The *autoCoin* procedure opens a direct payment channel for content trading and closes it after the trading finishes; however, this causes frequent access to the regional blockchain. Instead, vehicles can leave the payment channel open for future trading with the peer node; however, keeping the channel open incurs collateral costs.

Furthermore, multi-hop coin routing by concatenating the channels is possible, such as by packet routing in the Internet, as shown in Figure 6. When the vehicle A and D

have no payment channel, they can trade indirectly using intermediate nodes, *B* and *C* (i.e., payment service providers (PSP)). For this, a conditional payment locked by a secret hash is used, which prevents double spending and reimbursement without the original secret. Therefore, the linked transactions complete atomically at the moment the vehicle *D* releases the secret.

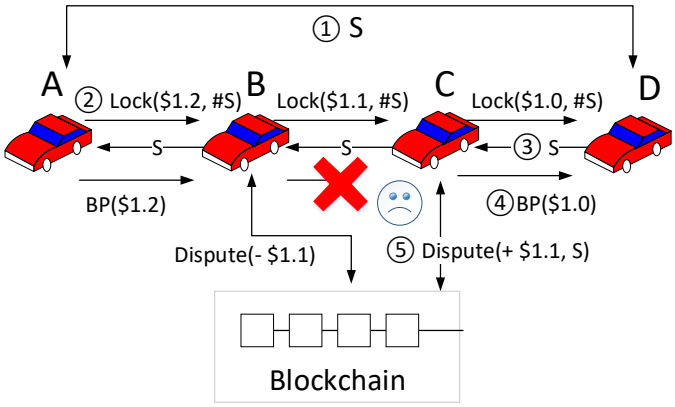


Figure 6. Multi-hop payment example from *A* to *D* in the Raiden network. 1. *A* sends a secret (*S*) to *D* as a proof of coin receipt. 2. Conditional payment by a locking deposit is conducted sequentially. 3. The secret is returned for the locked payment. 4. Each vehicle unlocks the amount using the final BP after comparing the received secret and a secret hash (#*S*). 5. A vehicle creates a dispute if the received secret is incorrect or the deposit is not unlocked in time.

3.6. autoCoin Consensus Algorithm

For the consensus algorithm of *autoCoin*, the PoW is inefficient, and a great deal of calculation is required to solve the hash puzzles, wasting the embedded hardware designed for vehicular intelligence. Thus, we consider a PoS-like consensus algorithm that replaces the stake of CPU power with coins to avoid hardware wastage for *autoCoin*, instead of the original PoS that still suffers from nothing-at-stake problems and common issues such as PoW. Proof-of-Stake at Deposit (PoSD) is the amount of collateral, in which a node that creates the most contents or contributes as a PSP becomes a leader in the next block's generation. This is easily confirmed by other parties checking the amount of coins in the smart contract. Probably, popular publishers or PSPs can be miners as often as key contributors, but a vehicle with even a small stake can be a leader in the micro blockchain. The leader may not abuse a chance for a meaningless attack because the deposit and reward are burned and cancelled by other nodes. Furthermore, the selected miner can be assigned randomly to one of the hierarchical blockchains, which prevents majority attacks, especially on the micro vehicular blockchain.

4. autoCoin Implementation and Experiment

We established an *autoCoin* testbed that consisted of two laptops and three EC instances of Amazon Web Services (AWS) as full nodes. For the three blockchain nodes, two vCPUs with an Intel 2.3 GHz Broadwell E5-2686v4 and 8 GB of memory were used with Linux 18.04 LTS, and two laptops equipped with an Intel core i5 7th, 8 GB of memory, SSD256 and Linux 18.04 LTS were used as vehicles, as shown in Figure 7. The laptops were connected via a Wi-Fi access point assuming vehicles are connected by V2I communication.

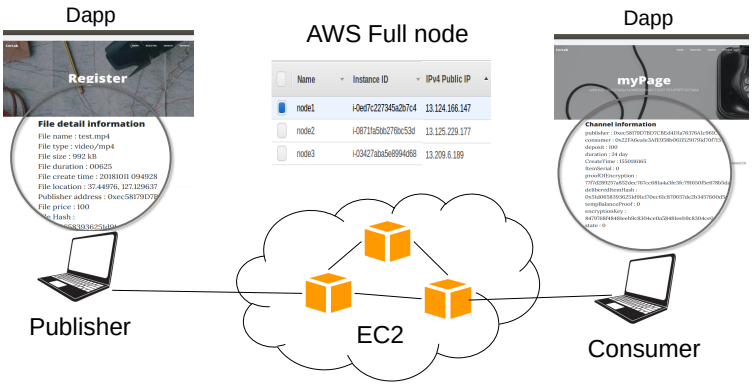
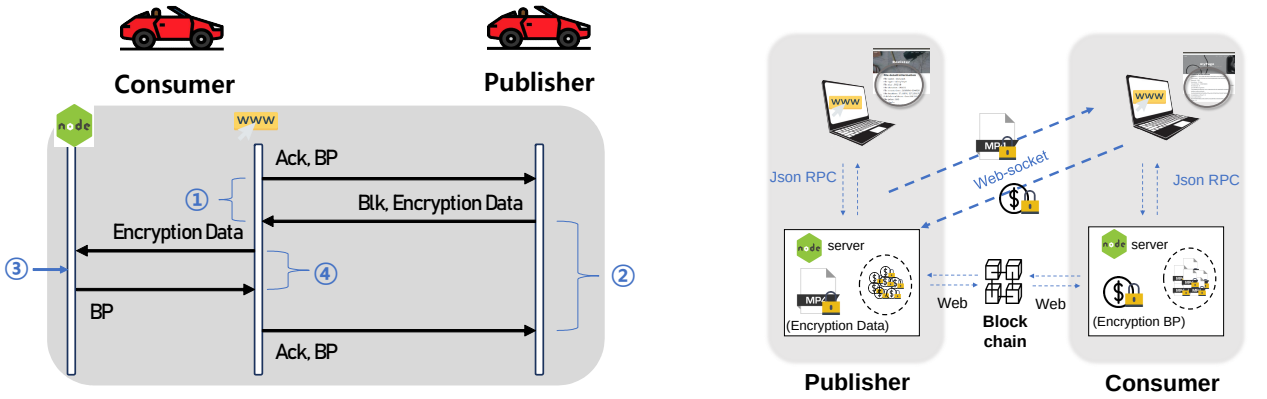


Figure 7. Implementation and testbed of the *autoCoin* blockchain.

The *autoCoin* system is extended from the Ethereum open source data, and so all smart contracts (refer to Appendix A) are written in the Solidity language running on top of EVM. In the publisher laptop, a web-camera captured a video, and then a distributed application (Dapp) registered its meta-data through a smart contract. The consumer Dapp searches for content using titles or data types and allows users to buy one of the contents with a search web page. After purchase, the Dapp allows users to see or manage all contents on their personal web page.

Figure 8a shows a single transaction of the Dapp performing off-chain trading with delay factors. We divide the steps of those delays into stages 1 to 4 in our implementation. Figure 8b describes the details of our implementation for the off-chain transaction. The publisher Dapp uploads a content file locally to a Node.js-based Web-server through a JSON RPC and receives its meta-data through an API (exiftool). Then, it deploys a transaction to save the meta-data in the blockchain. The consumer Dapp searches for the URI of a specific piece of content and queries its IP address from the blockchain nodes. The consumer Dapp creates a web-socket to the publisher's server using the IP address and asks for the content corresponding to the URI. Once the publisher server accepts trading, it ciphers the content using aes-256-CBC and notes the total number of segments and the price of each segment. The consumer Dapp saves the encrypted data chunks on a local web-server using the JSON RPC and replies with the BP and acknowledgement. Typically, the JSON RPC is used for a client to access a remote blockchain node as a remote procedure call. In our system, the web-based Dapp, run by the publisher and consumer, also accesses internal blockchain functions using the JSON RPC as all vehicles are blockchain nodes.

For flow control in the web-socket connection, timers and duplicate acknowledgement are defined. Once the timer on each node expires, the publisher or consumer can attempt to retransmit data or acknowledgement at most three times. During the grace period, they can restart the trading by connecting to each other again. Otherwise, either the publisher or consumer can suspend the trading if they are not willing to continue. If the consumer Dapp completes downloading, it registers a hash value derived from the received content at the blockchain.



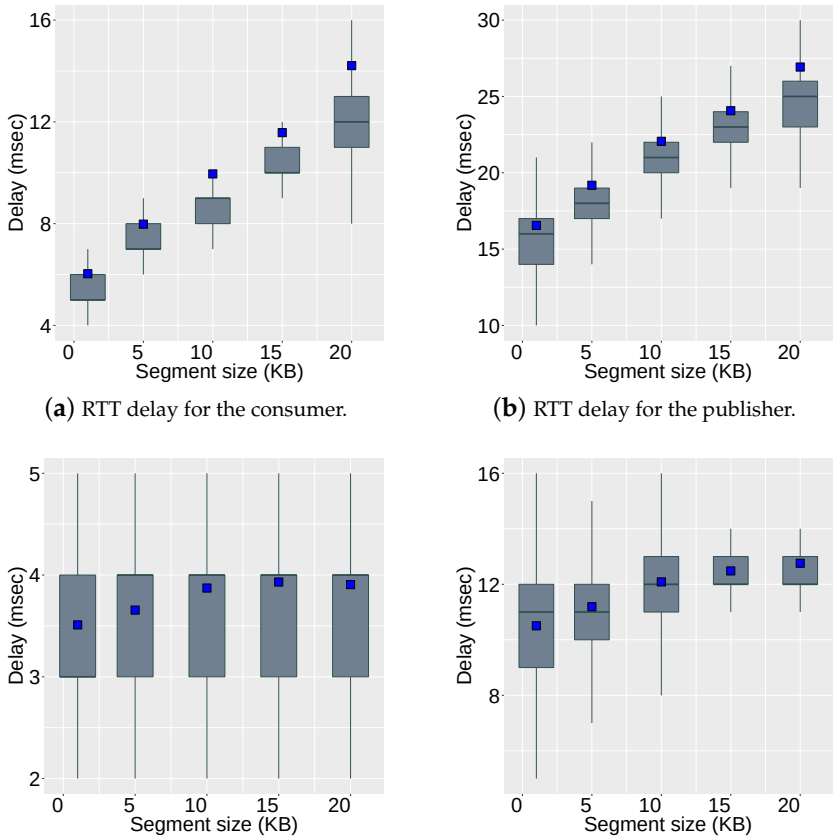
(**a**) Off-chain trading between two vehicles through Wi-Fi access points. (**b**) Implementation details of the off-chain transaction.

Figure 8. Implementation of the *autoCoin* off-chain.

Figure 9 shows the measurement results of the off-chain transaction experiment with our testbed, which were delayed in each step, as shown in Figure 8a. Tables 2–5 show the details of the measurement results for each step; Figure 9 does not include the minimum and maximum outliers noted in the tables. In this measurement, a node traded content with a size of 50 MB with different sizes of segments from 5 KB to 20 KB. To derive the average time delay, we conducted 50 K iterations for each case. The average time to cipher those segments as a preparation procedure was 1.09 s, which was required once the publisher received a content request from a consumer.

The average Round Trip Time (RTT) for the the consumer vehicle (i.e., delay 1 in Figure 8a) increased almost linearly as the segmentation size increased, as can be seen in Figure 9a and Table 2. The average of 8 ms with a 5 KB size increased up to an average of 14 ms with a 20 KB size; approximately 6 ms were consumed to send more than 15 KB data, which means that almost 10 packet transmissions occurred during the 6 ms (i.e., 0.6 ms for each RTT). Supposing that nearby vehicles exchange content and BP using V2V communications, the measured delay is reasonable. Figure 9b depicts the RTT of data and acknowledgement with the BP in the publisher vehicle; i.e., delay 2 in Figure 8a, which is the sum of delay 1 and 4, the transmission delay of packets and internal JSON RPC delay, respectively. Almost half of the delay resulted from the transmission delay. For example, the delay of 27 ms for 20 KB consisted of a transmission delay of 14 ms and JSON RPC delay of 13 ms. Accordingly, this publisher delay increased as the transmission delay increased, as shown in delay 1 of Figure 9a. Delay 4 in Figure 8a was measured as shown in Figure 9b, including BP creation and JSON RPC delay. Here, the BP creation time (i.e., delay 3 in Figure 8a) in Figure 9c was almost consistent with the segment sizes, at about 3.5 ms, even though the average changed slightly from 3.5 to 4 ms. According to the statistics in Table 4, half of the samples showed a delay of between 3 and 4 ms in all cases of segment sizes. Furthermore, the JSON RPC delay increased slightly with segment sizes, as can be seen in Table 5, as the large size data caused slightly more delay in processing for the function call.

For the transaction of content with a size of 50 MB, the total transaction times were 200 and 67.5 s for 5 and 20 KB segments, respectively. The transaction delay for the Dapp including a JSON RPC can be a burden as the number of transactions increases due to the small segment size. As a consequence, using a large data chunk is efficient, but the payment amount should increase in turn, and concurrent trading can suffer. A delay of about a minute is not excessive when buying 50 MB of video content, but this could increase further due to network congestion and delay in the real world.



(**a**) RTT delay for the consumer.

(**b**) RTT delay for the publisher.

(**c**) Delay of balance proof creation for the consumer.

(**d**) Delay of JSON RPC for the consumer, including BP creation and internal processing.

Figure 9. Delay in measurement for off-chain transactions.

Table 2. Statistics of consumer delay in Figure 9a.

Split Size	1	5	10	15	20
Min	2	4	4	4	6
1st Q	5	7	8	10	11
Median	5	7	9	10	12
Mean	6.032	7.981	9.955	11.58	14.21
3rd Q	6	8	9	11	13
Max	316	426	432	431	516

Table 3. Statistics of publisher delay in Figure 9b.

Split Size	1	5	10	15	20
Min	6	8	9	12	11
1st Q	14	17	20	22	23
Median	16	18	21	23	25
Mean	16.55	19.18	22.06	24.07	26.93
3rd Q	17	19	22	24	26
Max	314	378	338	415	604

Table 4. Statistics of BP creation delay in Figure 9c.

Split Size	1	5	10	15	20
Min	1	1	1	1	1
1st Q	3	3	3	3	3
Median	3	4	4	4	4
Mean	3.51	3.656	3.872	3.931	3.906
3rd Q	4.	4	4	4	4
Max	24	23	19	26	19

Table 5. Statistics of JSON RPC delay in Figure 9d.

Split Size	1	5	10	15	20
Min	3	2	3	2	3
1st Q	9	10	11	12	12
Median	11	11	12	12	12
Mean	10.5	11.19	12.09	12.48	12.76
3rd Q	12	12	13	13	13
Max	66	250	70	93	99

The next experiment involved indirect trading using multi-hop payment though intermediate vehicles, as described in Figure 6. For the test-bed shown in Figure 10. we created 10 vehicle instances in the cloud, including two publishers and consumers, respectively, and eight PSP vehicles, similar to vehicles *B* and *C* in Figure 6. We assumed that there were different contents available for the two publishers' and the two consumers' queries and requested the contents independently. However, other configurations for this experiment were the same as before. The path for the multi-hop payment could be different for each transaction; solid arrows indicate the possible payment direction for each vehicle and each vehicle decides their next hop based on the balance of the next-hop vehicles.

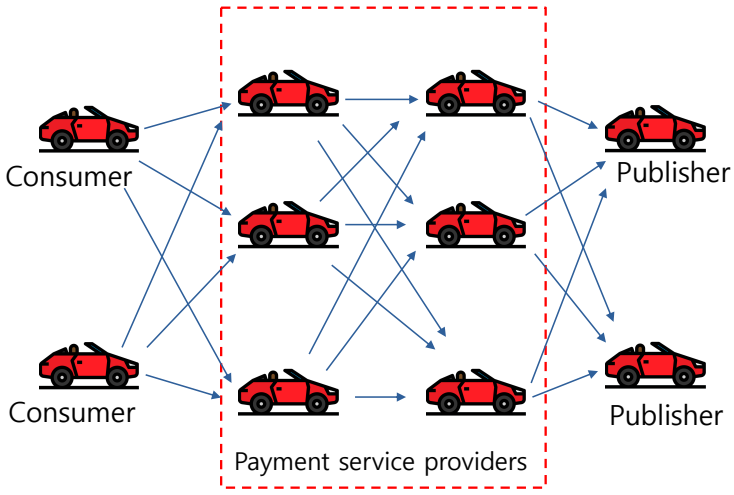
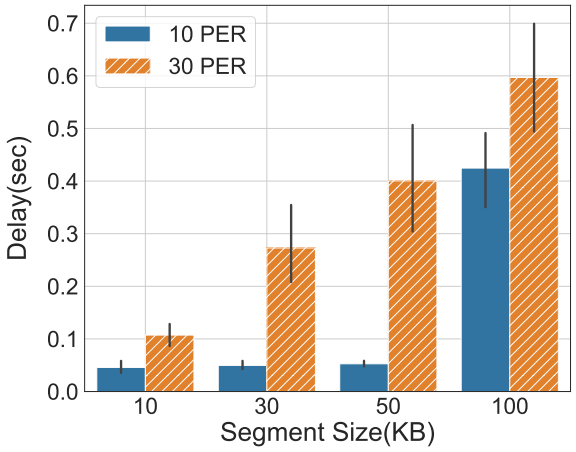


Figure 10. Multiple consumers and publishers with multi-hop *autoCoin* payment.

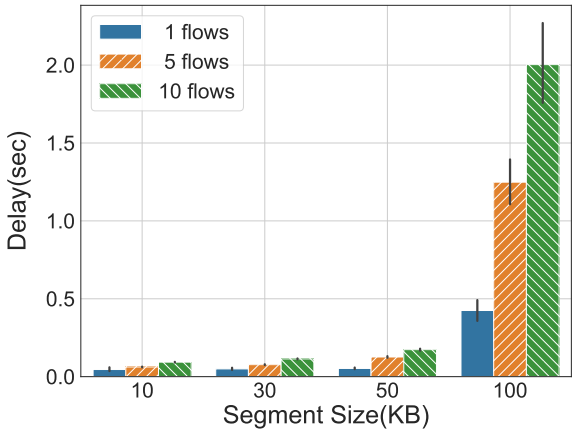
Figure 11a depicts the delay performance of the multi-hop payment and data delivery according to the segment size. As we observed in Figure 9, the delay for segment delivery increased with the segment size. The multi-hop payment also increased exponentially as the off-chain payment between the PSPs imposed an additional delay. Under 10% PER, the delay below 50 KB segment was comparable, but it increased drastically after a segment size of 100 KB. Meanwhile, the 30% PER showed a consistent increase in the

delay with growing segment sizes. Here, we can conjecture that the higher packet loss affects delay considerably due to payment message retransmissions, such as the lock, secret and BP rather than the segment size, especially for multi-hop payments. Consequently, the erroneous wireless connections for the vehicular cloud are critical for the payment throughput. Thus, off-chain payment can use a cellular network for the overlay blockchain network rather than a VANET, because the VANET connectivity is much lower than the cellular network.

The consumers can consume multiple contents at the same time, which causes concurrent payment and content delivery. Figure 11a shows the payment delay results for different numbers of trades, with 1, 5 and 10 data flows with 10% PER. The average payment delay increased in proportion to the number of multi-hop payments, but was comparable until 50 KB segment. For the 100 KB segment, the delay increased exponentially as packet loss occurred severely according to Figure 11b. The latency difference was distinguishable for the cases of flow numbers; the average delay for a segment was 2 s for 10 content flows, while it was around 0.4 s for 1 content flow. Increasing the data flow induces congestion in the payment processing of the PSPs and content forwarding in the underlying network.



(**a**) Packer error rates (PERs) of 10% and 30% for a payment channel.



(**b**) Trading of 1, 5 and 10 pieces of content in parallel.

Figure 11. Delay of multi-hop payment with varying segment sizes.

5. Conclusions

In this paper, we introduce an architecture of a vehicular blockchain which provides security and integrity for vehicular sensor data over distributed vehicles and road-side units. We propose *autoCoin*, a secure trading system for vehicular content. For these real-time vehicular transactions, we suggest a hierarchical side-chain system together with off-chain transactions. The side-chains are generated according to geo-locations and payment channels are created on the regional side-chain. We implement the *autoCoin* system by extending the Ethereum and its virtual machine, in which web-based distributed applications using smart contracts are implemented and executed in mobile computers. According to the measurement results of our *autoCoin* testbed, the off-chain trading delay would feasibly allow two vehicles to exchange a 50 MB file and tokens within about a minute. For multi-hop payment, wireless channel error is more critical than the segment size, especially as the number of payment flows increases. Our simple experiment shows the feasibility of the proposed scheme in terms of delay throughput, but more complicated scenarios need to be investigated in future work.

Author Contributions: W.K. conceived and designed the main idea and wrote the paper; K.R. performed the experiments. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Technology development Program of MSS (no. G21S299419101) and the ICT development RnD program of MSIT (no. NRF-2017R1C1B1006607).

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Smart Contract Codes for autoCoin

Listing A1 shows the smart contract codes written in the Solidity language [62] for content information and a channel structure for the off-chain transaction. The meta-data include some parameters to describe content, as we discuss in Section 3.1, and a hash value for data integrity. Furthermore, witness information can be added as digital signatures of witness addresses. The channel structure maintains information between two vehicles, such as the two vehicles' account addresses, the deposit amount, duration (i.e., grace period), content serials, balance proof and encryption keys for the off-chain trading. The channel has three states—initial, suspended and complete—and either a publisher and consumer can suspend and resume the off-chain transaction within the grace period.

Listing A1. Content information and channel structure.

```
  //Content MetaData struct
struct ItemMetaData {
    address publisher;
    uint256 itemPrice;
    string itemLocation;
    string itemSize;
    string itemType;
    uint256 itemDuration;
    uint256 itemCreateTime;
    bytes32 itemHash;
    bytes32[] cosigners;
    bool isRegistered;
    }
//Channel struct
struct Channel {
    address publisher;
    address consumer;
    uint256 deposit;
    uint256 duration;
    uint256 creatTime;
    uint256 itemSerial;
    string proofOfEncryption;
    bytes32 deliveredItemHash;
    string encryptionKey;
    uint256 state;
    bool isRegistered;
    }
```

Solidity codes in Listing A2 transfer tokens from a publisher account to an account of the smart contract or reimburse them to a consumer account. These two functions are called by a consumer and publisher, respectively. In Listing A3, the consumer creates a channel for the off-chain transaction by calling the deposit function. After content delivery, the publisher calls the complete function of Listing A4, where the smart contract compares two hash values given by the consumer and blockchain and checks the balance proof that contains the consumer signature before exchanging a token and key. If one of those conditions is not satisfied, the function returns with error messages. Otherwise, the smart

contract transfers the deposit to the publisher and stores the encryption key for the content in the blockchain. Later, the consumer can query the key in the complete channel state.

Listing A2. Smart contract for token deposit and reimburse.

```
//Channel deposit
function deposit(address consumer, uint256 value) internal {
    _balances[consumer] = _balances[consumer].sub(value);
]
//Channel complete
function complete(address publisher, uint256 value) internal {
    _balances[publisher] = _balances[publisher].add(value);
}
```

Listing A3. Smart contract for channel creation.

```
//Channel create
function createChannel(uint256 _deposit, uint256 _itemSerial) public
    returns(uint256) {
   uint256 _channelSerial = channelSerial++;
   require(registeredItemMetaData[_itemSerial].isRegistered == true);
    require(registeredItemMetaData[_itemSerial].itemPrice == _deposit);

   //deposit
   deposit(msg.sender, _deposit);

   registeredChannel[_channelSerial].publisher =
   registeredItemMetaData[_itemSerial].publisher;
   registeredChannel[_channelSerial].consumer = msg.sender;
   registeredChannel[_channelSerial].deposit = _deposit;
   registeredChannel[_channelSerial].duration = duration;
   registeredChannel[_channelSerial].createTime = now;
   registeredChannel[_channelSerial].itemSerial = _itemSerial;
   registeredChannel[_channelSerial].state = 0;
   registeredChannel[_channelSerial].isRegistered = true;

   return _channelSerial;

}
```

Listing A4. Smart contract for channel completion.

```
//Channel completion
function completeChannel(uint256 _channelSerial, string memory _encryptionKey,
    bytes32 _balanceProof,
uint8 v, bytes32 r, bytes32 s, uint256 _balance) public
    ongoingChannel(_channelSerial)
publisher(_c) {
   require(comparisonContent(getItemSerial(_channelSerial),
   registeredChannel[_channelSerial].deliverredItemHash));

   //check balance proof
   require(checkBalanceProof(registeredChannel[_channelSerial].consumer,
   _balanceProof, v, r, s));

   //check balance
    require(registeredChannel[_channelSerial].deposit == _balance);

   //deposit token refund
   complete(msg.sender, registeredChannel[_channelSerial].deposit);

   registeredChannel[_channelSerial].encryptionKey = _encryptionKey;
   registeredChannel[_channelSerial].state = 2;

}
```

References

1. Kim, W. Evolutionary Game for Content Cache in a mm-Wave-Based Vehicular Fog. *Electronics* **2020**, *9*, 1794. [CrossRef]
2. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 18 June 2021).
3. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
4. Sharma, P.K.; Moon, S.Y.; Park, J.H. Block-VN: A distributed blockchain based vehicular network architecture in smart City. *J. Inf. Process. Syst.* **2017**. [CrossRef]
5. Yang, Z.; Yang, K.; Lei, L.; Zheng, K.; Leung, V.C. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet Things J.* **2018**, *6*, 1495–1505. [CrossRef]
6. Su, Z.; Wang, Y.; Xu, Q.; Zhang, N. LVBS: Lightweight vehicular blockchain for secure data sharing in disaster rescue. *IEEE Trans. Dependable Secur. Comput.* **2020**. [CrossRef]
7. Zhang, X.; Chen, X. Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network. *IEEE Access* **2019**, *7*, 58241–58254. [CrossRef]
8. Kang, J.; Yu, R.; Huang, X.; Wu, M.; Maharjan, S.; Xie, S.; Zhang, Y. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet Things J.* **2018**, *6*, 4660–4670. [CrossRef]
9. Li, M.; Zhu, L.; Lin, X. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet Things J.* **2018**, *6*, 4573–4584. [CrossRef]
10. Feng, Q.; He, D.; Zeadally, S.; Liang, K. BPAS: Blockchain-assisted privacy-preserving authentication system for vehicular ad hoc networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4146–4155. [CrossRef]
11. Li, C.; Fu, Y.; Yu, F.R.; Luan, T.H.; Zhang, Y. Vehicle position correction: A vehicular blockchain networks-based GPS error sharing framework. *IEEE Trans. Intell. Transp. Syst.* **2020**. [CrossRef]
12. Fu, Y.; Yu, F.R.; Li, C.; Luan, T.H.; Zhang, Y. Vehicular blockchain-based collective learning for connected and autonomous vehicles. *IEEE Wirel. Commun.* **2020**, *27*, 197–203. [CrossRef]
13. Dai, Y.; Xu, D.; Zhang, K.; Maharjan, S.; Zhang, Y. Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4312–4324. [CrossRef]
14. Gerla, M.; Lee, E.K.; Pau, G.; Lee, U. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Korea, 6–8 March 2014; pp. 241–246.
15. Shojafar, M.; Cordeschi, N.; Baccarelli, E. Energy-efficient adaptive resource management for real-time vehicular cloud services. *IEEE Trans. Cloud Comput.* **2016**, *7*, 196–209. [CrossRef]
16. Zheng, K.; Meng, H.; Chatzimisios, P.; Lei, L.; Shen, X. An SMDP-based resource allocation in vehicular cloud computing systems. *IEEE Trans. Ind. Electron.* **2015**, *62*, 7920–7928. [CrossRef]

17. Yu, R.; Zhang, Y.; Gjessing, S.; Xia, W.; Yang, K. Toward cloud-based vehicular networks with efficient resource management. *IEEE Netw.* **2013**, *27*, 48–55. [CrossRef]
18. Lee, E.; Lee, E.K.; Gerla, M.; Oh, S.Y. Vehicular cloud networking: Architecture and design principles. *IEEE Commun. Mag.* **2014**, *52*, 148–155. [CrossRef]
19. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 26–36. [CrossRef]
20. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 1024–1049. [CrossRef]
21. Zhang, M.; Luo, H.; Zhang, H. A survey of caching mechanisms in information-centric networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1473–1499. [CrossRef]
22. Kim, W. Beyond LTE-advance for information centric networking. *Comput. Stand. Interfaces* **2017**, *49*, 59–66. [CrossRef]
23. Bouk, S.H.; Ahmed, S.H.; Kim, D. Vehicular content centric network (VCCN) a survey and research challenges. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, 13–17 April 2015; pp. 695–700.
24. Jacobson, V.; Mosko, M.; Smetters, D.; Garcia-Luna-Aceves, J. *Content-Centric Networking*; Whitepaper; Palo Alto Research Center: Palo Alto, CA, USA, 2007; pp. 2–4.
25. Eyal, I.; Sirer, E.G. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* **2018**, *61*, 95–102. [CrossRef]
26. King, S.; Nadal, S. 2021. Available online: https://https://www.peercoin.net/whitepapers/peercoin-paper.pdf (accessed on 18 June 2021).
27. Vasin, P. BlackCoin's Proof-of-Stake Protocol v2. Available online: https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf (accessed on 18 June 2021).
28. Nxt Wiki. 2021. Available online: https://https://www.jelurida.com/nxt (accessed on 18 June 2021).
29. Bentov, I.; Gabizon, A.; Mizrahi, A. Cryptocurrencies without proof of work. In Proceedings of the International Conference on Financial Cryptography and Data Security, Christ Church, Barbados, 22–26 February 2016; pp. 142–157.
30. BitShare. Available online: https://how.bitshares.works/en/master/ (accessed on 18 June 2021).
31. P4Titan. Slimcoin: A Peer-to-Peer Crypto-Currency with Proof-of-Burn. Available online: https://slimcoin.info/whitepaperSLM.pdf (accessed on 18 June 2021).
32. Tendermint. 2021. Available online: https://tendermint.com/ (accessed on 18 June 2021).
33. Intel. Sawtooth v1.0.1. 2021. Available online: https://www.hyperledger.org/use/sawtooth (accessed on 18 June 2021).
34. Park, S.; Pietrzak, K.; Alwen, J.; Fuchsbauer, G.; Gazi, P. Spacemint: A cryptocurrency based on proofs of space. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 480–499.
35. BurstCoin. 2021. Available online: https://www.burst-coin.org/ (accessed on 18 June 2021).
36. Castro, M.; Liskov, B. Practical Byzantine fault tolerance. In Proceedings of the OSDI, New Orleans, LA, USA, 22–25 February 1999; Volume 99, pp. 173–186.
37. Cachin, C. Architecture of the hyperledger blockchain fabric. In Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, USA, 25 July 2016; Volume 310.
38. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; p. 30.
39. Symbiont. 2021. Available online: https://www.symbiont.io/ (accessed on 18 June 2021).
40. R3 Corda. 2021. Available online: https://www.corda.net/ (accessed on 18 June 2021).
41. Schwartz, D.; Youngs, N.; Britto, A. 2021. Available online: https://ripple.com/files/ripple_consensus_whitepaper.pdf (accessed on 18 June 2021).
42. Mazieres, D. *The Stellar Consensus Protocol: A Federated Model for Internet-Level Consensus*; Stellar Development Foundation: San Francisco, CA, USA, 2015.
43. Ongaro, D.; Ousterhout, J.K. In search of an understandable consensus algorithm. In Proceedings of the USENIX Annual Technical Conference, Philadelphia, PA, USA, 19–20 June 2014; pp. 305–319.
44. De Angelis, S.; Aniello, L.; Baldoni, R.; Lombardi, F.; Margheri, A.; Sassone, V. PBFT vs. proof-of-authority: Applying the CAP theorem to permissioned blockchain. In Proceedings of the Italian Conference on Cyber Security, Milan, Italy, 6–9 February 2018.
45. Eyal, I.; Gencer, A.E.; Sirer, E.G.; Van Renesse, R. Bitcoin-NG: A Scalable Blockchain Protocol. In Proceedings of the NSDI, Santa Clara, CA, USA, 16–18 March 2016; pp. 45–59.
46. Raiden Network. 2021. Available online: https://raiden.network/ (accessed on 18 June 2021).
47. Lightning Network. 2021. Available online: https://lightning.network/ (accessed on 18 June 2021).
48. Poon, J.; Dryja, T. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. 2021. Available online: https://lightning.network/lightning-network-paper.pdf (accessed on 18 June 2021).
49. Plasma. 2021. Available online: https://plasma.io/plasma-deprecated.pdf (accessed on 18 June 2021).
50. Herrera-Joancomartí, J.; Pérez-Solà, C. Privacy in bitcoin transactions: New challenges from blockchain scalability solutions. In Proceedings of the Modeling Decisions for Artificial Intelligence, Sant Julià de Lòria, Andorra, 19–21 September 2016; pp. 26–44.

51. Luu, L.; Narayanan, V.; Zheng, C.; Baweja, K.; Gilbert, S.; Saxena, P. A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 17–30.
52. Danezis, G.; Meiklejohn, S. Centrally banked cryptocurrencies. *arXiv* **2015**, arXiv:1505.06895.
53. Kokoris-Kogias, E.; Jovanovic, P.; Gasser, L.; Gailly, N.; Syta, E.; Ford, B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–23 May 2018; pp. 583–598.
54. Zamani, M.; Movahedi, M.; Raykova, M. RapidChain: Scaling blockchain via full sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–18 October 2018; pp. 931–948.
55. Ratnasamy, S.; Karp, B.; Yin, L.; Yu, F.; Estrin, D.; Govindan, R.; Shenker, S. GHT: A geographic hash table for data-centric storage. In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, USA, 28 September 2002; pp. 78–87.
56. Gervais, A.; Karame, G.O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H.; Capkun, S. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 3–16.
57. Sompolinsky, Y.; Zohar, A. Secure high-rate transaction processing in bitcoin. In Proceedings of the International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico, 26–30 January 2015; pp. 507–527.
58. Decker, C.; Wattenhofer, R. Information propagation in the bitcoin network. In Proceedings of the IEEE P2P 2013 Proceedings, Trento, Italy, 9–11 September 2013; pp. 1–10
59. Cohen, B. Incentives build robustness in BitTorrent. In Proceedings of the Workshop on Economics of Peer-to-Peer systems, Berkeley, CA, USA, 21–22 February 2003; Volume 6, pp. 68–72.
60. Kaashoek, M.F.; Karger, D.R. Koorde: A simple degree-optimal distributed hash table. In Proceedings of the International Workshop on Peer-to-Peer Systems, Berkeley, CA, USA, 21–22 February 2003; pp. 98–107.
61. Kim, M.; Lim, J.; Yu, H.; Kim, K.; Kim, Y.; Lee, S.B. ViewMap: Sharing private in-vehicle dashcam videos. In Proceedings of the14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), Boston, MA, USA, 27–29 March 2017; pp. 163–176.
62. Dannen, C. *Introducing Ethereum and Solidity*; Springer: Berlin/Heidelberg, Germany, 2017.
63. Koponen, T.; Chawla, M.; Chun, B.G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007; pp. 181–192.