Pegem Journal of Education and Instruction, Vol. 15, No. 4 , 2025 (pp. 5207-5218)

RESEARCH ARTICLE WWW.PEGEGOG.NET

SecureChain-MD: A Consortium Blockchain Framework with Privacy-Preserving Smart Contracts for Malware Detection in Mobile Devices

Madupathi Parimala*[1] , Dr. Gaurav Tyagi[2]

1. Research Scholar, Dept. of Computer Science & Engineering, Chaudhary Charan Singh University, Meerut, Uttar Pradesh, India, 250005. email: pari.parillu@gmail.com
2. Associate Professor, Dept. of Computer Science & Engineering, Chaudhary Charan Singh University, Meerut, Uttar Pradesh, India, 250005.

Abstract-

Mobile devices, particularly those running Android, are increasingly targeted by malware due to their open-source nature and widespread adoption. Traditional detection methods struggle with code obfuscation, encryption, and evolving malware families. Blockchain has emerged as a promising solution for decentralized trust and tamper-proof evidence storage, while privacy-preserving smart contracts enhance compliance and confidentiality. This paper proposes SecureChain-MD, a unified consortium blockchain framework that integrates multi-feature malware detection with cryptographic smart contracts. The framework reduces false positives, improves detection accuracy, and ensures privacy-preserving evidence management. Experimental validation demonstrates superior performance compared with existing detection algorithms, with enhanced scalability and compliance readiness.

Keywords: Privacy-Preserving Smart Contracts, Malware Detection, Mobile Security , Android Malware , Multi-Feature Detection Model, Sensitive Behavior Graph (SBG).

1. INTRODUCTION

Malware detection in mobile devices remains a critical challenge. Static analysis methods are efficient but vulnerable to obfuscation, while dynamic analysis provides deeper insights but suffers from limited coverage and high overhead. Consortium blockchain offers a partially decentralized model in which trusted members collaboratively detect malware and store evidence. Privacy-preserving smart contracts further ensure secure authentication, compliance, and confidentiality in mobile ecosystems. SecureChain-MD merges these approaches to provide a scalable, trustworthy, and privacy-preserving malware detection system.

Blockchains like Bitcoin reach consensus not only on a stream of data but also on computations involving this data. In Bitcoin specifically, the data include money transfer transactions proposed by users, and the computation involves transaction validation and updating a data structure called the unspent transaction output set which, imprecisely speaking, keeps track of users' account balances.

Newly emerging cryptocurrency systems such as Ethereum embrace the idea of running arbitrary user-defined programs on the blockchain, thereby creating an expressive decentralized smart contract system.

In this paper, we consider smart contract protocols in which parties interact with such a blockchain. Assuming that the decentralized consensus protocol is secure, the blockchain can be thought of as a conceptual party (in reality decentralized)that can be trusted for correctness and availability, but not for privacy. Such a blockchain provides a powerful abstraction for the design of distributed protocols. The blockchain's expressive power is further enhanced by the fact that blockchains naturally embody a discrete notion of time, i.e., a clock that increments whenever a new block is mined. The existence of such a trusted clock is crucial for achieving financial fairness in protocols. In particular, malicious contractual parties may prematurely abort from a protocol to avoid financial payment.

However, with a trusted clock, timeouts can be employed to make such aborts evident, such that the blockchain can financially penalize aborting parties by redistributing their collateral deposits to honest,

non-aborting parties. This makes the blockchain model of cryptography more powerful than the traditional model without a blockchain, where fairness is long known to be impossible.

in general, when the majority of parties can be corrupt. In summary, blockchains allow parties, mutually unbeknownst, to transact securely without a centrally trusted intermediary, thereby avoiding high legal and transactional costs.

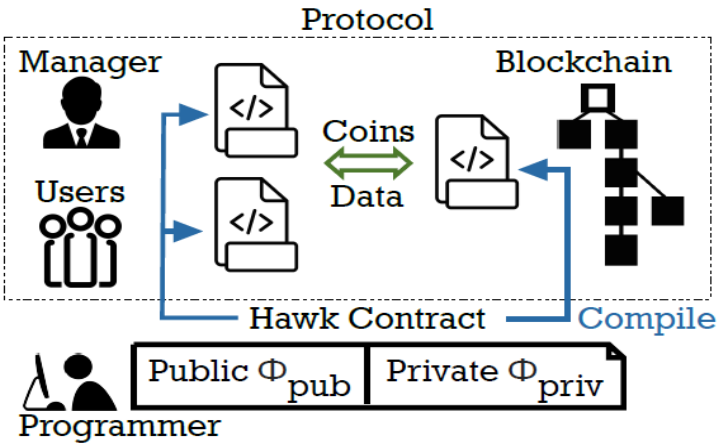

Figure 1. Cryptographic protocol between the blockchain and the users

A framework for building privacy-preserving smart contracts. With Hawk, a non-specialist programmer can easily write a Hawk program without having to implement any cryptography. Our Hawk compiler is in charge of compiling the program to a cryptographic protocol between the blockchain and the users. As shown in Figure 1, a Hawk program contains two parts:

1) A private portion denoted by -priv which takes in parties' input data (e.g., choices in a "rock, paper, scissors" game) as well as currency units (e.g., bids in an auction). -priv performs computation to determine the payout distribution among the parties. For example, in an auction, the winner's bid goes to the seller, and the others' bids are refunded. The private Hawk program, PRIV, is meant to protect the participants' data and the exchange of money.

2) A public portion denoted pub that does not touch private data or money.

Our compiler will compile the Hawk program into the following pieces, which jointly define a cryptographic protocol Between users, the manager, and the blockchain.

Generally, the existing malware detecting technologies for Android devices could be divided into two categories: static-based analysis and dynamic-based analysis. The static-based

analysis method used the decompiler technology or performance analysis of the control flow and data flow in the smali intermediate code, which was applicable for automatic analysis through a large amount of software samples. But it could not solve the problems of code obfuscation, encryption, and other issues, which could only be performed dynamically.

execution.

The dynamic-based analysis method stimulated the execution of software to avoid the code obfuscation and encryption problems. However, the coverage of its dynamic testing code always is not enough. Besides, some malicious programs might camouflage themselves when running under simulators . Moreover, there were a variety of malware across different families with various features, which also increased the difficulty of detection. At the same time, the extraction of some features in the existing methods required a high time cost.

2. RELATED WORKS

The static-based analysis method can implement efficient and automatic analysis in some ways. However, it cannot detect code obfuscation or encryption, and it is insufficient to decrypt malicious code during dynamic execution. Also, it uses a coarse-grained detection

approach for information flows between applications, which makes it easy to produce false positives [21].

The dynamic-based analysis method of Android-based software collects application behavior information during its operation. They can solve problems such as code obfuscation and encryption. Nevertheless, malware usually has a well-designed triggering mechanism when facing dynamic tests, while some malicious programs can detect their own operating environments and automatic crash behaviors when running under the simulator.

The architecture of a consortium blockchain in malware detection.

Our CB-MDEE framework consists of a consortium blockchain and a public blockchain, as shown in Fig. 2, where CB represents the Consortium Blockchain and PB represents the Public Blockchain. The CB is the core chain, composed of the members in distributed malware detection. organizations. These members build a fact-base of the distributed malicious code. The PB is the application chain, open to any user who needs to provide detection and evidence services for joining as a member node.

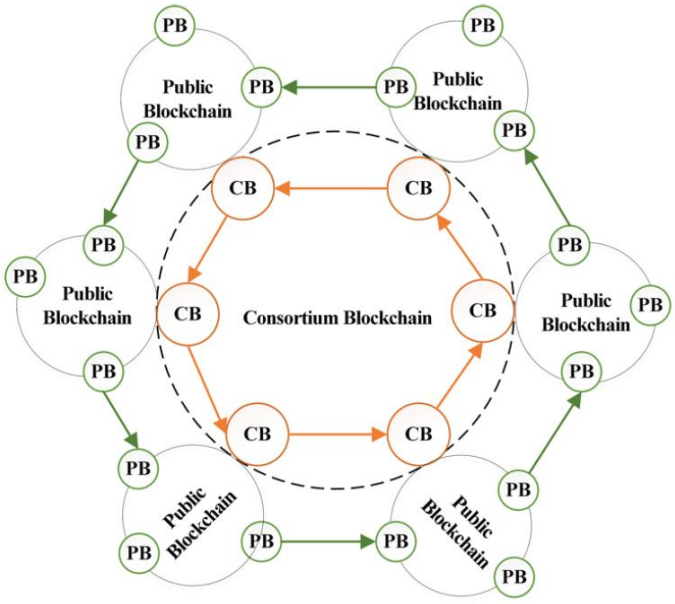

Figure 2: Structure of consortium blockchain

III. THE BLOCKCHAIN MODEL OF CRYPTOGRAPHY

A. The blockchain model. We begin by informally describing the trust model and assumptions. We then propose a formal framework for the "blockchain model of cryptography" for specifying and reasoning about the security of protocols.

In this paper, the blockchain refers to a decentralized set of miners who run a secure consensus protocol to agree upon the global state.

We therefore regard the blockchain as a conceptual trusted party that is trusted for correctness and availability, but not trusted for privacy. The blockchain not only maintains a global ledger that stores the balance for every pseudonym, but also executes user-defined programs.

More specifically, we make the following assumptions: • Time. The blockchain is aware of a discrete clock that increments in rounds. We use the terms rounds and epochs interchangeably. • Public state. All parties can observe the state of the blockchain. This means that all parties can observe the public ledger on the blockchain, as well as the state of any user-defined blockchain program (part of a contract protocol). • Message delivery. Messages sent to the blockchain will arrive at the beginning of the next round. A network adversary may arbitrarily reorder messages that are sent to the blockchain within the same round. This means that the adversary may attempt a front-running attack (also referred to as the rushing adversary by cryptographers), e.g., upon observing that an honest user is trading a stock, the adversary preempts by sending a race transaction trading the same stock. Our protocols should be proven secure despite such adversarial message delivery schedules.

B. Formally Modeling the Blockchain Our paper adopts a carefully designed notational system such that readers may understand our constructions without understanding the precise details of our formal modeling. We stress, however, that we give formal, precise specifications of both functionality and security, and our protocols are formally proven secure under the Universal Composability (UC) framework. In doing so, we make a separate contribution of independent interest: we are the first to propose a formal, UC-based framework for describing and proving the security of distributed protocols that interact with a blockchain — we refer to our formal model as “the blockchain model of cryptography”.

Programs, wrappers, and functionalities. """ In the remainder of the paper, we will describe ideal specifications, as well as the pieces of the protocol executed by the blockchain, the users, and the manager, respectively, as programs written in pseudocode. We refer to them as the ideal program (denoted Ideal), the blockchain program (denoted B or Blockchain), and the user/manager program (denoted UserP) respectively.

All of our pseudo-style programs have precise meanings in the UC framework. To “compile” a program to a UC-style functionality or protocol, we apply a wrapper to a program. Specifically, we define the following types of wrappers:

• The ideal wrapper F(·) transforms an ideal program IdealP into a UC ideal functionality F(IdealP).

• The blockchain wrapper G(·) transforms a blockchain program B to a blockchain functionality G(B). The blockchain functionality G(B) models the program executing on the blockchain.

• The protocol wrapper Π(·) transforms a user/manager program UserP into a user-side or manager-side protocol Π(UserP).

One important reason for having wrappers is that wrappers implement a set of common features needed by every smart contract application, including time, the public ledger, pseudonyms, and adversarial reordering of messages— in this way, we need not repeat this notation for every blockchain application.

IV.MULTI-FEATURE DETECTION OF ANDROID MALWARE

The structure of multi-feature malware detection on Android-based mobile devices is shown in Fig.3. For an Android-based software, first establish the CFR. Second, compare the CFR of the malware family database and compare its similarity with each family to detect the software. If the software is malware, its belonging family should be given. Third, extract evidence of malicious codes and add the related information to the malware fact-base.

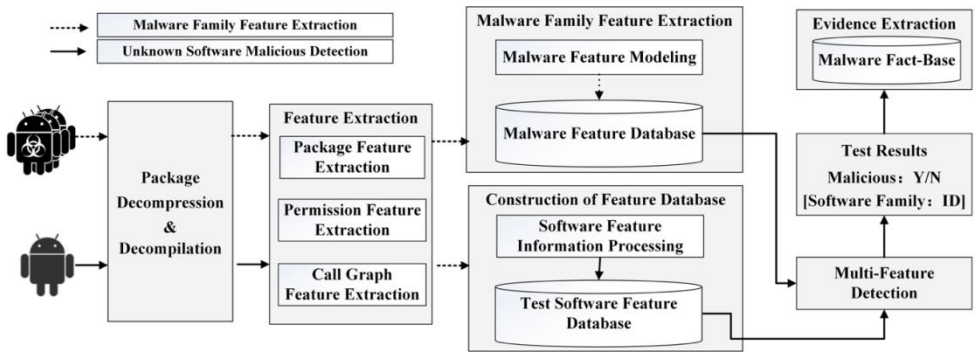

Figure 3: Multi-feature detection of Android malware

A. Multi-feature detection

For improving the detection ability of malware variants, we propose a fuzzy comparison method by marking functions to detect the multi-feature. When calculating the similarity during extracting the CFR of the test software, we adopt three different parts of the software's CFR.

B. The fact-base of malicious codes based on blockchain

In the case of detecting malicious codes, if a problem is found in the result of the detection, the related information about the detection will be collected to constitute a block of data and submitted to the fact-base. It can be used as evidence of malicious codes and to update the feature base of the malware family. The information can also be analyzed to provide support for the new detecting rules. In our CBMDEE, each data block includes some testing fact information, such as the sensitive behavior feature, the installation package feature, and the

permission feature of the software. There is some basic information, including a timestamp, the hash value of the previous block, and a random number for verifying hash values.

IV. EXPERIMENTAL RESULTS

A. Data sets and experimental environment settings

We perform our experiments in the CPU-- Intel Core i7-3770, whose main memory is 16 GB. The operating system is Ubuntu 15.10. The malware data set comes from the Drebin data set. The benign software data is extracted by modifying the crawler programing -- Google Play. We select 4486 malware samples and 2140 benign software samples from 24 malware families. For example, a.dex file in a real-world Android malware, whose MD5 value is 3de513a148400b457dd8d8fa9238804db3ec031a0b526d4a04 b77e5112aa2dcf [22]. For the benign software, we manually perform tests to determine that it is not a malware by using VirusTotal (https://www.virustotal.com/#/home/upload) and Dr. Web Anti-virus (https://download.drweb.com/?lng=en).

Table 1. List of malware detections

Feature	Characteristic Component	Probability
Installation Package Feature	Feature Vectors	(1.0, 0.17, 0.0,0.028)
Permission Features	android.permission.ACCESS_NETWORK_STATE	1.0
	android.permission.INTERNET	1.0
	android.permission.WRITE_EXTERNAL_STAORAGE	1.0
	android.permission.READ_PHONE_STATE	0.94
	...	...
Call Graph Feature	323,20,41,8255	1.0
	293,91	0.96
	...	...

In the malware samples, 75% samples from the 24 malware families were selected as features for extraction to construct a signature database. For example, the partial feature information of the malware family Gappusin [34] is obtained after the analysis as shown in Table. Due to the large number of files and the list of files in the package, here we give partial features of permission and call graph features in Table IV, where the methods involved in the call graph feature are replaced by numbers.

B. Experiments of malware detection

In order to verify our CB-MDEE model, we first classify and detect the remaining 25% samples by using the constructed feature database of the malware family. The results are shown in Table V. Due to the limited space, we only present the results of 10 families.

The first column of the table is the names of malware families, the second column is the numbers of samples for feature extraction, and the third column isnumbers of tested samples. The fourth and fifth columns are the numbers of false-negatives (FN) and false-positives (FP) in Test results, respectively. FN indicates that a software p belongs to the malware family M, but it is judged as a benign software or is classified as the sample number of other families.

FP indicates that a software p does not belong to the malware family M, but it is incorrectly classified as the sample number of the M family. The last column is the overall accuracy of the current malware family classification, which is calculated by dividing the number of correctly sorted samples by the total number of samples.

C. Performance Evaluation

We evaluated the performance for various examples using an Amazon EC2 r3.8xlarge virtual machine. We assume a maximum of 2 64 leaves for the Merkle trees and present results for both 80-bit and 112-bit security levels. Our benchmarks actually consume at most 27GB of memory and 4 cores in the most expensive case. Tables 2 and II illustrate the results – we focus on evaluating the zk-SNARK performance of the zk-SNARK circuits for the manager circuit finalize for different applications. The manager circuits are the same for both security levels. MUL denotes multiple (4) cores, and ONE denotes a

single core.

Table 2: k-SNARK performance Performance

		swap	rps	auction		crowdfund	
#Parties		2	2	10	100	10	100
KeyGen(s)	MUL	8.6	8.0	32.3	300.4	32.16	298.1
	ONE	27.8	24.9	124	996.3	124.4	976.5
Prove(s)	MUL	3.2	3.1	15.4	169.3	15.2	169.2
	ONE	7.6	7.4	40.1	384.2	40.3	377.5
Verify(ms)		8.4	8.4	10	19.9	10	19.8
EvalKey(GB)		0.04	0.04	0.21	1.92	0.21	1.91
VerKey(KB)		3.3	2.9	12.9	113.8	12.9	113.8
Proof(KB)		0.28	0.28	0.28	0.28	0.28	0.28
Stmt(KB)		0.22	0.2	1.03	9.47	1.03	9.47

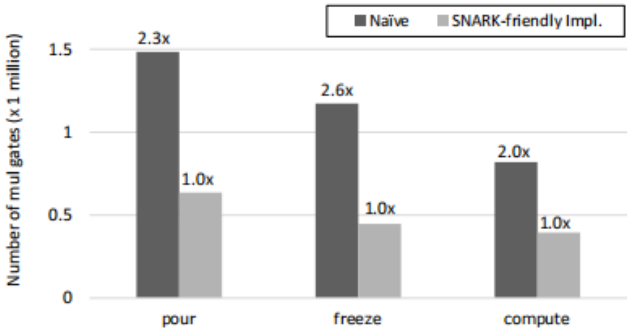

Figure 4. k-SNARK Performance

Gains from using SNARK-friendly implementations for the user-side circuits: power, freeze, and compute at 80-bit security. since all other computation time is negligible in comparison. We highlight some important observations: • On-chain computation (dominated by zk-SNARK verification time) is very small in all cases, ranging from 9 to 20 milliseconds. The running time of the verification algorithm is just linearly dependent on the size of the public statement, which is far smaller than the size of the computation, resulting in a small verification time.

V. CONCLUSION

Assuming a blockchain trusted for correctness and availability (but not for privacy), an interesting notion of fairness which we refer to as "financial fairness" can be attained, as shown by recent works. In particular, the blockchain can financially penalize aborting parties by confiscating their deposits. Earlier works in this space focus on protocols that retrofit the artifacts of Bitcoin's limited scripting language. Specifically, a few works use Bitcoin's scripting language to construct intermediate abstractions such as "claim-refund" or "multi-lock" and build atop these

abstractions to construct protocols. In a generic blockchain model where the blockchain can run Turing-complete programs, we can improve the efficiency of financially fair MPC protocols. Fair MPC with private deposits. We further illustrate how to perform financially fair MPC using private deposits, i.e., where the amount of deposits cannot be observed by the public. framework CB-MMDE, to detect and classify malware on Android-based mobile devices through blockchain technology. We analyze multiple features of malware families, propose a malware feature model--MFM for mobile devices based on the Android system, and design a malware detection and classification algorithm.

REFERENCES

[1] A. Dehghantanha, H. Karimipour, and A. Azmoodeh, "Cybersecurity in smart farming: Canada market research," *arXiv preprint arXiv:2104.05183*, 2021.

[2] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[3] A. Ericsson, *Ericsson Mobility Report: On the Pulse of the Networked Society*, Tech. Rep. EAB-14, 61078, Sweden: Ericsson, 2015.

[4] A. Ferdowsi and W. Saad, “Brainstorming generative adversarial networks (BGANs): Towards multi-agent generative models with distributed private datasets,” *arXiv preprint arXiv:2002.00306*, 2020.

[5] T. M. Fernández-Caramés and P. Fraga-Lamas, “A review on the use of blockchain for the internet of things,” *IEEE Access*, vol. 6, pp. 32979–33001, 2018.

[6] W. Gao, W. G. Hatcher, and W. Yu, “A survey of blockchain: Techniques, applications, and challenges,” in *Proc. 27th Int. Conf. Computer Communication and Networks (ICCCN)*, 2018, pp. 1–11.

[7] J. Gu, B. Sun, X. Du, J. Wang, Y. Zhuang, and Z. Wang, “Consortium blockchain-based malware detection in mobile devices,” *IEEE Access*, vol. 6, pp. 12118–12128, 2018.

[8] M. U. Hassan, M. H. Rehmani, and J. Chen, “Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions,” *Future Generation Computer Systems*, vol. 97, pp. 512–529, 2019.

[9] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, “Multitier fog computing with large-scale IoT data analytics for smart cities,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 677–686, 2017.

[10] White House, *Consumer Data Privacy in a Networked World: A Framework for Protecting Privacy and Promoting Innovation in the Global Digital Economy*. Washington, DC, USA: White House, 2012, pp. 1–62.

[11] S. Ji, P. Mittal, and R. Beyah, “Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1305–1326, 2016.

[12] P. Kairouz *et al*., “Advances and open problems in federated learning,” *arXiv preprint arXiv:1912.04977*, 2019.

[13] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, “Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory,” *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10700–10714, 2019.

[14] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, “Reliable federated learning for mobile networks,” *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[15] L. U. Khan, W. Saad, Z. Han, and C. S. Hong, “Dispersed federated learning: Vision, taxonomy, and future directions,” *arXiv preprint arXiv:2008.05189*, 2020.

[16] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, “Federated learning for internet of things: Recent advances, taxonomy, and open challenges,” *arXiv preprint arXiv:2009.13012*, 2020.

[17] H. Kim, J. Park, M. Bennis, and S.-L. Kim, “Blockchained on-device federated learning,” *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.

[18] Y. Kim, J. Sun, H. Yu, and X. Jiang, “Federated tensor factorization for computational phenotyping,” in *Proc. 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2017, pp. 887–895.

[19] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.

[20] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,” in *Proc. IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 839–858.

[21] K. Krombholz, K. Busse, K. Pfeffer, M. Smith, and E. von Zezschwitz, “‘If HTTPS were secure, I wouldn’t need 2FA’: End user and administrator mental models of HTTPS,” in *Proc. IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 246–263.

[22] U. H. T. Le, N. H. Tran, Y. K. Tun, Z. Han, and C. S. Hong, “Auction based incentive design for efficient federated learning in cellular wireless networks,” in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[23] T. Lepoint and M. Naehrig, “A comparison of the homomorphic encryption schemes FV and YASHE,” in *Int. Conf. Cryptology in Africa*, Springer, 2014, pp. 318–335.

[24] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, “CreditCoin: A privacy-preserving blockchain-based incentive announcement network for communications of

smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[25] M. Li, L. Zhu, and X. Lin, "Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4573–4584, 2018.

[26] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *arXiv preprint arXiv:1907.09693*, 2019.

[27] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[28] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.