APPB: Anti-Counterfeiting and Privacy-Preserving Blockchain-Based Vehicle Supply Chains

Can Zhang, Liehuang Zhu, *Senior Member, IEEE*, Chang Xu, *Member, IEEE*, Kashif Sharif, *Senior Member, IEEE*, Rongxing Lu, *Fellow, IEEE*, and Yupeng Chen

***Abstract*—The vehicle supply chain industry is in a stage of rapid development with the popularization of electric vehicles. However, counterfeits, also known as fake products, present a non-negligible effect on vehicle supply chains. Counterfeit vehicular parts have created significant losses in both monetary value and social terms. For example, tainted and sub-standard tires or batteries used in vehicles even have caused traffic accidents or loss of human life. Due to these dominant reasons, many researchers have focused on building anti-counterfeiting supply chain systems. More recently, several blockchain-based solutions have been presented, whereas most of them lack privacy protection. In this paper, aiming at the issue, we propose a reliable and Anti-counterfeiting and Privacy-Preserving Blockchain-based vehicle supply chains, called APPB. APPB is characterized by offering dual advantages, by utilizing the immutability of blockchain and keeping the privacy of product selling information and business relationship. Both security analysis and experimental results are conducted, and the results indicate that APPB can achieve privacy protection and anti-counterfeiting with acceptable efficiency.**

***Index Terms*—Vehicle supply chain, anti-counterfeiting, privacy protection, blockchain.**

I. Introduction

THE emergence of Electric Vehicles (EVs) has promoted the prosperity of the vehicle supply chain industry. Due to the global nature of the vehicle supply chain, more and more countries and enterprises have established global collaborations to reach efficient vehicular part delivery for greater profits. For example, like many high-tech products, the manufacture of EV batteries needs to be completed in different locations, which increases the complexity of vehicle supply chains.

Counterfeits, also known as fake products, have become a pressing issue in the vehicle supply chain industry that is difficult to solve. The European Office of Intellectual Property (EUIPO) estimated that counterfeit tire and battery selling cause a total loss of 2.4 billion every year across the European Union. Besides, a report[1] written by World Trademark Review (WTR) shows that counterfeit vehicular parts may not guarantee the protection to the vehicular user and instead cause injury or even death in the traffic accident. Unfortunately, for consumers, it is difficult to distinguish between genuine and counterfeit vehicular parts only by their appearance.

To fight against counterfeits, research efforts have focused on how to establish an anti-counterfeiting supply chain to prevent the counterfeits from their production to propagation. In recent years, RFID (Radio Frequency Identification) is utilized to achieve anti-counterfeiting in some proposed schemes, where an NFC (Near Field Communication) tag called EPC (Electronic Product Code) is used to identify each product with a unique PID (Product Identifier). When a seller sells a batch of products to a buyer, the buyer first checks the validity of these products by reading its EPC. If the check is failed, such products can be considered counterfeits, and the buyer can reject them.

However, the centralized architecture of the traditional anti-counterfeiting supply chain may bring about challenges of unreliability. For instance, the centralized server undertakes a high computational burden, and the data stored on the server can be compromised or modified when the server suffers from attacks by either external or internal adversaries (e.g., malicious sellers). Hence, there exists a strong demand to build a reliable supply chain that resists counterfeiting attacks.

In recent years, blockchain, as a new type of distributed and immutable ledger, has become more popular, and a design choice for different solutions. Based on World Economic Forum prediction, 10% of global GDP will be recorded using blockchain technology, by 2027 [1]. A blockchain contains several blocks linked by time sequence as a chain, where each block contains some transactions. Blockchain was first used to realize distributed ledgers and trust-less cryptocurrencies which can offset the requirement of central banks. Following this, a series of blockchain-based applications have been proposed beyond cryptocurrency to solve the problems arising from centralization and unreliability. In these applications, blockchain is utilized to guarantee decentralized, reliable, and transparent data storage and verification. Blockchain also uses cryptographic

Manuscript received 3 August 2021; revised 4 June 2022; accepted 14 July 2022. Date of publication 3 August 2022; date of current version 19 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grants 61972037, 61872041, and U1836212, and in part by the Industrial Internet innovation and development project 2020 under Grants TC200H033 and TC200H01S. The review of this article was coordinated by Prof. Amiya Nayak. *(Corresponding authors: Liehuang Zhu; Chang Xu.)*

Can Zhang, Liehuang Zhu, and Chang Xu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: canzhang@bit.edu.cn; liehuangz@bit.edu.cn; xuchang@bit.edu.cn).

Kashif Sharif is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: 7620160009@bit.edu.cn).

Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton E3B 5A3, Canada (e-mail: rlu1@unb.ca).

Yupeng Chen is with the China Automotive Engineering Research Institute Co., Ltd, Chongqing 400000, China (e-mail: chenyupeng@caeri.com.cn).

Digital Object Identifier 10.1109/TVT.2022.3196051

[1]https://www.worldtrademarkreview.com/anti-counterfeiting/counterfeit-automotive-parts-increasingly-putting-consumer-safety-risk

primitives to guarantee its security and transparency. Hence, for adversaries, it is impossible to modify the data stored on the blockchain unless it controls the majority power of the whole blockchain network.

To prevent the shortcomings of existing centralized anti-counterfeiting supply chains, some blockchain-based approaches are proposed to realize a decentralized and anti-counterfeiting supply chain. In these schemes, blockchain transactions are sent from the seller to the buyer, and both the product information (e.g., manufacturer, production date, PID) and the product flow (e.g., logistical information and ownership transfer information) are stored as public blockchain transactions. To achieve anti-counterfeiting, buyers can get the product information and track the product flow by its PID. If there exists any inconsistency between the received product and the information stored on the blockchain, the buyer can reject this product as a counterfeit.

However, these schemes present some privacy issues. Due to the transparency of the public blockchain, everyone can obtain the information stored on block data, which leads to privacy leakage. In blockchain-based vehicle supply chains, sensitive information such as sales information and business relationship can be inferred from transactions. The *product selling information* reveals the sales volume, status, and profits of a given kind of product (e.g., how many products have been sold last year). The *business relationship* reveals the supplier and customer information of a given entity. These two kinds of information are usually considered business secrets that should be protected. This means that in current blockchain-based supply chains, transparency (i.e., everyone can query the ownership information of products) and privacy protection (i.e., the sensitive information cannot be leaked) are difficult to be achieved simultaneously.

Motivated by these shortcomings in existing schemes, we proposed a novel, reliable and anti-counterfeiting vehicle supply chain scheme with privacy protection, called APPB. To the best of our knowledge, APPB is the first decentralized and reliable vehicle supply chain scheme that satisfies the anti-counterfeiting requirement of modern vehicle supply chain solutions and protects the privacy of product selling information and business relationship. The main contributions of this work are three-fold as follows.

- We present a novel and complete system architecture and threat model of a decentralized and privacy-preserving vehicle supply chain that supports anti-counterfeiting. In the system model, we introduce a trusted supervisor in trust-less circumstances to realize product tracing without losing the decentralization property of blockchain.
- We propose APPB, the first decentralized privacy-preserving vehicle supply chain that eliminates the propagation of counterfeit vehicular parts. We creatively make use of both cryptographic primitives and blockchain technology to balance the contradiction between transparency and privacy. In APPB, everyone can check the Proof-of-Ownership of vehicular parts, whereas sensitive information such as product selling information and business secrets cannot be leaked. Note that the proposed scheme can also be used in relevant supply chain solutions.
- We make a thorough security analysis to prove that APPB achieves privacy protection and realizes anti-counterfeiting. We also make a comprehensive experimental evaluation to show that APPB has acceptable efficiency and it is practical in real-world scenarios.

The rest of this paper is organized as follows. In Section II, we describe the related work. In Section III, we make a brief introduction to blockchain, smart contract, and cryptographic primitives used in the proposed scheme. The formalized system model, security model, and design goals are defined in Section IV. Detailed descriptions of APPB are presented in Section V. Security analysis is given in Section VI, and experimental evaluation is presented in Section VII. Finally, Section VIII concludes this paper.

II. Related Work

Counterfeits can be considered one of the most challenging issues in supply chains. In the vehicle supply chain, counterfeit vehicular parts may cause severe traffic accidents or even death. Due to the detrimental nature of counterfeit products, researchers have proposed several anti-counterfeiting approaches.

In some proposed schemes, RFID was utilized to achieve anti-counterfeiting. Staake et al. [2] first discovered the potential of RFID in anti-counterfeiting. They concluded that using a cheap and passive tag to store a unique EPC of each product is practical in many applications. Zanetti et al. [3] proposed a simple and practical approach to detect cloned RFID tags in supply chains. It creates a random tail to RFID tags which makes cloning detectable by a centralized detector. Shi et al. [4] proposed a Batch Clone Detection (BCD) scheme that only needs a few bits to deter counterfeits. Khalil et al. [5] surveyed previous research where RFID is used in anti-counterfeiting systems and explored possible future directions to achieve better security, privacy, and adaptability. They also proposed an RFID-based anti-counterfeiting scheme for retail environments [6], and they proved that the proposed scheme can resist several attacks such as tag cloning attacks and modification attacks.

Because of the prevalence of blockchain, some blockchain-based supply chain systems have been presented to solve the centralization issues of traditional approaches and achieve system reliability. In the supply chain scenario, Tan et al. [7] provided a qualitative thematic analysis to illustrate the impacts on food supply chains when embracing blockchain. Chang et al. [8] made a statistical analysis of relevant literature to analyze the trend and potential application of the blockchain and smart contract used in supply chains. Gonczol et al. [9] also reviewed existing solutions & implementations for blockchain-based supply chains. They all believe that introducing blockchain technology to supply chains can bring huge development potential and provide numerous opportunities for researchers and companies.

To realize a blockchain-based anti-counterfeiting supply chain, Toyoda et al. [10] presented an anti-counterfeiting blockchain-based product ownership management system (POMS). This system uses the Ethereum blockchain and smart contract to enable customers to verify product ownership. In POMS, a customer can reject the product when the seller cannot provide the correct ownership information. Their experimental

TABLE I
COMPARISON BETWEEN APPB AND EXISTING WORKS

Scheme	Anti-Counterfeiting	Decentralization	Anonymity	Product Information Protection	Business Relationship Protection	Public Verification	Traceability
[2]	✓	×	×	×	×	×	×
[10]	✓	✓	×	×	×	✓	✓
[11]	✓	✓	✓	✓	×	×	×
APPB	✓	✓	✓	✓	✓	✓	✓

results show that the managing cost is less than 1 USD when the number of owner transfers is small. Alzahrani et al. [12] proposed a blockchain-based supply chain that can resist counterfeiting attacks with acceptable performance and the security level. Yiu et al. [13] also presented a blockchain-based supply chain to resist counterfeiting attacks and achieve product traceability.

Unfortunately, for blockchain-based anti-counterfeiting supply chain services, the schemes mentioned above do not consider privacy issues about the leakage of product selling information and business relationship. More specifically, everyone can trace the product by invoking smart contracts. Hence, adversaries can infer the above sensitive information by performing inference attacks on the blockchain, which can be considered privacy leakage.

Although some privacy-preserving blockchain-based supply chains [11], [14], [15] have been presented in the academic area, other blockchain-based anti-counterfeiting supply chains are also leveraged such as Everledger,[2] Luxtag,[3] and tradeLens[4] by enterprises. To the best knowledge, none of them focus on how to achieve anti-counterfeiting in the blockchain-based vehicle supply chain that can protect the product selling information and the business relationship of the entities participating in the vehicle supply chain, which forms the basic motivation of our work.

The comparison between APPB and typical existing schemes is listed in Table I, which shows that the proposed APPB scheme achieves all the listed features compared with other schemes. Note that APPB realizes conditional anonymity and conditional tracing, which means only the trusted authority can trace the product flow and find the identities participating in one product flow.

III. PRELIMINARIES

In this section, we present the basic information related to blockchain, the smart contract, and some cryptographic primitives used in the proposed scheme.

A. Blockchain

Blockchain technology is a combination of smart contracts, consensus algorithms, and distributed ledgers, which enables decentralized, immutable, pseudonymous, and transparent transactions. It was first used in Bitcoin and has been utilized for different applications since then. Below, we briefly identify key properties and core functionality.

- *Decentralization:* Blockchain network can be established without any centralized authority. For instance, the Bitcoin blockchain network is maintained by nodes that are distributed all around the world. Hence, blockchain cannot be controlled, monopolized, or compromised by any single company, organization, or even government.
- *Immutability:* Once data on a transaction has been committed to the blockchain ledger, it cannot be modified or deleted. In the blockchain, every peer node stores a complete replica of blockchain data, and its integrity can be guaranteed by cryptographic technologies. This mechanism makes blockchain an append-only ledger.
- *Pseudonymity:* Blockchain uses pseudonyms to identify different blockchain users. For users, their pseudonyms can be generated by themselves, and everyone can possess one or more unique pseudonyms. Hence, users' personal information cannot be directly inferred by pseudonyms on the blockchain.
- *Transparency:* Blockchain can be classified to: *public blockchain*, *private blockchain*, and *consortium blockchain*. Anyone can access the information stored on public and several consortium blockchains which realize data transparency. Some blockchain-based applications utilize the blockchain to publish the information because of this property.

Blockchain can also be considered as a linked structure. Fig. 1 illustrates the block structure of a simple chain. Each block includes a block head and a block body. Every block links to its previous block by *Previous Block Hash*, which is the hash value of the previous block. To check the integrity of transaction data, *Transaction Root Hash* is used to verify whether the transactions stored in a block are valid. These two kinds of hash values achieve the immutability of blockchain data. Any changes in one block require a change in all following blocks, which is impossible for adversaries.

Each blockchain user can use their unique blockchain address (i.e., pseudonym) to send their assets to other addresses by blockchain transaction. Formally, we use function $T_{id} \leftarrow \mathit{SendTx}(addr_s, addr_r, amount, data)$ to define the process of sending a transaction that stores additional data, which will be used in Section V. The first three parameters include sender's address $addr_s$, receiver's address $addr_r$, and the $amount$ to transfer. The $data$ represents the data stored in this transaction. This function returns a transaction id T_{id} that will be used to query the detailed information of this transaction stored on the blockchain.

[2]https://everledger.io
[3]https://www.luxtag.io
[4]https://www.ibm.com/blockchain/container-logistics

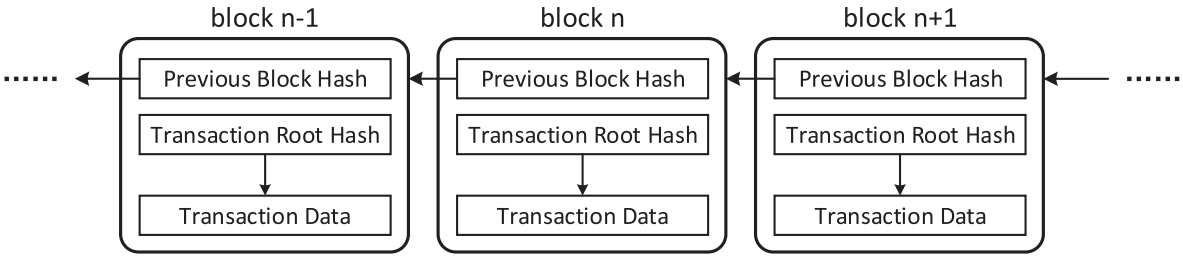

Fig. 1. Block Structure of a Simple Blockchain.

B. Smart Contract

The smart contract is a self-executing piece of code, which implements the terms and conditions of a (smart) contract [16]. The concept of the smart contract is extensively utilized in blockchain systems. Ethereum[5] and Hyperledger Fabric[6] is the largest consortium blockchain platform that supports smart contract, respectively.

The smart contract realizes the programmability of blockchain, and there are a large number of smart contracts used in blockchain-based applications. In Fabric, the smart contract (also called *chaincode*) can be written in multiple languages such as C++, Java, and Go. The contract creator can deploy it to the blockchain with a unique contract address $addr_c$, and others can invoke it when knowing $addr_c$.

Formally, we use $R \leftarrow \textit{Invoke}(addr_i, addr_c, func, params)$ to define the process of invoking a smart contract, which will also be used in Section V. The $func$ stands for the function that will be invoked in this contract, $params$ represents the parameter of contract functions, $addr_i$ and $addr_c$ represents the invoker's address and contract address, respectively.

C. Cryptographic Primitives

We make use of certain cryptographic primitives to construct the proposed decentralized and privacy-preserving vehicle supply chain scheme APPB.

A secure Pseudo-Random Function (PRF) and a secure hash function are utilized to achieve privacy protection and integrity verification, which are both polynomial-time functions. A secure PRF means that any Probabilistic Polynomial-Time (PPT) adversaries cannot distinguish between PRF and random function. A secure hash function is collision-resistant. It takes inputs of variable length and gives a fixed-length compressed output. For any PPT adversaries, they cannot find two different inputs that return the same output. Interested readers can read [17] to find the formal definition of the secure pseudo-random function and secure hash function.

We also use Symmetric-Key Encryption (SKE) to encrypt the sensitive information stored on the blockchain. An SKE consists of the following three functions:

- $K \leftarrow$ KeyGen(λ) is a probabilistic algorithm that receives a secure parameter λ as input, and it gives out a secret key K.
- $c \leftarrow$ Enc(K, m) is a deterministic algorithm which receives K and a message m as input, and it outputs an encrypted ciphertext c.
- $m \leftarrow$ Dec(K, c) is a deterministic algorithm which receives K and a ciphertext c as input, and it outputs a decrypted message m.

We also leverage the group signature [18] for the protection of identity privacy, and achieving traceability simultaneously. A typical group signature GS consists of four algorithms:

- $(gpk, gsk, sk_1, \ldots, sk_n) \leftarrow$ KeyGen(λ, n) is a probabilistic algorithm which receives a security parameter λ, and the number of group members n as input. It outputs a group public key gpk, a group private key gsk owned by the group manager only, and n private keys $sk_1, \ldots, sk_n$ for respective group members.
- $\sigma \leftarrow$ Sign(gpk, sk, m) is a probabilistic algorithm that receives a group public key gpk, a private key sk of a group member, and a message m as input. It outputs a group signature σ.
- $\{true, false\} \leftarrow$ Verify(gpk, σ, m) is a deterministic algorithm which receives a group public key gpk, a group signature σ, and a message m as input. If σ is valid, it will output $true$, otherwise, it will output $false$.
- $A_i \leftarrow$ Open(gpk, gsk, σ_i, m_i) is a deterministic algorithm which receives a public key gpk, a master private key gsk, a signature σ_i, and a message m_i as input. It outputs an indicator A_i, and the group manager can use A_i to find the identity of the signer i.

If adversaries cannot distinguish the signer by the group signature, we can conclude that this kind of group signature scheme is anonymous. Interested readers can read [18] to find the formal definition of anonymity.

By introducing GS, the group manager can trace the signer's identity while others cannot get this sensitive information.

IV. System Design

In this section, we present the system model, threat model as well as design goals of the proposed scheme.

A. System Model

The system comprises six entities, as shown in Fig. 2: Trust Authority (TA), Blockchain Network, Vehicle Part Manufacturer, Vehicle Part Agent, Vehicle Part Retailer, and Customer.

The TA is tasked with identity registration and product tracing. Every manufacturer, agent, and retailer should register with

[5]http://gavwood.com/paper.pdf
[6]https://www.hyperledger.org/use/fabric

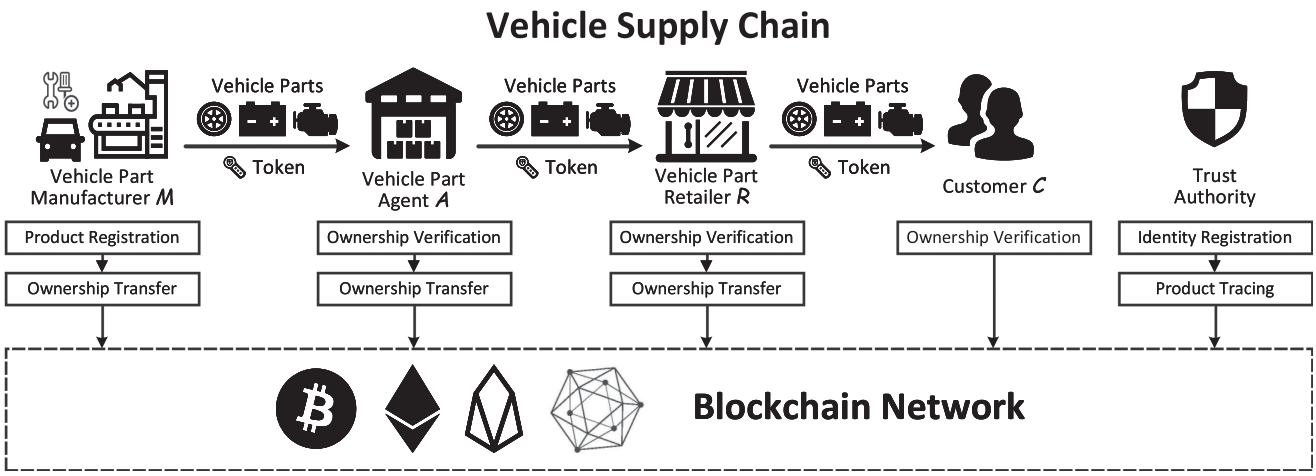

Fig. 2. System Model.

TA with their real identities. Besides, TA is also in charge of product tracing as a supervisor in case of any quality conflicts, which means that the whole vehicle supply chain of a given product identifier *PID* can only be traced by TA. Note that TA remains offline unless some dispute occurs, hence the existence of TA does not affect the property of decentralization. In fact, in typical consortium blockchain platforms such as Fabric, a trusted authority (like *fabric-ca* in Fabric) is needed for peer authentication and setting system parameters recorded in the genesis block.

The Blockchain Network (e.g., Ethereum and Hyperledger Fabric, we use the word *blockchain* for simplicity) can be considered a decentralized and reliable network. The information of the produced vehicle part (also named *products* in the following parts) is stored on the blockchain as unmodifiable credentials. All the manufacturers, agents, and retailers can publish messages (i.e., Product Information (PI), Ownership Transfer Information (OTI), and Proof-of-Ownership (PoO)) stored in blockchain transactions. In addition, everyone can invoke the smart contract to query the PI with a given *PID*. Note that the proposed APPB scheme does not modify the underlying blockchain platform, hence it is platform independent and can be deployed on the existing blockchain platforms such as Ethereum and Hyperledger Fabric. Besides, the detailed setting on the blockchain itself such as the consensus mechanism is out of our consideration because, in APPB, we focus on how to achieve privacy-preserving and anti-counterfeiting supply chains with the help of blockchain, instead of optimizing the blockchain platform itself.

The manufacturers produce the vehicle parts (e.g., tires, batteries, engines, etc.) and give each vehicle part an RFID tag called EPC (Electronic Product Code) that stores a unique product identifier *PID*. Any entity can use an RFID reader to obtain the *PID* associated with each vehicle part. Then they register the vehicle part information by invoking a smart contract. The vehicle parts can be sold to agents, and OTI is published to the blockchain by generating transactions. The transaction search tokens are then given to agents.

There are multi-level agents, especially in large-scale vehicle supply chains. In the following description, we assume that there is only one agent for simplicity. More specifically, a manufacturer first sells the vehicle parts to the agent, and it sells them to retailers, and finally, retailers will sell them to customers. In the proposed system, after receiving the vehicle parts sold by manufacturers, the agent first uses the received search token to find the OTI stored on the blockchain. Then it verifies the PoO and accepts/rejects the products. If the products are accepted, it updates the proof stored on the blockchain for them. When the agent wants to sell these vehicle parts to the retailer, it will publish the OTI to the blockchain and send the search token to the retailer.

The retailers receive both the vehicle parts and the corresponding search token, find the OTI, and verify PoO similar to the agent process. When selling vehicle parts, the retailer gives both the vehicle parts and the PoO to customers.

The customers can buy vehicle parts from retailers and check the product ownership by verifying the PoO to avoid counterfeiting. For a customer, if the verification is successful, he will the product and pays the money for the vehicle part, the product status will also be updated by the smart contract. Otherwise, he will reject the product.

B. Threat Model

In this work, the threat model assumes that TA is *trustworthy*, which means that it uses secure channels to communicate with others, and it cannot be compromised by any kind of attack. Hence, TA cannot leak any information when tracing the product flow of given *PID*s.

Data stored on the blockchain cannot be modified unless the adversaries control the majority power, which is not practical, especially in large-scale blockchain networks. However, anyone can access the data stored on a permission-less blockchain. This kind of transparency allows the external adversaries to perform inference attacks to determine the whole product flow and the business relationship between manufacturers, agents, and retailers in the vehicle supply chain, which can be considered a kind of business secret in some scenarios. Adversaries can perform inference attacks by analyzing the historical product

information (which can be returned by invoking smart contracts) and analyzing transaction data stored on the blockchain.

We assume that the manufacturers are *honest*, which means they do not produce counterfeits or publish fake PI. This assumption is reasonable because such behavior will harm their reputation and profits, especially for large manufacturers.

We also assume that both the agents and the retailers can be seen as internal adversaries because they are *untrusted*. As intermediate sellers, they will procure and sell counterfeit to others to make illegal profits. However, intermediate buyers do not want to buy counterfeits as it will cause economic loss.

For customers, we assume that they are also *honest*. They want to buy genuine vehicle parts instead of counterfeits, and they are not interested in the product flow of the corresponding vehicle parts or the business relationship of those entities.

C. Design Goals

Based on the earlier discussed models, here we define the design goals of our decentralized and privacy-preserving vehicle supply chain management scheme APPB.

1) Privacy Protection: The scheme should protect the privacy of the product selling information and the business relationship among manufacturers, agents, and retailers. As discussed earlier, achieving privacy protection means that adversaries cannot obtain or deduce the above sensitive information by analyzing the data stored on the blockchain.

2) Anti Counterfeiting: The scheme should prevent malicious agents and retailers from selling counterfeits. In other words, each *PID* should associate with a unique PoO. Customers can check the product information that includes the product identifier, manufacturer, PoO, and product status (i.e., whether this vehicle part has been sold to other customers). Customers can also use this information to judge whether this vehicle part is counterfeit. Besides, the scheme can resist the counterfeiting attack that copies the *PID* from a genuine vehicle part's EPC and creates a fake EPC with the same *PID* for several counterfeits.

3) Acceptable Efficiency: The scheme should be efficient at each phase. Due to transaction verification and consensus formation, blockchain solutions suffer from performance issues, which are not present in centralized systems. The objective of this work is not to improve the efficiency of the blockchain itself. Hence, acceptable efficiency means that the additional computation and transmission overhead should be as low as possible.

4) Product Traceability: In case of any disputes among any of the entities of the system, the traceability of the seller, as well as product flow, should be possible. It should be able to identify the source of the problem and assign responsibility. Hence, the scheme should create a balance between traceability and privacy protection. To achieve traceability, a trustworthy supervisor TA should be introduced to trace the product information. However, to guarantee privacy protection, only the TA can trace the products.

V. The Proposed APPB Scheme

In this section, we give detailed information about the proposed APPB scheme.

TABLE II
List of Notations

Notation	Description
λ	Security parameter.
$P = \{PID_1, \cdots, PID_n\}$	Product set with n product identifiers.
PI, PO	Product information set and PoO set.
$\|\|$	String concatenation operator.
K_M	*TA*'s master secret key.
$U = \{u_1, \cdots, u_m\}$	Register information set with m entities.
ID_i	Entity i's pseudo-ID.
K_i	Entity i's secret key.
F	A secure pseudo-random function.
H	A secure hash function.
GS	The group signature scheme [18].
gpk	Group public key for group signature.
gsk	Master private key for group signature.
sk	Private key for group signature.
σ	Group signature.
s_c^a, s_c^b	The state information in the c^{th} ownership transfer/verification operation.
$\mathcal{T}$	Transaction search token.
$\mathcal{H} = \{h_1, \cdots, h_p\}$	One-way hash chain with p hash values.
$addr$	Address of the deployed smart contract.

A. Overview

The APPB scheme consists of five processes: *System Initialization*, *Product Registration*, *Ownership Transformation*, *Product Information Query*, and *Ownership Verification*.

In the proposed scheme, we use a secure pseudo-random function F and a secure hash function H modeled as random oracles, as is illustrated in (1):

$$F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda \tag{1a}$$

$$H : \{0,1\}^* \to \{0,1\}^\lambda \tag{1b}$$

where λ is the security parameter.

Moreover, one-way hash chains are used to realize identification and one-time key generation. More specifically a one-way hash chain $\mathcal{H} = \{h_1, \ldots, h_p\}$ is constructed by (2):

$$h_i = \begin{cases} \{0,1\}^\lambda, & i = p \\ H(h_{i+1}), & 0 \le i < p \end{cases} \tag{2}$$

where λ is the security parameter. The term *hash chain* is used to represent the one-way hash chain for simplicity.

Note that anyone with the knowledge of h_{i+1} can calculate $h_i = H(h_{i+1})$, however, h_i cannot be computed without knowing h_{i+1}. Hence, the hash chain can be used to realize one-time credentials for authentication. Assume `Alice` pre-generates a hash chain $\mathcal{H}_{ab} = \{h_1, \ldots, h_p\}$ to authenticate with `Bob`. In the i^{th} authentication process, `Alice` sends h_i to `Bob`. After receiving h_i, `Bob` verifies `Alice`'s credential by checking whether $H(h_i) = h_{i-1}$ holds or not, where h_{i-1} is the credential sent by `Alice` in $(i-1)^{th}$ authentication process. If the verification is successful, `Bob` accepts `Alice`'s authentication.

We instantiate a simple vehicle supply chain as shown in Fig. 2. This vehicle supply chain includes four entities: vehicle part manufacturer $\mathcal{M}$, agent $\mathcal{A}$, retailer $\mathcal{R}$, and customer $\mathcal{C}$. These symbols are also used in later sections to represent the scheme. Table II lists the symbols used in this work.

Algorithm 1: Init.

Input: A security parameter λ, and a register information set $U = \{u_1, \ldots, u_m\}$.
Output: A register result set $R = \{r_1, \ldots, r_m\}$, a group public key gpk, and a group private key gsk.

1: init R.
2: set $K_M \leftarrow \{0,1\}^{\lambda}$.
3: compute $(gpk, gsk, sk_1, \ldots, sk_m) =$ GS.KeyGen (λ, m).
4: **for** $i = 1$ to m **do**
5: compute $ID_i = F(K_M, u_i)$.
6: set $K_i \leftarrow \{0,1\}^{\lambda}$.
7: set $r_i = (ID_i, u_i, sk_i, K_i)$.
8: add r_i to R.
9: **end for**
10: return R, gpk and gsk.

Algorithm 2: ProductReg.

Input: A set of vehicle parts $P = \{PID_1, \ldots, PID_n\}$, a security parameter λ, a secret key K for PoO generation, a group public key gpk, and a private key sk.
Output: A PoO set $PO = \{PO_1, \ldots, PO_n\}$.

1: init PI, PO.
2: **for each** $PID_i \in P$ **do**
3: compute $p_i = F(K, ID||PID_i)$.
4: compute $h_i = H(p_i)$.
5: set $\sigma_i =$ GS.Sign(gpk, sk, h_i).
6: add $(PID_i, h_i, ID, false, \sigma_i)$ to PI.
7: add (PID_i, p_i) to PO.
8: **end for**
9: perform $Invoke(*, addr, Reg, PI)$.
10: return PO.

B. Definition of Smart Contract

We define a smart contract *APPBContract* with contract address $addr$, that includes three following functions used in processes of product registration, product information query, and product information update.

- $\perp \leftarrow Reg(PI)$: The contract function for product registration that takes a product information set PI as input. Only $\mathcal{M}$ can invoke it.
- $R \leftarrow Query(P)$: The contract function for product information query that takes a product set P as input, and it outputs the query result R that includes the product information for each $PID \in P$. Everyone can invoke it to search the product information of a given P.
- $\perp \leftarrow Update(S)$: The contract function for product information update that takes an update set S as input. Each $(PID, k, v) \in S$ includes a new value v associated with key $k \in \{ProofHash, Status, Sig\}$ in product PID's information. Only authorized $\mathcal{A}$ and $\mathcal{R}$ can invoke this function to update the PoO or product status.

C. System Initialization

In this process, TA first executes *Init* algorithm to register for each entity, generates the public parameter, and sends them to each entity in secure channels. Then TA deploys the smart contract mentioned earlier by using standard blockchain deployment tools. After system initialization, the TA will be offline unless some disputes happen.

The *Init* shown in Algorithm 1 works as follows: First, TA initializes a set R that will be used to store the registration result. Each $u_i \in U$ includes entity i's real-world identity id_i and entity type $type_i \in \{manufacturer, agent, retailer\}$. In the scenario mentioned above, U includes the identity information of $\mathcal{M}$, $\mathcal{A}$ and $\mathcal{R}$. Next, TA generates the master secret key K_M, the group public key gpk, the group private key gsk, and m private keys $\{sk_1, \ldots, sk_m\}$ for m entities.

For each entity i, TA generates a registered information r_i which includes i's pseudo-ID ID_i, the real-world identity u_i, the secret key K_i for PoO generation, and the private key sk_i for group signature. Then TA adds r_i to R. Finally, TA outputs a set of m register results and a group public key gpk as the public parameter. The gsk is owned by TA to realize the product tracing. Besides, $\mathcal{M}$, $\mathcal{A}$, and $\mathcal{R}$ need to generate a blockchain address and send it to TA, we omit the detailed description.

Product Information

ProductID: Product identifier.
ProofHash: The PoO hash of current product owner.
Manufacturer: The manufacturer of this product.
Status: Whether this product has been sold to customer.
Sig: The signature of the current owner.

Fig. 3. Product Information.

D. Product Registration

In this process, $\mathcal{M}$ executes *ProductReg* algorithm to register for a batch of vehicle parts $P = \{PID_1, \ldots, PID_n\}$, and publishes the product information set $PI = \{PI_1, \ldots, PI_n\}$ to the blockchain by invoking the smart contract.

The *ProductReg* shown in Algorithm 2 works as follows: First, it initializes two empty sets PI and PO. Then for each $PID_i \in P$, it computes the ownership proof p_i, the hash h_i which represent the PoO hash of vehicle part with PID_i, and a group signature σ_i. Then it adds $(PID_i, h_i, ID, false, \sigma_i)$ to PI, and adds (PID_i, p_i) to PO, where ID is the manufacturer of this vehicle part, and $false$ means this vehicle part has not been sold to any customers.

Finally, it invokes the smart contract to store PI on the blockchain and returns PO as the output. The structure of PI that is stored in the blockchain is shown in Fig. 3.

E. Ownership Transformation

In this process, when the current owner ID_a (which can be $\mathcal{M}$ or $\mathcal{A}$) sells a batch of vehicle parts $P = \{PID_1, \ldots, PID_n\}$ to a new owner ID_b (which can be agent $\mathcal{A}$ or retailer $\mathcal{R}$), it executes

Algorithm 3: OwnerTrans.

Input: A PoO set PO, a hash chain $\mathcal{H} = \{h_1, \ldots, h_p\}$, and the state information s_c^a in the cth ownership transfer operation.
Output: A transaction search token $\mathcal{T}$, and the updated state s_{c+1} used for the $(c+1)$th ownership transfer operation.

1: parse s_c^a as (c, β_c^a).
2: set $\beta_{c+1}^a = h_{c+1}$.
3: set $T_{id} = SendTx(*, \text{SKE.Enc}(\beta_{c+1}^a, PO))$.
4: set $\mathcal{T} = \text{SKE.Enc}(\beta_c^a, T_{id}||\beta_{c+1}^a)$.
5: set $s_{c+1}^a = (c+1, \beta_{c+1}^a)$.
6: return $\mathcal{T}$, s_{c+1}^a.

OwnerTrans algorithm to transfer the ownership from ID_a to ID_b. The ownership transfer information will be encrypted and stored as a blockchain transaction with the transaction identifier T_{id} to achieve privacy-preserving and reliable ownership transfer.

The *OwnerTrans* shown in Algorithm 3 works as follows: Assume that the ownership is transferred from ID_a to ID_b, and the state $s_c^a = (c, \beta_c^a)$ which contains a counter c and a hash value β_c^a in the hash chain $\mathcal{H}$ (which is constructed by (2)). We assume that before the first ownership transfer operation, ID_a needs to send the first hash h_1 in $\mathcal{H}$ to ID_b through off-chain channels. When $c = 1$, the initial state between ID_a and ID_b is $s_1^a = (1, \beta_1^a)$ where $\beta_1^a = h_1$. Note that ID_a uses $\mathcal{H}$ to authenticate with ID_b only, if ID_a wants to authenticate with another entity, he must use a different hash chain.

Next, the algorithm gets β_{c+1}^a from ID_a's pre-constructed hash chain $\mathcal{H}$, and sends a blockchain transaction to store encrypted PO. Finally, it generates a transaction search token $\mathcal{T}$ and updates the state with s_{c+1}^a as output, then sends $\mathcal{T}$ to ID_b by off-chain channels.

It is important to note that in Algorithm 3, the first three parameters of $SendTx$ are $*$, which means that the $addr_{in}$, $addr_{out}$ and $amount$ can be any legal value. In other words, we do not need to designate $addr_{in}$ to ID_a's blockchain address and designate $addr_{out}$ to ID_b's blockchain address like other blockchain-based schemes do, which gives APPB the ability to resist the inference attack on the blockchain, where the attacker can infer the business relationship based on the blockchain transaction relationship.

F. Ownership Verification

When querying the product information, $\mathcal{A}$, $\mathcal{R}$ and $\mathcal{C}$ can verify the PoO to judge whether the received vehicle parts are counterfeits. If the verification is successful, the vehicle parts will be acceptable because they are genuine, otherwise, they will be rejected. The process for *Agents & Retailers* and *Customers* are different, hence we separately introduce each one.

1) Agents & Retailers: For $\mathcal{A}$ or $\mathcal{R}$, they execute *OwnerVerA* algorithm to verify the PoO for the current owner. For each vehicle part *PID*, if the verification is successful, they update the product information and accept *PID*, otherwise, *PID* will be rejected.

Algorithm 4: OwnerVerA.

Input: A transaction search token $\mathcal{T}$, receiver ID_b's secret key K_b for PoO generation, a group public key gpk, a private key sk, and the state information s_c^b.
Output: An accepted PoO set PO', and an updated state s_{c+1}^b used for the $(c+1)$th ownership verification.

1: init tmp, PO'.
2: parse s_c^b as (c, β_c^b).
3: set $\mathcal{T}_{dec} = \text{SKE.Dec}(\beta_c, \mathcal{T})$.
4: parse $\mathcal{T}_{dec}$ as $T_{id}||\beta_{c+1}$.
5: **if** $H(\beta_{c+1}) \neq \beta_c^b$ **then**
6: return $\perp$, s_c.
7: **end if**
8: set $\beta_{c+1}^b = \beta_{c+1}$.
9: parse $data$ stored on transaction T_{id}.
10: set $PO = \text{SKE.Dec}(\beta_{c+1}^b, data)$.
11: set $tmp = \{PID|(PID, p) \in PO\}$.
12: set $PI = PIQuery(tmp)$.
13: **for each** $PO_i \in PO$ **do**
14: parse PO_i as (PID_i, p_i).
15: parse PI_i as (PID, h, mf, $sold$, σ).
16: compute $h' = H(p_i)$.
17: **if** $h' == h$ and $sold == false$ **then**
18: **if** GS.Verify$(gpk, \sigma, h) == true$ **then**
19: compute $p_i' = F(K_b, ID_b||PID_i)$.
20: compute $h_i' = H(p_i')$.
21: set $\sigma_i = \text{GS.Sign}(gpk, sk, h_i')$.
22: add (PID_i, p_i') to PO'.
23: add (PID, $ProofHash$, h_i') to S.
24: add (PID, Sig, σ_i) to S.
25: **end if**
26: **end if**
27: **end for**
28: perform $Invoke(*, addr, Update, S)$.
29: set $s_{c+1}^b = (c+1, \beta_{c+1}^b)$.
30: return PO', s_{c+1}^b.

The *OwnerVerA* shown in Algorithm 4 works as follows: Assume ID_b receives the token $\mathcal{T}$ from ID_a, the algorithm first initializes two sets tmp and PO'. Then it parses the state information s_c^b. Note that as mentioned in Section V-E, when $c = 1$, $s_1^b = (1, \beta_1^b)$ where $\beta_1^b = h_1$ (h_1 is the shared hash by ID_a through off-chain channels). Next, it decrypts the token and parses the decrypted result to get the transaction id T_{id} and new one-time key β_{c+1} which can be used for authentication because only the owner of sk knows β_{c+1}. If $H(\beta_{c+1}) \neq \beta_c^b$ which means β_{c+1} is illegal, all the products will be rejected. If the verification is successful, it will extract PO. Next, it finds the product information *PI* associated with those PIDs in PO from the blockchain by executing *PIQuery* (which will be introduced in the next section).

Then for each $PO_i \in PO$, the algorithm verifies the PoO and generates a new PoO as given in lines 12–26. First it parses PO_i

Algorithm 5: OwnerVerC.

Input: A PoO PO, and a group publik key gpk.
Output: $accept$ or $reject$.
1: init S.
2: parse PO as (*PID*, p).
3: set $PI = PIQuery(PID)$.
4: parse PI as (*PID*, $h, mf, sold, \sigma$).
5: compute $h' = H(p_i)$.
6: **if** $h' \neq h$ or $sold == true$ **then**
7: return $reject$.
8: **end if**
9: **if** GS.Verify$(gpk, \sigma, h) == false$ **then**
10: return $reject$.
11: **end if**
12: $\mathcal{R}$ adds (*PID*, $status, true$) to S.
13: $\mathcal{R}$ performs $Invoke(*, addr, Update, S)$.
14: return $accept$.

and PI_i, then it computes the hash h' of proof p_i. If $h' == h$ and $sold == false$, which means the proof verification is successful, and this vehicle part has not been sold to any customers, it will verify the group signature σ signed by the current owner. If the verification is successful, ID_a will accept this product. Then the algorithm generates the new PoO p'_i, new PoO hash h'_i, and new signature σ_i signed by the receiver's private key sk. Next, it adds (PID_i, p'_i) to PO', and adds both the proof update information and signature update information to S. Finally, it updates proof information by invoking the smart contract, returns PO' and s^b_{c+1}.

Note that during the cth ownership transfer/verification round, $s^a_c = s^b_c$ holds because:

- When $c = 1$, $s^a_1 = (1, \beta^a_1)$ and $s^b_1 = (1, \beta^b_1)$. Because h_1 is the shared hash between ID_a and ID_b, $\beta^a_1 = \beta^b_1 = h_1$ holds. Hence, when $c = 1$, $s^a_1 = s^a_1$ holds.
- Assume when $c = k$, $s^a_k = s^b_k$ holds (i.e., $\beta^a_k = \beta^b_k$ holds), when $c = k + 1$, as shown in Algorithm 3, $s^a_{k+1} = (k + 1, \beta^a_{k+1})$, and $\beta^a_{k+1} = h_{k+1}$ that satisfies $\beta^a_k = H(\beta^a_{k+1})$. Besides, as shown in Algorithm 4, $s^b_{k+1} = (k + 1, \beta^b_{k+1})$ only if $\beta^b_k = H(\beta^b_{k+1})$ holds. Based on the assumption when $c = k$, $H(\beta^a_{k+1}) = H(\beta^b_{k+1})$ holds. Because of the collision resistance property of the hash function H, the equation $H(\beta^a_{k+1}) = H(\beta^b_{k+1})$ holds only if $\beta^a_{k+1} = \beta^b_{k+1}$ holds. Hence, when $c = k + 1$, $s^a_c = s^b_c$ holds.

2) Customers: For the customer $\mathcal{C}$, he executes *OwnerVerC* algorithm to verify the vehicle part *PID*'s PoO, which is given by $\mathcal{R}$. If the verification is successful, he will accept the product.

The *OwnerVerC* shown in Algorithm 5 works as follows: First it finds the product information PI by executing *PIQuery*, then it verifies the PoO PO given by $\mathcal{R}$. If $h' \neq h$ or $sold == true$, which means that the proof verification is failed or this vehicle part has been sold to other customers, $\mathcal{C}$ will reject the product. In addition, if the signature verification is failed, $\mathcal{C}$ will also reject the product. Otherwise, $\mathcal{C}$ will supervise the retailer to invoke the smart contract for updating *PID*'s sale status and accept the product.

Algorithm 6: PIQuery.

Input: A product set P.
Output: Queried product information set PI.
1: set $PI = Invoke(*, addr, Query, P)$.
2: return PI.

G. Product Information Query

In this process, when TA, $\mathcal{A}$, $\mathcal{R}$ and $\mathcal{C}$ want to query the product information by its identifier *PID*, they execute *PIQuery* algorithm to get the detailed product information stored in blockchain.

The *PIQuery* shown in Algorithm 6 works as follows. Given a product set P that includes some PIDs that will be queried, it first invokes the smart contract to query the corresponding product information for each vehicle part. Finally, it returns the query result PI as the output. Everyone can get the queried product information when knowing the *PID*, which makes $\mathcal{A}$, $\mathcal{R}$ and $\mathcal{C}$ use their blockchain address to get the PoO hash, and recognize counterfeits easily & transparently without knowing the whole product flow.

VI. Security Analysis

To prove the security capabilities of APPB, we present detailed security analyses in this section. First, we prove that APPB can protect the privacy of sensitive information while realizing traceability by TA. Second, we prove that APPB also achieves anti-counterfeiting and can resist counterfeiting attacks on vehicle supply chains.

A. Privacy Protection

In traditional blockchain-based supply chains, the detailed product ownership information is stored on the blockchain as plaintexts to enable product tracing. Smart contracts are utilized in this process, that is, anyone can invoke the smart contract to trace the product flow of every produced vehicular part *PID*. However, it brings privacy issues because adversaries can perform inference attacks to deduce sensitive information stored on the public blockchain. In the following sections, we analyze these privacy issues from two different perspectives: *Smart Contract* and *Blockchain Transactions*, followed by a detailed analysis of the proposed scheme.

1) Perspective of Smart Contract: Consider a scenario where an external adversary `Adv` wants to obtain a specific product flow by invoking a product information query contract. Assume the smart contract returns $\mathcal{M} \rightarrow \mathcal{A} \rightarrow \mathcal{R}$, then `Adv` can easily infer that their exists a vehicle supply chain which consist of at least three entities: $\mathcal{M}$, $\mathcal{A}$ and $\mathcal{R}$. Besides, if `Adv` can obtain historical transactions of product flow, he can recover intricate details of the product selling information and the business relationship of each entity, which can be seen as business secrets. It is easy for `Adv` to perform these attacks, as access to data is easy in a blockchain system where the stored data are published to everyone.

In APPB, everyone can query the product information by invoking function $Query$ in smart contract *APPBContract*. However, the result cannot leak any information about the product flow. As shown in Fig. 3, Adv only knows the fields of "ProofHash" (i.e., the hash of PoO), "Manufacturer," "Status" (i.e., whether this vehicle part has been sold), and "Sig" (i.e., the group signature). Assume Adv get all the historical product information during the *ProofUpdate* and *StatusUpdate* process. We can conclude that: if the pseudo-random function F and the hash function H are secure, and the group signature scheme GS is anonymous, the adversary Adv cannot perform successful inference attacks to infer the product flow by invoking the *Query* contract and analyzing the historical product information.

Below we give a sketch proof: In APPB, the product information update includes the proof update and status update. The "Status" field cannot be utilized by Adv to infer the product flow, so we focus on the update of "ProofHash" and "Sig" fields. Given a vehicle part *PID*, its historical "ProofHash" are $\{h_1, \ldots, h_t\}$, each hash h is generated by $h = H(p)$, where p represents the PoO, and $p = F(K, ID||PID)$. If the hash function H is secure, Adv cannot recover the proof p by its hash value h. Similarly, if the PRF F is secure, Adv cannot infer the current owner *ID* without knowing the secret key K owned by *ID*. Moreover, if GS is anonymous, the adversary Adv cannot infer the identity from the group signature. Hence, in APPB, querying the product information by invoking a smart contract cannot leak any information related to the product flow.

2) Perspective of Blockchain Transactions: Transactions stored on blockchain are immutable and public, which enables both transparency and vulnerability because simply storing data on blockchain jeopardizes data privacy. Hence, data encryption can be considered a straightforward way to achieve confidentiality and protect data privacy.

In APPB, all the OTI stored on the blockchain are encrypted by a one-time secret key generated by a secure hash function H. Assume that a seller $\mathcal{S}$ sends encrypted OTI stored on transaction T_{id} to a buyer $\mathcal{B}$, and the one-time secret key is β_c generated by $\mathcal{S}$. If the SKE scheme is secure, and both $\mathcal{S}$ and $\mathcal{B}$ do not leak β_c intentionally, adversary Adv cannot obtain the OTI. Even if β_c is compromised, without knowing the transaction ID T_{id}, Adv has to filter the designated transaction from massive transactions stored on the blockchain by checking whether the transaction data can be decrypted, which is not practical. Moreover, β_c is a one-time secret key, and the key used for future encryption $\beta_{c+1} = H^{-1}(\beta_c)$ cannot be compromised because of the one-way property of the secure hash function H.

If an adversary Adv wants to get sensitive information by analyzing the transaction data, he needs to get all the secret keys and transaction IDs used in the whole process. Hence, the probability of performing this attack successfully is negligible. In essence, the encrypted data stored in massive blockchain transactions cannot be used to infer the product selling information. Besides, during the PoO transfer process, the transaction is not sent from the buyer's address to the seller's address like traditional solutions, hence the business relationship between them does not be leaked. Therefore, adversaries cannot perform successful inference attacks to get sensitive information by analyzing the transaction data or transaction relationship stored on the blockchain, respectively.

3) Traceability: When some disputes (e.g., product quality problems) occur, TA will provide the proof information to realize the non-repudiation of malicious sellers. In addition, TA can also trace the product flow of a given product identifier. Assume the vehicle part *PID* sold by retailer $\mathcal{R}$ to the customer $\mathcal{C}$ has quality problems, $\mathcal{C}$ first tell TA the product identifier *PID*. Then TA gets the corresponding product information by invoking *PIQuery* algorithm to get the signature stored on the blockchain, and executes GS.Open(gpk, gsk, σ, h) to find the signer's identity. Because the signature signed by $\mathcal{R}$ has been published to the blockchain, it cannot be modified or deleted. Moreover, TA can also trace the historical signature related to *PID* by synchronizing the blockchain data periodically and getting the complete product flow.

B. Anti-Counterfeiting

In this section, we will prove that APPB achieves anti-counterfeiting. First, we assume the following scenario: the new product owner (i.e., the buyer) $\mathcal{B}$ wants to check the genuineness of a given vehicle part received from its current owner (i.e., the seller) $\mathcal{S}$. The following five possible situations that may lead to counterfeit selling from $\mathcal{S}$ to $\mathcal{B}$.

- **Situation 1:** $\mathcal{S}$ sells a counterfeit with a fake PID.
- **Situation 2:** $\mathcal{S}$ has never owned the PoO of a genuine vehicle part PID and sells a counterfeit.
- **Situation 3:** $\mathcal{S}$ has the historical PoO of a genuine vehicle part PID and sells a counterfeit.
- **Situation 4:** $\mathcal{S}$ has the current PoO of a genuine vehicle part PID that has been sold to customers and sells a counterfeit.
- **Situation 5:** $\mathcal{S}$ has the current PoO of a genuine vehicle part PID that has not been sold to customers and sells a counterfeit.

In the following section, we analyze these situations separately to prove that the proposed scheme APPB is anti-counterfeiting.

Situation 1: In this situation, $\mathcal{S}$ forges a fake PID and PoO. When receiving the counterfeit, $\mathcal{B}$ first query the product information with PID in *OwnerVerA* or *OwnerVerC* algorithm. In APPB, only the manufacturer can register the vehicle part with its unique PID. Hence, if $\mathcal{S}$ sells a counterfeit with a fake PID, the verification will fail because querying an unregistered vehicle part by *PIQuery* will return $\perp$. In this situation, $\mathcal{B}$ will reject this product as a counterfeit.

Situation 2: In this situation, $\mathcal{S}$ uses a genuine PID which can be acquired by RFID cloning from a genuine vehicle part's EPC and forges a fake PoO. When receiving the counterfeit, like **Situation 1**, $\mathcal{B}$ first queries the product information, then verifies the PoO. Because the PID is associated with a genuine vehicle part, the query result can be found. However, without knowing the current PoO, the hash of a fake PoO cannot be equal to the corresponding "ProofHash" field stored on the blockchain.

Hence, the PoO hash verification will fail, which means $\mathcal{B}$ will reject this product as a counterfeit.

Situation 3: In this situation, $\mathcal{S}$ has the historical PoO of the vehicle part. However, after the owner's transfer, his PoO is obsolete. Hence, like **Situation 2**, the PoO hash verification will also fail because $\mathcal{S}$ does not have the current PoO of this vehicle part. Thus, $\mathcal{B}$ will reject this product as a counterfeit.

Situation 4: Unlike **Situation 2** and **Situation 3**, in this situation, the PoO hash verification will be successful because $\mathcal{S}$ has the current PoO of the vehicle part. However, this vehicle part has been sold, and this status can be found by querying the product information because the "Status" field is $true$. When $\mathcal{B}$ finds the vehicle part sold by $\mathcal{S}$ has been sold to another buyer, it is obvious that the vehicle part is not genuine. Hence, $\mathcal{B}$ rejects this product as a counterfeit.

Situation 5: In this situation, $\mathcal{S}$ possesses both the genuine vehicle part PID and its current PoO. If $\mathcal{S}$ sells a counterfeit to $\mathcal{B}$, the product verification will be successful. In other words, $\mathcal{B}$ will accept this counterfeit. However, once $\mathcal{B}$ accepts this product, if $\mathcal{B}$ is an agent or a retailer, he will update the PoO information; if $\mathcal{B}$ is a customer, he will supervise $\mathcal{S}$ to update the sale status. After the update of PoO, $\mathcal{B}$ turns into the current owner of PID, and $\mathcal{S}$'s PoO becomes obsolete. After the update of the sale status, this vehicle part cannot be sold to other customers. As is described in **Situation 3** and **Situation 4**, both the counterfeit and his genuine vehicle part associated with PID cannot be sold to others. It is straightforward that the profit of selling a counterfeit is much lower than the loss caused by not being able to sell the corresponding genuine vehicle part. Hence, there are no economic benefits for the genuine vehicle part owner $\mathcal{S}$ to sell a counterfeit instead of selling the genuine vehicle part.

Based on the detailed analyses above, in APPB, $\mathcal{B}$ can easily distinguish the counterfeit by querying the PID's product information stored on the blockchain and verifying $\mathcal{S}$'s Proof-of-Ownership. Hence, APPB achieves privacy-preserving anti-counterfeiting on blockchain-based vehicle supply chains.

VII. Experimental Evaluation

To evaluate the performance, we use two blockchain platforms and make a quantitative analysis to measure the efficiency of APPB. These experimental results indicate that the proposed APPB scheme has acceptable efficiency.

A. Experimental Setup

We implement the proposed APPB scheme, where programs are written in Go and compiled with Go 1.14.15 x64 environment. We set the security parameter $\lambda = 256$, the number of hash values in a hash chain $p = 100$, and use HMAC-SHA256 and SHA256 to instantiate the pseudo-random function F and the hash function H, respectively.

We choose Ethereum and Hyperledger Fabric as the underlying blockchain platforms for our evaluation, which support smart contracts (named *chaincode* in Fabric). The Ethereum-based and Fabric-based APPB instantiations are named "APPB-Ethereum" and "APPB-Fabric," respectively. We simulate a simple Fabric blockchain network consisting of four peers, each of them running on a virtualized server with 2 GB RAM, Intel i5-7300 U CPU, and 64-bit Ubuntu 16.04 operating system. We also deploy a private Ethereum blockchain with a single node running Go Ethereum[7] client on a virtualized server with 2 GB RAM, Intel i5-7300 U CPU, and 64-bit Windows 10 operating system.

To compare the efficiency with existing blockchain-based anti-counterfeiting supply chains, we choose scheme [10] because it is the most relevant to the proposed scheme APPB. Besides, it also supports public verification for customers with traceability. To realize the comparability between these two schemes, we implement a modified version of [10], named the "Naive" scheme, and deployed it to the Ethereum blockchain platform.

To evaluate the efficiency of APPB, we set 10 different types of products per batch, where the number of products per batch ranges from 100 to 1000. For example, 100 products per batch mean that there are 100 PIDs to register and 100 PoOs to prove and transfer.

B. Performance Evaluation and Analysis

We simulated the complete vehicle part selling process as shown in Fig. 2, from the vehicle part manufacturer $\mathcal{M}$ to the agent $\mathcal{A}$, the retailer $\mathcal{R}$, and finally, the customer $\mathcal{C}$. For each product number, we simulate this process 5 times and calculate the average time for each algorithm. As can be seen in Fig. 4, the x-axis represents the number of products per batch, and the y-axis represents the average execution time for each process. Even for the most time-consuming process, the time cost is under 1200 ms when the number of products is 1000.

In the following parts, we will make a thorough analysis of each process. In the *System Initialization* process, TA only registers the entities in the system, which is not related to the number of products. Hence, as can be seen in Fig. 4(a), the execution time will not be affected by the increase in the number of products. Besides, we note that the execution time of "APPB-Fabric" is nearly that of the "Naive" scheme on Ethereum, and much lower than that of "APPB-Ethereum". In the "Naive" scheme, the system initialization does not need to generate the pseudonym and the private key for the group signature. Hence, the execution time is the lowest in our evaluation, whereas it does not support anonymity.

In the *Product Registration* process, $\mathcal{M}$ needs to sign the group signature for each product. Assume the time of signing/verifying a group signature is t_{sign}/t_{ver}, respectively, and the time of invoking a contract is t_{invoke}. Hence, it takes about $n \cdot t_{sign} + t_{invoke}$ in this process when the number of products is n, the corresponding time complexity is $O(n)$. The time cost of other operations such as calculating hash values is negligible, hence we omit them. As shown in Fig. 4(b), as the product number increases, the average execution time will also increase. Because of the low throughput of Ethereum, even the "Naive" scheme has a longer execution time than that of the privacy-preserving "APPB-Fabric" scheme.

[7]https://geth.ethereum.org

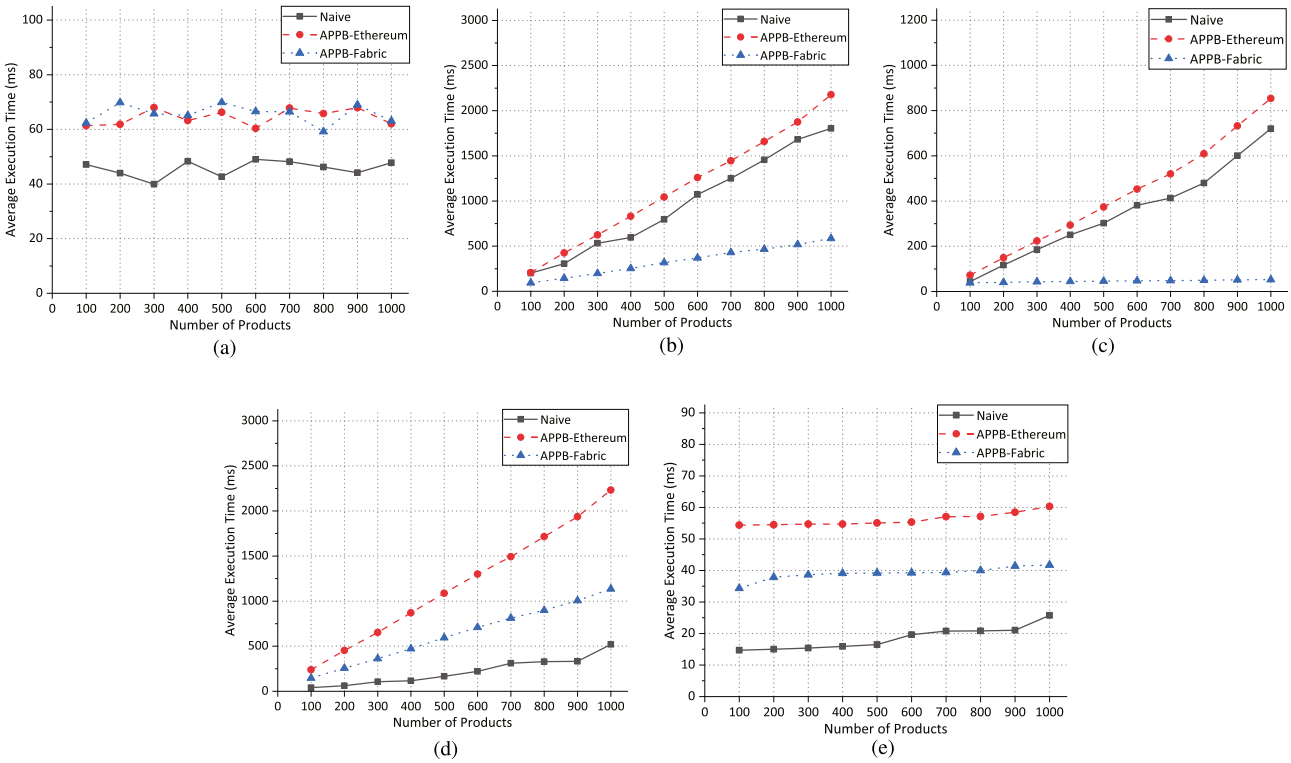

Fig. 4. Computation Cost for each Process in APPB. (a) Average execution time in the system initialization process. (b) Average execution time in the product registration process. (c) Average execution time in the ownership transformation process. (d) Average execution time in the ownership verification process by the agent and retailer. (e) Average execution time in the ownership verification process by the customer.

In the *Ownership Transformation* process, only one symmetric encryption operation is performed, of which it takes about t_{enc} in this process, and the corresponding time complexity is $O(1)$. As is illustrated in Fig. 4(c), as the product number increases, the execution time will increase. Note that the execution time incremental of the "Naive" and "APPB-Ethereum" scheme is much larger than that of "APPB-Fabric". Because of the `Gas` mechanism of Ethereum, the product information should be split into more transactions, otherwise, the `Gas` used will exceed the `GasLimit` (interested readers can refer to the Ethereum Yellow Paper[8] to learn more about the Ethereum `Gas` mechanism), which means that with the number of products increases, the number of the transaction need to be sent will also increase.

In the *Agent & Retailer Verification* process, for each product, the $\mathcal{A}$ or $\mathcal{R}$ first verifies the current signature and then generates a new signature. Hence, the time cost of this process is $n \cdot (t_{ver} + t_{sign}) + t_{invoke}$, and the corresponding time complexity is $O(n)$. In Fig. 4(d), as the product number increases, the average execution time will also increase. The execution time of the "Naive" scheme is lower than that of both two APPB instantiations because it does not need to verify the group signature.

In the *Customer Verification* process, only one contract invoke operation is executed, of which the time cost is t_{invoke}. Hence, the time complexity of this process can be considered as $O(1)$. As can be seen in Fig. 4(e), compared to the "Naive" scheme where the group signature verification operation is not executed, the additional time cost of APPB on the customer side is just under 40 ms even if choosing Ethereum as the underlying blockchain platform.

Based on the above analyses, we can conclude that the proposed APPB scheme has acceptable efficiency when choosing Fabric as the underlying blockchain platform, which means it can be used in real-world vehicle supply chain scenarios.

VIII. Conclusion

In this work, we propose APPB, a reliable and anti-counterfeiting vehicle supply chain scheme with privacy protection. In APPB, the proof of ownership can be guaranteed by the immutability of blockchain and secure cryptographic primitives. As a supervisor, TA can trace the product flow when a counterfeit issue occurs, which is realized by the traceability of the group signature. However, adversaries cannot infer the selling information of vehicular parts and the business relationship stored on blockchain by using inference attacks. Detailed security analysis and experimental evaluations prove that APPB has acceptable efficiency and achieves anti-counterfeiting with privacy protection on vehicle supply chains. In future works, we will consider more privacy issues in blockchain-based vehicle supply chains and propose more privacy-preserving techniques that can be used in real-world vehicle supply chain applications.

[8]https://ethereum.github.io/yellowpaper/paper.pdf

REFERENCES

[1] K. Schwab, "Deep shift: Technology tipping points and societal impact," *KSII Trans. Internet Inf. Syst.*, vol. 10, pp. 1311–1320, 2015.

[2] T. Staake, F. Thiesse, and E. Fleisch, "Extending the EPC network: The potential of RFID in anti-counterfeiting," in *Proc. ACM Symp. Appl. Comput.*, Santa Fe, New Mexico, USA, Mar. 2005, pp. 1607–1612.

[3] D. Zanetti, S. Capkun, and A. Juels, "Tailing RFID tags for clone detection," in *Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp.*, San Diego, California, USA, Feb. 2013, pp. 1–17.

[4] J. Shi, S. M. Kywe, and Y. Li, "Batch clone detection in RFID-enabled supply chain," in *Proc. IEEE Int. Conf. RFID*, Orlando, FL, USA, 2014, pp. 118–125.

[5] G. Khalil, R. Doss, and M. U. Chowdhury, "A comparison survey study on RFID based anti-counterfeiting systems," *J. Sens. Actuator Netw.*, vol. 8, no. 3, p. 37, pp. 1–15, 2019.

[6] G. Khalil, R. Doss, and M. U. Chowdhury, "A novel RFID-based anti-counterfeiting scheme for retail environments," *IEEE Access*, vol. 8, pp. 47952–47962, 2020.

[7] B. Tan, J. Yan, S. Chen, and X. Liu, "The impact of blockchain on food supply chain: The case of Walmart," in *Proc. Smart Blockchain - 1st Int. Conf.*, 2018, pp. 167–177.

[8] S. E. Chang and Y. Chen, "When blockchain meets supply chain: A systematic literature review on current development and potential applications," *IEEE Access*, vol. 8, pp. 62478–62494, 2020.

[9] P. Gonczol, P. Katsikouli, L. Herskind, and N. Dragoni, "Blockchain implementations and use cases for supply chains-a survey," *IEEE Access*, vol. 8, pp. 11856–11871, 2020.

[10] K. Toyoda, P. T. Mathiopoulos, I. Sasase, and T. Ohtsuki, "A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain," *IEEE Access*, vol. 5, pp. 17465–17477, 2017.

[11] S. Jangirala, A. K. Das, and A. V. Vasilakos, "Designing secure lightweight blockchain-enabled RFID-based authentication protocol for supply chains in 5G mobile edge computing environment," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7081–7093, Nov. 2020.

[12] N. Alzahrani and N. Bulusu, "Block-supply chain: A new anti-counterfeiting supply chain using NFC and blockchain," in *Proc. 1st Workshop Cryptocurrencies Blockchains Distrib. Syst.*, Munich, Germany, 2018, pp. 30–35.

[13] N. C. K. Yiu, "Toward blockchain-enabled supply chain anti-counterfeiting and traceability," 2021. [Online]. Available: https://arxiv.org/abs/2102.00459

[14] S. K. Dwivedi, R. Amin, and S. Vollala, "Blockchain based secured information sharing protocol in supply chain management system with key distribution mechanism," *J. Inf. Secur. Appl.*, vol. 54, 2020, Art. no. 102554.

[15] M. Lou, X. Dong, Z. Cao, and J. Shen, "SESCF: A secure and efficient supply chain framework via blockchain-based smart contracts," *Secur. Commun. Netw.*, vol. 2021, pp. 8884478:1–8884478:18, 2021.

[16] N. Szabo, "Smart contracts," 1994. [Online]. Available: http://szabo.best.vwh.net/smart.contracts.html

[17] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.

[18] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Adv. Cryptol. - CRYPTO, 24th Annu. Int. Cryptol. Conf.*, 2004, pp. 41–55.

Can Zhang received the B.E. degree in computer science and technology in 2017 from the Beijing Institute of Technology, Beijing, China, where he is currently working toward the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His current research interests include security and privacy in VANET, cloud computing security, and blockchain technology.

Liehuang Zhu (Senior Member, IEEE) received the Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004. He is currently a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include security protocol analysis and design, group key exchange protocols, wireless sensor networks, cloud computing, and blockchain applications.

Chang Xu (Member, IEEE) received the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2013. She is currently an Associate Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. Her research interests include security and privacy in VANET, and Big Data security.

Kashif Sharif (Senior Member, IEEE) received the M.S. degree in information technology in 2004, and P.hD. degree in computing and informatics from the University of North Carolina at Charlotte, Charlotte, NC, USA, in 2012. He is currently an Associate Professor with Beijing Institute of Technology, China. His research interests include information centric networks, blockchain and distributed ledger technologies, wireless and sensor networks, software defined networks, and data center networking. He is an Associate Editor for IEEE ACCESS.

Rongxing Lu (Fellow, IEEE) is currently an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada. He has authored or coauthored extensively in his areas of expertise (with H-index 72 from Google Scholar as of November 2020). His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He was the recipient of nine best (student) paper awards from some reputable journals and conferences. Dr. Lu is the Vice-Chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee).

Yupeng Chen received the Ph.D. degree in computer science and technology from Jilin University, Changchun, China. He has authored or coauthored seven papers in journals and international conference. His research interests include vehicle of cyber security, machine learning, and object detection. He is also a Valued Member of Society of Automotive Engineers International.