sustainability

MDPI

Article

Blockchain-Based Anti-Counterfeiting Management System for Traceable Luxury Products

Chin-Ling Chen [1,2], Long-Hui Guo [3], Ming Zhou [3], Woei-Jiunn Tsaur [4,5,*], Hongyu Sun [6,7,*], Wanbing Zhan [3], Yong-Yuan Deng [2] and Chun-Ta Li [8]

1 School of Information Engineering, Changchun Sci-Tech University, Changchun 130600, China
2 Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung City 413310, Taiwan
3 School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China
4 Computer Center, National Taipei University, New Taipei City 237303, Taiwan
5 Department of Computer Science and Information Engineering, National Taipei University, New Taipei City 237303, Taiwan
6 Department of Computer Science, Jilin Normal University, Siping 136000, China
7 State Key Laboratory of Numerical Simulation, Siping 136000, China
8 Department of Information Management, Tainan University of Technology, Tainan 71002, Taiwan
* Correspondence: wjtsaur@mail.ntpu.edu.tw (W.-J.T.); hongyu@jlnu.edu.cn (H.S.)

Abstract: In recent years, counterfeit luxury products have become a major concern for consumers worldwide. The reason for the proliferation of counterfeit products is that the manufacturing and distribution process is not transparent to consumers and this information can be easily falsified or altered by others. To solve this problem, this paper proposes the development of a management system using blockchain and smart contract technology to solve the problems of data forgery and data tampering, while tracking the information related to luxury products and ensuring the accuracy and authenticity of the relevant data, to achieve the purpose of luxury product anti-counterfeiting. When using Hyperledger Fabric to deploy the blockchain and execute smart contracts, all information related to the production and logistics process of luxury goods will be uploaded to the blockchain. No human intervention is required to create a complete, traceable, tamper-proof, and trusted repository. Compared to previous work, this paper combines blockchain technology with specific processes in the supply chain, employing a variety of security methods to secure the communication process. Moreover, our proposed solution is more flexible in transmission, with more secure protocols also making data harder to tamper with and falsify, thereby solving the problem of forgery and tracking of luxury products.

Keywords: luxury anti-counterfeiting; blockchain; traceability; logistics; Hyperledger Fabric

Citation: Chen, C.-L.; Guo, L.-H.; Zhou, M.; Tsaur, W.-J.; Sun, H.; Zhan, W.; Deng, Y.-Y.; Li, C.-T. Blockchain-Based Anti-Counterfeiting Management System for Traceable Luxury Products. *Sustainability* **2022**, *14*, 12814. https://doi.org/10.3390/su141912814

Academic Editor: Cheolho Yoon

Received: 17 August 2022
Accepted: 24 September 2022
Published: 8 October 2022

1. Introduction

1.1. Background

Luxury products are internationally known as consumer products that are unique, scarce, rare, and expensive beyond people's needs for survival and development; these are also known as non-essential products. Currently, the world's most famous and well-known luxury brands are Chanel, Givenchy, Hermes, Louis Vuitton, Prada, and Gucci, whose commercial products cover all aspects: jewelry and accessories, watches, bags, and luxury clothing. Due to the rapid development of the world economy over the years, the market for the sale of luxury products has also been expanding, leaving people increasingly eyeing the profit margin of counterfeit luxury products. Seeking to make huge profits from luxury products, many have now entered the counterfeit luxury industry, flooding the market with counterfeit products. In 2017, the total global loss of counterfeit products (including loss of brands, jobs, and consumers' health and safety) was US $1.2 trillion, equivalent to 10% of

China's GDP that year. The 2019 Harvard Business Review estimated the global counterfeit trade to be around US $4.5 trillion. Of this, the luxury products industry alone accounts for 60–70% of all counterfeit transactions, far exceeding the values of the pharmaceutical and entertainment industries [1].

The information above shows that the counterfeit industry is causing huge financial losses to consumers, businesses, and governments. In response to the current situation of endless counterfeit luxury products, various brands of luxury products have taken measures to prevent consumers from buying counterfeit products, including hiring plainclothes detectives, setting up special teams to prevent and respond to counterfeit products [2], and introducing the use of QR codes and RFID chips to obtain proof of product information on official websites. Yet, these approaches have not achieved great results and counterfeit luxury products still occupy a large part of the consumer market. Today, counterfeit luxury products are a major problem for brands in the process of selling their products, which is a common concern worldwide. To protect the consumers' rights and interests and maintain the huge consumer market of luxury products, we urgently need more effective anti-counterfeiting technology to stop the current phenomenon of counterfeit products.

In today's luxury market, consumers are unable to distinguish the authenticity of luxury products as a direct result of the fact that the manufacturing and distribution process is not transparent to them. Consumers can only see the products, not the production and sales process and the raw materials used to create them. Furthermore, thanks to the booming global economy and trade, today, the research and development, production, and sales of luxury products for a brand are no longer managed by a few manufacturers but spread across several. As a result, consumers cannot tell the authenticity of a luxury product simply by the manufacturer's information; this leaves them vulnerable to being easily fooled by lawless people into buying counterfeit products.

To remedy the situation, we need anti-counterfeiting technology that is honest, decentralized, irreplicable, and traceable. The blockchain and smart contracts can answer this call, bringing new solutions to several industry problems [3–6]. The blockchain [7] is essentially a shared database, with decentralization, tamper resistance, traceability, and anonymity that can ensure the data stored are not tampered with and that there is no hegemony of a "central" institution or monopoly of a few people. In the luxury sector, its tamper resistance and traceability can support the accurate traceability of luxury information. The smart contract, meanwhile, is a protocol for storing contracts in an informative, digital form on a computer and executing them on the computer. It is non-interfering, traceable, and irreversible, allowing trusted transactions to be made without the involvement of a third party and the status of the transaction to be tracked. The characteristics of the smart contract ensure that once the execution of the contract has begun, the system can automatically execute until the end of the contract without any external interference.

In the luxury sector, if the information relating to the production and sale of a luxury product is defined as a smart contract in advance, then once the "contract" is executed, the information relating to the production and sale process can be recorded without human interference and any tampering. Based on a close connection between the production and sales of products and the supply chain, if the design, production, and sales of luxury products are combined with blockchain and smart contract technology—for example, product manufacturers and raw material suppliers reach an agreement on the supply of raw materials through smart contracts, or product manufacturers and product distributors reach an agreement on the sale of products through smart contracts—then consumers and third parties will be able to trace the production and circulation history of products at any time, simply and completely, thereby benefitting from product traceability and anti-counterfeiting.

This paper proposes a blockchain and smart contract-based anti-counterfeiting management system for traceable luxury products. We show that when using the decentralized, consensus mechanism, the anonymity, immutability, transparency, and high reliability and confidentiality of blockchain technology make it inherently more advantageous for

anti-counterfeiting than other methods proposed. Combining the blockchain with smart contract technology solves the problem of mistrust in the system. Without a third party and without establishing a trust relationship, the internal system can automatically conduct transactions according to the content of the smart contract, free of human interference. The operation of the entire system and the chain storage of data are secure and transparent, thus eliminating the possibility of forgery and tampering. The study also uses the Hyperledger Fabric architecture to implement a traceability prototype, which is an open-source blockchain project for enterprise applications launched by the Linux Foundation in December 2015 [8]. This can process transaction requests efficiently and without relying on digital currency. By utilizing the prototype, the Hyperledger Fabric architecture, combined with blockchain and IoT technologies, enables secure and true anti-counterfeit traceability of products.

1.2. Related Works

The most widely used anti-counterfeiting systems in the current market for clothing and watches are those based on QR codes and databases and those based on RFID, which have the following drawbacks:

1. Anti-counterfeiting systems based on QR codes and databases: QR codes can be copied, and since their contents are fixed, unscrupulous parties can achieve counterfeiting effects in this way; lawless parties can forge or copy the database of genuine products according to the contents of each column of the genuine serial number; lawless parties can forge and create a pirated website similar to the genuine one and pretend to be genuine merchants to deceive consumers; merchants have the right to change the database, so merchants can manipulate the database themselves, making online verification no longer credible [9–13];
2. RFID-based anti-counterfeiting systems: Since the data inside the chip can be read and written, there are many forged tags, which do not have their unique attributes. In the verification process of anti-counterfeiting, the information is transmitted through the Internet channel. Once the encryption measures of the commodity information are not in place, since the data management system tends to be centrally managed, there is a risk of the information being cracked and stolen, leading to the leakage and exposure of the commodity information [14].

To solve the problem of counterfeiting luxury clothing as well as products such as watches, many different solutions have been proposed based on previous concepts, as shown in Table 1.

Dan et al. [2] proposed the use of the EPC Internet of Things for the anti-counterfeiting of luxury products. Although information about the products can be obtained, there is a problem with information security because the data are transmitted through a network, and the cost is too high for small products such as cosmetics and skin care products because each product has to be equipped with an RFID microchip. Hochholdinger et al. [15] proposed that the physical and chemical analysis of marks or traces can achieve the effect of watch anti-counterfeiting, as well as provide a way to determine where the corresponding parts were produced, thereby achieving a certain degree of traceability. Yet, the process is too specialized, and ordinary consumers must seek professional help to obtain authenticity results; furthermore, its traceability is unstable, with a certain degree of chance. Perez et al. [16] proposed the latest solution and framework for traceability of garments, which enables the tracking of all suppliers and customers in the logistics chain, but lacks a specific description of the data flow framework. Kumar et al. [17] proposed a blockchain-based traceability framework for verifying and tracking the supply chain of clothing, but due to the lack of some IoT technologies, consumers cannot access the transaction information on the blockchain themselves.

Table 1. Comparison with existing anti-counterfeiting traceability methods.

Authors	Year	Objective	Technologies	Merits	Demerits
Dan et al. [2]	2012	Proposed a luxury anti-counterfeiting system architecture and hierarchy based on the EPC Internet of Things (IoT)	EPC and IoT	Proposed a system architecture that can meet the demand for luxury anti-counterfeiting	Encryption and decryption methods are not specific enough and need to be improved
Hochholdinger et al. [15]	2019	Marks or traces, from a forensic intelligence perspective, can achieve a watch anti-counterfeiting effect	Link Analysis and Chemical and Physical Profiling	Revealed the links between watches that were unknown or uncertain and demonstrated the interconnection of all watches on a chemical and physical level	Specialized personnel are required to conduct the identification operation, and the marks produced are sporadic and unstable
Perez et al. [16]	2020	Introduced the latest traceability program and recommended a framework for garments	Blockchain and Hash Functions	Ensures the authenticity, reliability, and integrity of clothing while ensuring the transparency of the supply chain	The specific process for garment production was not presented
Kumar et al. [17]	2021	Investigated and proposed a blockchain-based traceability framework for traceability in a multi-tier T&C supply chain	Blockchain and Smart Contract	A framework combining the blockchain and supply chain was proposed	The specific flow of data was not reflected

The anti-counterfeiting solutions proposed in the above-mentioned literature have some deficiencies, such as (1) the difficulty of integrating blockchain technology with the specific processes of the supply chain. (2) The data transmission process has information security problems and cannot effectively protect product-related information in practical operations. Through learning from and improving on the solutions in the above literature, this paper proposes a blockchain-based anti-counterfeiting management system for traceable luxury products. The Hyperledger Fabric is used to build a consortium blockchain to deploy and execute smart contracts, and to upload all the information related to the production of raw materials, producers, consumers, and the flow and logistics of the products in the production process of luxury products. Given the decentralized, tamper-evidencing, and traceable features of the blockchain, the system can achieve decentralized storage of data. The system does not rely on other regulatory bodies or hardware facilities to store the information on the chain, making it a complete, traceable, and credible record. The system also makes use of the characteristics of smart contracts to strictly enforce the pre-agreed rules without human intervention, and it can transparently disclose the current logistics flow of products in real-time according to the execution of smart contracts.

Thus, this anti-counterfeiting management system can achieve the following effects:

1. In terms of security, this system adopts the Elliptic Curve Digital Signature Algorithm (ECDSA) in cryptography to protect the integrity, traceability, and non-repudiation of data, thus further ensuring the security of information related to luxury products in the process of transmission;
2. In terms of anti-counterfeiting and traceability, the system incorporates the Internet of Things (IoT), thereby enabling consumers to query specific information on the production and sales process of luxury products and the flow of products in real-time,

and uses smart contracts to prevent human intervention in the process and ensure the accuracy of the data;

3. In terms of logistics and distribution, when the logistics delivery is made, both parties must confirm the integrity of the luxury products. If their integrity is confirmed, both parties will scan a code to confirm, which will trigger the smart contract and upload the logistics information, a timestamp, and the identity information of both parties to the blockchain center. The corresponding responsible unit can then be contacted for that information if needed, for instance, if the products are damaged or swapped;
4. In terms of its framework, this system uses Hyperledger Fabric, which combines the supply chain with the blockchain to further ensure the correct transmission of data.

The remainder of the paper is organized as follows: Section 2 introduces the technologies used in the study; Section 3 presents the architecture and research methodology of the system; Section 4 provides a series of analyses of the system, and Section 5 provides a discussion of the system's performance. Finally, the paper is summarized in Section 6.

2. Preliminary

2.1. Elliptic Curve Digital Signature Algorithm (ECDSA)

In the cryptography field, the Elliptic Curve Digital Signature Algorithm (ECDSA) [18] provides a variant of the standard Digital Signature Algorithm (DSA). As with elliptic curve cryptography, the bit size of the public key required for ECDSA is about twice the security level. For example, to achieve an 80 bit security level, the size of an ECDSA public key needs to be 160 bits, while to achieve the same 80 bit security level, a DSA public key needs to be at least 1024 bits in size.

When signing and verifying the ECDSA, initially, both parties must agree on the curve parameter $(CURVE, G, n)$, In addition to the equation of the curve and the base point G on the curve, the sender needs the private key dA, the public key QA, and the message M (where $K = kG$).

1. When the sender needs to send a message, they first choose a random value k in [1, n - 1], calculate it $z = h(m)$, $(x_1, y_1) = kG$, $r = x_1 \ mod \ n$, $s = k^{-1}(z + r * dA) \ mod \ n$, and then send the ECDSA signature pair (r, s) together with the original message M to the receiver (where h is the hash function);
2. Once the receiver has received the signature pair (r, s) and the original message M, it verifies the validity of the ECDSA signature. First, it calculates $z\prime = h(M)$, and second, it calculates $u_1 = (z\prime * s^{-1}) \ mod \ n$ and $u_2 = (r * s^{-1}) \ mod \ n$. Then, it determines the equality of r and ${x_1}' \ mod \ n$. If the values are equal, the receiver confirms that the ECDSA signature and message sent by the sender are valid.

2.2. Smart Contract

The smart contract was first proposed by interdisciplinary legal scholar Nick Szabo in 1995. It is defined as follows: a smart contract is a group of commitments defined in digital form, including the contract participants. Contract participants can execute the agreements reached through smart contracts [19]. The blockchain can allow multiple business entities to collaborate and foster trust through smart contracts, thus expanding the scope and depth of mutual cooperation between participants. When the blockchain hears the trigger condition of a smart contract, it will put the contract into a queue and distribute it to each node. When each node receives the contract, it will first verify the correctness of the contract, and if it is correct, it will execute the corresponding contract content code and package the final result on the blockchain.

2.3. Blockchain

The blockchain is essentially a centrally maintained decentralized database [20]. It has features such as decentralization, consensus mechanisms, the immutability of data, and transparency [21]. Decentralization means that in the blockchain, each node has the same right, and there is no node with the highest right. Compared to the centralized system,

it can effectively prevent the highest power of person or institution from changing the data at will, so that all users who join the blockchain can participate in data authenticity verification, thus eliminating the shortcomings of a single authentication center in the traditional authentication system. At the same time, all nodes on the blockchain can reach a consensus mechanism; as long as one node has a different concept from the others, no consensus can be reached, which can effectively deal with the problem of a "Byzantine General". The high degree of data inerrability and its transparency are the basis of the entire blockchain. The blockchain is composed of multiple blocks, where the generation of the next block relies on the hash value of the data of the previous block. Once the data of the previous block are tampered with, the hash value of the block will be changed and the rest of the nodes will collectively consider this node a "bad" node, thus ensuring the security of the data. Luxury products are suitable candidates for the application of the blockchain for product anti-counterfeiting traceability [22].

2.4. Hyperledger Fabric

In 2017, the Linux Foundation launched the open-source blockchain project Super Ledger (Hyperledger) [23], which has five main subprojects: Fabric, Sawtooth, Indy, Burrow, and Iroha. Among them, the most popular is the Fabric consortium blockchain [24]. Unlike Bitcoin and Ethereum, Hyperledger Fabric is a consortium blockchain platform designed specifically for enterprise-level blockchain applications, which does not have any cryptocurrency, and where member nodes must be authorized to join and gain access. It also has the decentralized, tamper-proof, transparent, and traceable features of the blockchain, supports multiple smart contract writing languages, has a pluggable consensus, and has a unique communication mechanism for sharing information between members, which provides excellent transaction throughput while ensuring good privacy [25]. In these ways, Hyperledger Fabric is highly suited to the application needs of data sharing between enterprises. Hyperledger Fabric, as a consortium blockchain, mainly consists of Certificate Authorities (CA), a Peer Node, Chaincode, Channel, Ordering Service, Client, etc. [26].

3. Proposed Scheme

3.1. System Structure

In this study, a blockchain-based anti-counterfeiting management system for the traceability of luxury products is proposed to be designed using the Elliptic Curve Digital Signature Algorithm (ECDSA). As shown in Figure 1, blockchain technology is applied to the complete production and distribution history of luxury products, allowing third parties to verify that at any time.

In this system, the Brand Party (BP), Product Manufacturer (PM), Material Supplier (MS), Product Distributor (PD), and Logistics Provider (LP) will form a consortium blockchain, which together with the Third-Party Verify Organization (TPOV), Blockchain Center (BCC), Consumers (CO), and Deliveryman (DM), will form a Hyperledger Fabric-based luxury anti-counterfeiting management system.

1. Blockchain Center (BCC): the BCC accepts registrations from all parties and issues proof of identity and public/private key pairs to each party;
2. Brand Party (BP): The BP is responsible for the design of the product, reviewing whether the product manufacturer has produced a product that meets the standards, and controlling the eligibility of the product to be marketed. The supply chain only operates when the brand decides to produce a luxury product;
3. Product Manufacturer (PM): Based on the product design drawings sent by the BP, the PM sends raw material requirements to the MS. When the BP gives them the go-ahead, the PM will produce the product and attach a web page or client interactive website containing the blockchain network to the product, which it sends to the distributor;
4. Material Supplier (MS): the MS provides the raw materials for manufacturing products according to the raw material demand information sent by the PM, and records the sources of materials;

5. Product Distributor (PD): the PD is a third-party sales platform or direct shop that sells products to customers, and which can upload transactions;
6. Logistics Provider (LP): responsible for the delivery of products from third-party sales platforms or directly managed shops to consumers, able to confirm upload transactions with consumers and deliverymen, and record the logistics path of products;
7. Consumers (CO): when the integrity of the products received is confirmed, the CO (or DM) will confirm the upload with the logistics provider;
8. Deliveryman (DM): when the integrity of the product is confirmed, the DM (or CO) will confirm the upload with the logistics provider;
9. Third-Party Verify Organization (TPVO): inspection agencies that check the information about luxury products through an app or with the client.

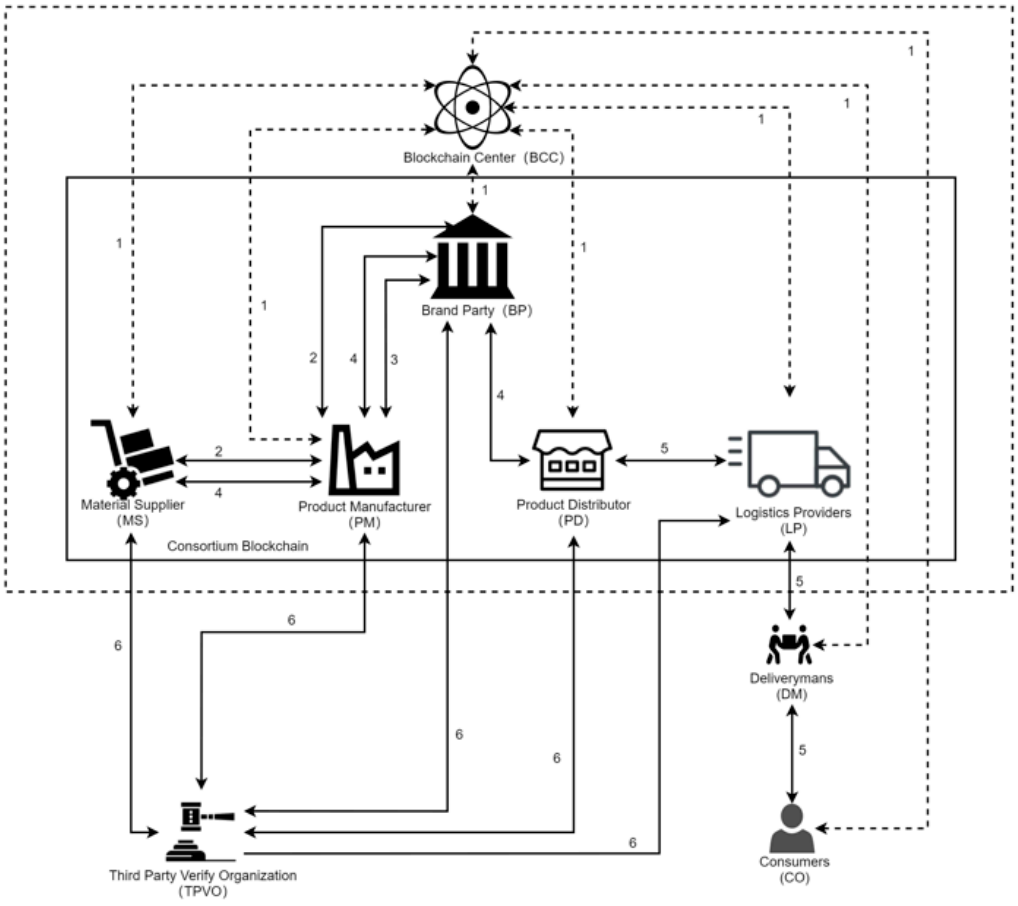

Figure 1. System architecture diagram.

Step 1: The registration phase for each role in the system. All Brand Parties (BP), Material Suppliers (MS), Product Manufacturers (PM), Product Distributors (PD), Logistics Providers (LP), Deliverymen (DM), and Consumers (CO) need to authenticate their access to the blockchain at the CA node in the center of the blockchain. Then, they will get the ECDSA signed public and private keys issued by the CA node, which will add them to the blockchain network, where they interact through a channel;

Step 2: When a BP wants to design a new product, it enters the product design development phase. The BP makes a design for the new product, applies for a patent, signs the design and the patent license with a private key, and sends it to the PM, who verifies that the signature is correct and then signs the information about the raw materials needed to develop the new product, sending this to the MS. After the MS verifies the correctness of the signature, it returns a confirmation message of the product's raw material content, which contains the product's raw material identification code, the

product's raw material content, and the source of the raw material. After verifying the correctness of the message, the PM adds the information that the product is a sample and signs the above product information, and uploads it to the blockchain center via a sorting node to update the local ledger;

Step 3: After the PM has completed the production of the product sample, it enters the product evaluation and approval phase. The PM carries out a corresponding evaluation of the produced sample product and, after the evaluation, sends the product-related evaluation data and raw material-related information, signed with a private key, to the BP. If the BP approves the product, they return a signed production certificate for the product to the PM;

Step 4: After the PM has obtained the production certificate for the product and verified its correctness, it enters the ordering and production phase. The PD sends the order to the BP, who confirms the order information and sends it to the PM with a signature. The PM produces the product according to the order information, signs the information on the raw materials required to produce the product, and sends it to the MS, who verifies that the signature is correct and provides the required raw materials for production, sending the information on the raw materials to the PM with a signature. The PM verifies the correctness of the signature and then proceeds with the mass production of the product, while marking the product with a unique identification code that allows the user to query the product information in the future. Once the product has been produced, the PM sends the product to the PD and integrates the information about the product into the order information, signing it with a private key and uploading it to the blockchain center via a sorting node to update the local ledger;

Step 5: The product purchase and logistics delivery phase. After the CO purchases the product from the PD, the product is signed by both parties and enters the logistics delivery phase. During the product delivery process, the CO can check the information related to the product logistics in real-time. There are two situations in logistics delivery: (1) When two DMs hand over the product, both DMs need to jointly scan the code to confirm the product is complete and error-free, triggering a smart contract to upload the logistics information, timestamp, and identity information of both parties to the blockchain center. (2) When the CO receives the product, they also need to confirm that the product is complete and then scan the code with the DM to trigger the smart contract, which uploads the logistics information, timestamp, name of the signatory, and the DM's information to the blockchain center. When there is a problem with the delivery of the product, the logistics and delivery can be traced;

Step 6: All TPVO validation (e.g., consumers, inspection bodies, blockchain values, product approval certificates, etc.) can be verified using the public keys of the BP, MS, PM, and PD. After that, they can verify the complete production history and legitimacy of the product. Furthermore, the TPVO verifies the signature information of the PM and MS to confirm that the content of the raw material is correct and that the product is a sample or commercially available. Moreover, the TPVO verifies the signatures of the PM and BP and the product approval certificate to verify the marketing authorization of the product. In addition, the TPVO verifies the signatures of the PM and the BP to obtain test data for the product. Finally, the TPVO can verify the route of the product during logistics and the handover between the two parties.

3.2. Notation

Table 2 provides definitions for the symbols that will be used in this paper.

Table 2. Notations.

Symbol	Description
ID_i	User identification
$product\_ID_i$	Identification of the i-th product of the BP
$order\_ID_i$	Identification of the i-th product order
$logistics\_ID_i$	Identification of the i-th logistics order
$List < product\_ID_i >$	A collection of N product representations
$List < order\_ID_i >$	A collection of N product order representations
$List < logistics\_ID_i >$	A collection of N logistics order representations
E	Elliptic curves defined on a finite group
G	A generated point based on an elliptic curve E
k_i	i-th random value on the elliptic curve
d_X	Private key for user X
Q_X	Public key for user X
(r_{Xi}, s_{Xi})	Elliptic curve eigenvalues of X
(x_{Xi}, y_{Xi})	ECDSA signature value for X
$E_{PukX}(M)$	Encryption of message M using the public key of user x
$D_{PrkX}(M)$	Decryption of message M using the private key of user x
$H(M)$	One-way hash function for message M
h_{Xi}	i-th hash value of X
T_i	i-th timestamp
ΔT	Threshold for checking the validity of timestamps
M_{BP_DES}	Information from the BP (product information)
M_{BP_Order}	Information from the BP (product orders)
M_{X_Y}	Message is sent from X to Y
M_{MS}	Information from MS (type of raw material, content, source, etc.)
$A1 \stackrel{?}{=} A2$	Verify if $A1$ is equal to $A2$

3.3. Initialization Phase

This research will build a consortium blockchain, and deploy and execute smart contracts through the Hyperledger Fabric. Some key information about the design, production, sales, logistics, and distribution of luxury products will be stored and verified through a blockchain center. This blockchain key information needs to be defined in a smart contract. Figure 2 shows the blockchain smart contract structure for this solution, and Figure 3 shows the structure of the roles of those involved in the supply chain. The following key information will be stored in the blockchain.

(1) In the luxury product structure, information about the product, its specific content, and the material supplier, product manufacturer, consumer, and logistics order numbers are recorded;
(2) Role types and related procedural structures are assigned to those involved in the supply chain;
(3) In the structure of the product order, information about the product order, its specific content, and status, as well as the product distributor and product manufacturer numbers, are recorded;
(4) In the structure of the logistics order, information about the logistics order, specific contents, status, the number of couriers handing it over during the logistics delivery, the handover time, and the numbers of product distributors, consumers, and logistics companies are recorded.

```
struct smart contract LuxuryProduct{
    string Product_idNumber; //ID
    string Product_name; //Name
    string Product_kind; //Type
    string Product_designDate; //Design Time
    string Product_createDate; //Production time
    string Product_BP_idNumber; //BP UID
    string Product_MS_idNumber; //MS UID
    string Product_material; //Material Information
    string Product_PM_idNumber; //PM UID
    string Product_testResult; //Test Result
    string Product_PD_idNumber; //PD UID
    string Product_CO_idNumber; //CO UID
    string Product_courier_idNumber; //Logistics Or-
der UID
    string Product_price; //Price
    Content Product_content; //Product Content
    string BP_Signature; //Signature of BP
    string PM_Signature; //Signature of PM
    string MS_Signature; //Signature of MS
    string PD_Signature; //Signature of PD
    string LP_Signature; //Signature of LP
    string[] DM_Signature; //Signature of Deliveryman
    string CO_Signature; //Signature of Consumer
}
```

```
struct smart contract ProductOrder{
    string Order_PD_idNumber; //PD UID
    string Order_BP_idNumber; //BP UID
    string Order_PM_idNumber; //PM UID
    string Order_kind; //Order Type
    Content Order_information; //Order Content
    string Order_createDate; //Order Generation Time
    string Order_state; //Order State
}

struct smart contract LogisticsOrder{
    string Logistics_idNumber; //Logistics Order UID
    string Logistics_PD_idNumber; //PD UID
    string Logistics_LP_idNumber; //LP UID
    string Logistics_CO_idNumber; //CO UID
    string Logistics_product_idNumber; //Product UID
    string Logistics_kind; //Order Type
    var Logistics_inf []LogisticsRecord; //Logistics
                                                Content
    string Logistics_createDate; //Order Generation Time
    string Logistics_finishDate; //Order Completion Time
    string Logistics_state; //Order State
}
```

Figure 2. Structure of the smart contract in the proposed scheme.

```
type Roles string const{
    BrandParty;
    MaterialSupplier;
    ProductManufacturer;
    ProductDistributor;
    LogisticsProvider;
    Deliveryman;
    Consumer;
    ThirdParty;
}
```

```
Type Roles_Information struct{
    string idNumber;
    string name;
    string detail;
    Var RoleTypes Roles;
}
Type LogisticsRecord struct{
    Shipper_idNumber string//Deliveryman UID
    Recipient_idNumber string//Deliveryman or Consumer UID
    Recipient_Location string//Deliveryman or Consumer Receiving Express
                                Location
    Recipient_time time.Time//Deliveryman or Consumer Receiving Express
                                Time
}
```

Figure 3. Structure and enumeration of the roles of those participating in the supply chain.

3.4. Registration Phase

During the registration phase, all participants (BP, PM, MS, PD, LP, DM, CO, and TP) are required to register in the Blockchain Centre when using the system for the first time. They follow the instructions on the registration page to register their information by their identity. Once registration is complete, if the information is verified to be correct, the selected role identifier will be sent to the blockchain center and the system will generate a specific public-private key pair for the user based on the registration information. It will also bind the user's password and username to the public-private key pair, which will then

be returned to the merchant or individual. Figure 4 uses "Roles X" to represent all arbitrary roles in the blockchain system, with a flow chart of the registration phase shown.

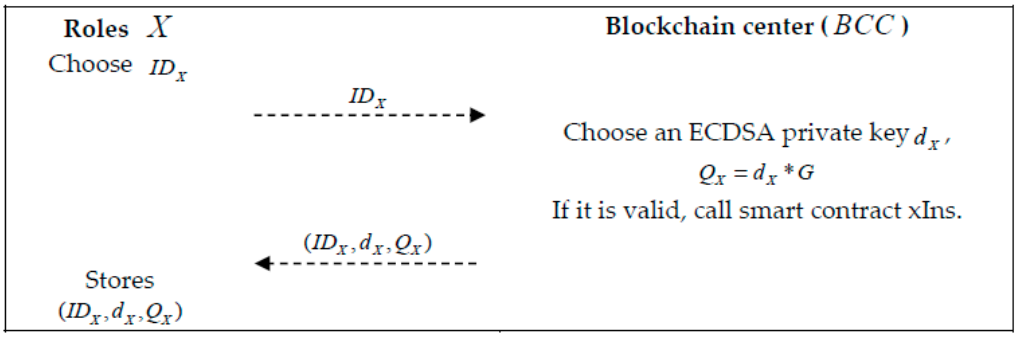

Figure 4. Registration stage flow chart.

Step 1: Those involved in the system, in all roles $(Roles\ X)$, each generate an ID_X identity and then send their generated ID_X through the application to a CA node in the blockchain center for registration and validation, to determine the legitimacy of their identity;

Step 2: The CA node at the center of the blockchain generates an ECDSA private key d_X based on the roles in the system and calculates the public key Q_X:

$$Q_X = d_X * G \tag{1}$$

When the correct identities of all roles have been verified, the smart contract registration will be triggered, as shown in Algorithm 1;

Step 3: All roles in the system receive the signature message parameter (ID_X, d_X, Q_X) and store it.

Algorithm 1. A scheme for chain code registration.

```
var X[]Roles X
func Registration (X_name string, X_detail string, var X_role RoleType) (idNumber string) {
    idNumber = GenerateUniqueID()
    X = append (X, Roles X{
    idNumber: product_idNumber,
    name: X_name,
    detail: X_detail,
    Role: X_role,
})
    return idNumber
}
```

3.5. Product Design and Development Phase

In the product design and development phase, the BP sends the drawings of the newly designed product to the PM, who produces the product according to the designed drawings and asks the MS for the raw materials of the product according to the type and content of the raw materials in the drawings. The main participants in this phase are the BP, the PM, and the MS, and the flow chart is shown in Figures 5 and 6.

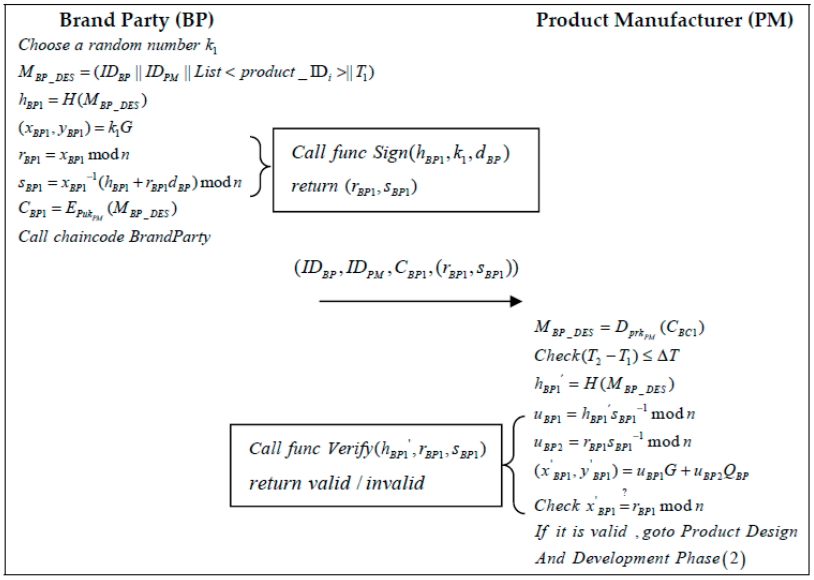

Figure 5. Product design and development phase (1).

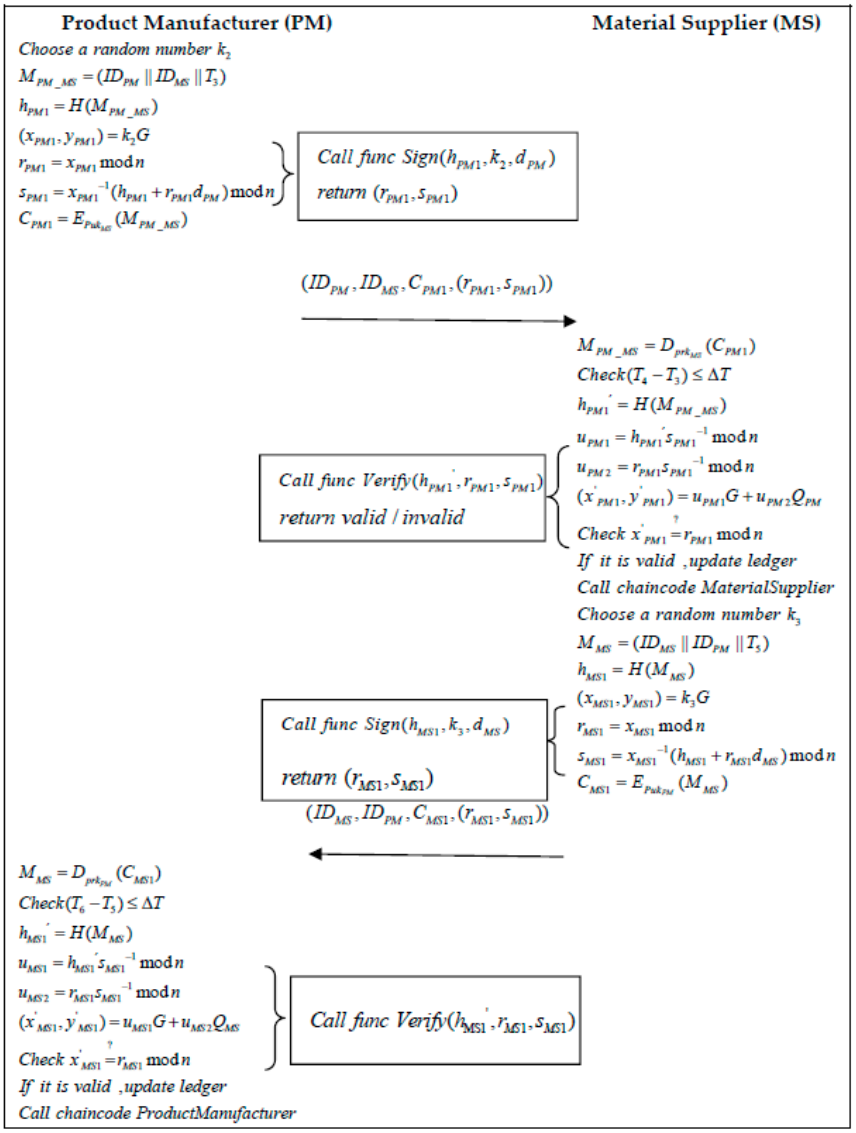

Figure 6. Product design and development phase (2).

Step 1: When the BP designs a new product, it signs the design and the patent license and delivers it to the PM at time T_1. The design includes product information and product material information. The BP chooses a random number k_1 and generates a message M_{BP_DES} containing multiple product identifiers:

$$M_{BP\_DES} = (ID_{BP}||ID_{PM}||List < product\_\mathrm{ID}_i > ||T_1) \quad (2)$$

The BP then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{BP1}, s_{BP1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{BP1} = H(M_{BP\_DES}) \quad (3)$$

$$(r_{BP1}, s_{BP1}) = Sign(h_{BP1}, k_1, d_{BP}) \quad (4)$$

The generated encrypted message is encrypted by the PM's public key:

$$C_{BP1} = E_{Puk_{PM}}(M_{BP\_DES}) \quad (5)$$

The *BP* then executes the "BrandParty" function, which sends the $(ID_{BP}, ID_{PM}, C_{BP1}, (r_{BP1}, s_{BP1}))$ information to the PM, as shown in Algorithm 3.

Algorithm 2. A scheme for a communication protocol.

```
func Sign (h string,k string, d string)(r string, s string){
   (x,y) = k*G;
   r = x%n
   s = (h + r*d)/x%n
   return r,s
}
func Verify(h string,r string,s string)(result string){
   u1 = h/s%n
   u2 = r/s%n
   (x,y) = u1*G + u2*Q
   If x = r{
return "valid"
   }else{
return "invalid"
   }
}
```

Step 2: When the PM receives a message from the BP at T_2 moment, it first decrypts the message using its private key:

$$M_{BP\_DES} = D_{prk_{PM}}(C_{BC1}) \quad (6)$$

Secondly, the PM verifies the validity of the timestamp:

$$Check(T_2 - T_1) \leq \Delta T \quad (7)$$

Once the validity of the time has been verified, the PM uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{BP1}{}' = H(M_{BP\_DES}) \quad (8)$$

$$Verify(h_{BP1}{}', r_{BP1}, s_{BP1}) \quad (9)$$

If the signature verification is valid, the PM signs the raw material type and content requirements for the product at time T_3 and delivers them to the MS. The PM chooses

a number k_2 at random and also generates the product raw material requirement information M_{PM_MS}:

$$M_{PM\_MS} = (ID_{PM}||ID_{MS}||T_3) \tag{10}$$

The PM then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{PM1}, s_{PM1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{PM1} = H(M_{PM\_MS}) \tag{11}$$

$$(r_{PM1}, s_{PM1}) = Sign(h_{PM1}, k_2, d_{PM}) \tag{12}$$

The generated message is encrypted with the MS public key:

$$C_{PM1} = E_{Puk_{MS}}(M_{PM\_MS}) \tag{13}$$

The PM then sends the $(ID_{PM}, ID_{MS}, C_{PM1}, (r_{PM1}, s_{PM1}))$ information to the MS.

Algorithm 3. Chain code of the product design and development phase.

```
var LP []LuxuryProduct; count = 0;
func BrandParty (int num, P_Name string, P_kind string, ID_BP string, Date string, Product_INF
string, Signature string) (Product_IDs []string){
for i:= 0; i < num; i++ {
count = count + 1
LP = append (LP, new LuxuryProduct{Product _idNumber = Generate_product_id()})
Product_IDs = append(Product_IDs, TP[count]. Product_idNumber)
}
for i:= 0; i < num; i++ {
index:= SearchProduct_ID(Product_IDs[i]);
   LP[index].Product_name = P_Name
   LP[index].Product_kind = P_kind
   LP[index].Product_brand_idNumber = ID_BP
   LP[index].Product_information = append(LP[index].Product_information, Product _INF)
   LP[index].designDate = Date
   LP[index].BP_Signature = Signature
}
}
func ProductManufacturer (Material string, ID_PM string, Product_INF string, price string,
Product_IDs []string, Signature string) {
for i:= 0; i < Product_IDs.Length; i++ {
index:= SearchProduct_ID(Product_IDs[i]);
   LP[index].Product_material = Material
   LP[index].Product_information = append(LP[index].Product_information, Product _INF)
   LP[index].Product_price = price
   LP[index].createDate = time.Now()
   LP[index].Product_manufacturer_idNumber = ID_PM
   LP[index].PM_Signature = Signature
}
}
func MaterialSupplier (ID_MS string, Product_IDs []string, Signature string){
for i:= 0; i < Product_IDs.Length; i++ {
index:= SearchProduct_ID(Product_IDs[i]);
   LP[index].Product_MS_idNumber = ID_MS
   LP[index].MS_Signature = Signature
}
}
```

Step 3: When the MS receives a message from the PM at T_4 moment, it first decrypts the message using its private key:

$$M_{PM\_MS} = D_{prk_{MS}}(C_{PM1}) \tag{14}$$

Secondly, the PM verifies the validity of the timestamp:

$$Check(T_4 - T_3) \leq \Delta T \tag{15}$$

Once the validity of the time has been verified, the PM uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{PM1}{}' = H(M_{PM\_MS}) \tag{16}$$

$$Verify(h_{PM1}{}', r_{PM1}, s_{PM1}) \tag{17}$$

If the signature verification is valid, the MS executes the "Material Supplier" function at time T_5. The algorithm of this function is shown in Algorithm 3, which provides the PM with the raw materials needed to produce the product according to the raw material requirements in the PM's message, as well as the type, content, and source of the raw materials. The MS randomly selects a number k_3 and generates a message M_{MS}:

$$M_{MS} = (ID_{MS}||ID_{PM}||T_5) \tag{18}$$

The MS then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{MS1}, s_{MS1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{MS1} = H(M_{MS}) \tag{19}$$

$$(r_{MS1}, s_{MS1}) = Sign(h_{MS1}, k_3, d_{MS}) \tag{20}$$

The generated message is encrypted by the PM's public key:

$$C_{MS1} = E_{Puk_{PM}}(M_{MS}) \tag{21}$$

The MS then sends the $(ID_{MS}, ID_{PM}, C_{MS1}, (r_{MS1}, s_{MS1}))$ information to the PM.

Step 4: When the PM receives a message from the MS at T_6 moment, it first decrypts the message using its private key:

$$M_{MS} = D_{prk_{PM}}(C_{MS1}) \tag{22}$$

Secondly, the PM verifies the validity of the timestamp:

$$Check(T_6 - T_5) \leq \Delta T \tag{23}$$

Once the validity of the time has been verified, the PM uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{MS1}{}' = H(M_{MS}) \tag{24}$$

$$Verify(h_{MS1}{}', r_{MS1}, s_{MS1}) \tag{25}$$

If the signature verification is valid, the PM executes the "ProductManufacturer" function, the algorithm of which is shown in Algorithm 3, updates the local ledger with the information, and produces a sample of the product.

With these four steps, the product design and development phase is completed and details such as the quantity of raw materials and the design concept of the product should be updated in the account library.

3.6. Product Evaluation and Approval Phase

In the product evaluation and approval phase, the PM tests the performance of the manufactured product and sends the results of the test to the BP, who decides whether to market the product or not based on the results of the performance test. The main players in this phase are the BP and the PM, as shown in Figure 7.

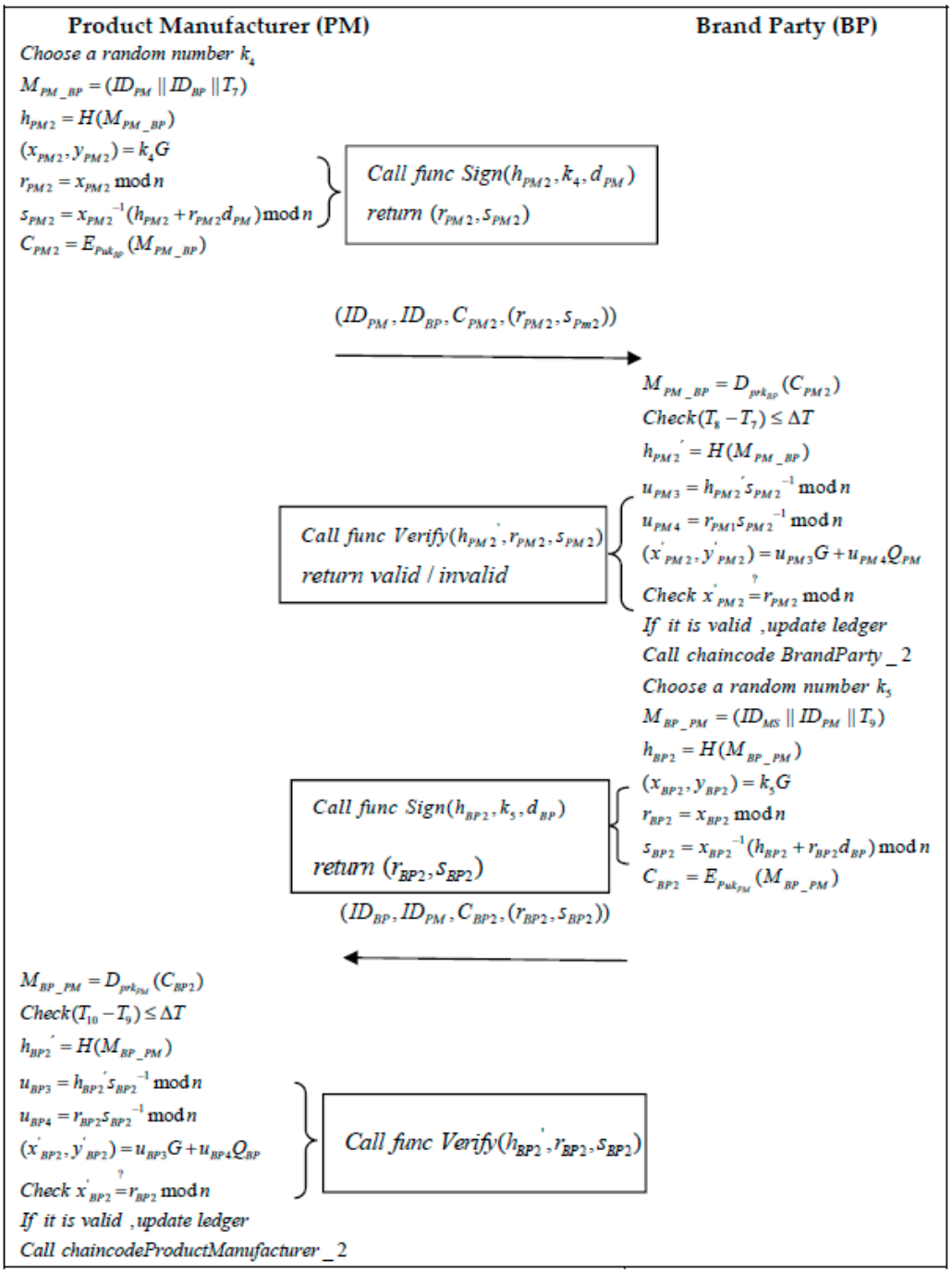

Figure 7. Product evaluation and approval phase.

Step 1: For the product samples produced during the product design and development phase, the PM evaluates the product, finishes the evaluation at moment T_7, and

sends the test results to the BP with a signature. The PM randomly selects a number k_4 and generates the product evaluation result information M_{PM_BP}:

$$M_{PM\_BP} = (ID_{PM}||ID_{BP}||T_7) \tag{26}$$

The PM then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{PM2}, s_{PM2}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{PM2} = H(M_{PM\_BP}) \tag{27}$$

$$(r_{PM2}, s_{PM2}) = Sign(h_{PM2}, k_4, d_{PM}) \tag{28}$$

The generated message is encrypted by the BP public key:

$$C_{PM2} = E_{Puk_{BP}}(M_{PM\_BP}) \tag{29}$$

The PM then sends the $(ID_{PM}, ID_{BP}, C_{PM2}, (r_{PM2}, s_{PM2}))$ information to the BP.

Step 2: When the BP receives a message from the PM at moment T_8, it first decrypts the message using its private key:

$$M_{PM\_BP} = D_{prk_{BP}}(C_{PM2}) \tag{30}$$

Secondly, the BP verifies the validity of the timestamp:

$$Check(T_8 - T_7) \leq \Delta T \tag{31}$$

Once the validity of the time has been verified, the BP uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{PM2}{}' = H(M_{PM\_BP}) \tag{32}$$

$$Verify(h_{PM2}{}', r_{PM2}, s_{PM2}) \tag{33}$$

If the signature verification is valid, the BP looks at the test data of the product at moment T_9. If it is satisfied with the test data of the product, it executes the "BrandParty_2" function, the algorithm of which is shown in Algorithm 4, signs the production license of the product, and gives it to the PM. The BP selects a random number k_5 and generates the product license information M_{BP_PM}:

$$M_{BP\_PM} = (ID_{MS}||ID_{PM}||T_9) \tag{34}$$

The BP then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{BP2}, s_{BP2}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{BP2} = H(M_{BP\_PM}) \tag{35}$$

$$(r_{BP2}, s_{BP2}) = Sign(h_{BP2}, k_5, d_{BP}) \tag{36}$$

The generated encrypted message is encrypted by the PM's public key:

$$C_{BP2} = E_{Puk_{PM}}(M_{BP\_PM}) \tag{37}$$

The BP then sends the $(ID_{BP}, ID_{PM}, C_{BP2}, (r_{BP2}, s_{BP2}))$ information to the PM.

Step 3: When the PM receives a message from the BP at moment T_{10}, it first decrypts the message using its private key:

$$M_{BP\_PM} = D_{prk_{PM}}(C_{BP2}) \tag{38}$$

Secondly, the PM verifies the validity of the timestamp:

$$Check(T_{10} - T_9) \leq \Delta T \tag{39}$$

Once the validity of the time has been verified, the PM uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{BP2}{}' = H(M_{BP\_PM}) \tag{40}$$

$$Verify(h_{BP2}{}', r_{BP2}, s_{BP2}) \tag{41}$$

If the signature verification is valid, the PM executes the "ProductManufacturer_2" function, the algorithm for which is shown in Algorithm 4, and updates the local ledger with the information.

With these three steps, the product evaluation and approval phase is complete.

Algorithm 4. Chain code of the product evaluation and approval phase.

```
func BrandParty_2 (ID_BP string, Product_INF string, Product_IDs []string, Signature string) {
for i:= 0; i < Product_IDs.Length; i++ {
index:= SearchProduct_ID(Product_IDs[i]);
   LP[index].Product_brand_idNumber = ID_BP
   LP[index].Product_information = append(LP[index].Product_information, Product _INF)
   LP[index].BP_Signature = Signature
}
}
func ProductManufacturer_2 (result string, ID_PM string, Product_INF string, Product_IDs
[]string, Signature string) {
for i:= 0; i < Product_IDs.Length; i++ {
index:= SearchProduct_ID(Product_IDs[i]);
   LP[index].testResult = result
   LP[index].Product_information = append(LP[index].Product_information, Product _INF)
   LP[index].Product_manufacturer_idNumber = ID_PM
   LP[index].PM_Signature = Signature
}
}
```

3.7. Product Ordering and Production Phase

In the product ordering and production stage, the PD needs to send the ordering information to the BP. After the BP receives the ordering information, if it agrees to the PD's order, it sends the product order to the PM for production. During the PM's production, it needs an MS to provide information on the raw material of the product and the source of the raw material. Then, the production is completed and sent to the PD. The main players in this phase are the BP, PM, MS, PD, and CO, as shown in the flowchart in Figures 8–10:

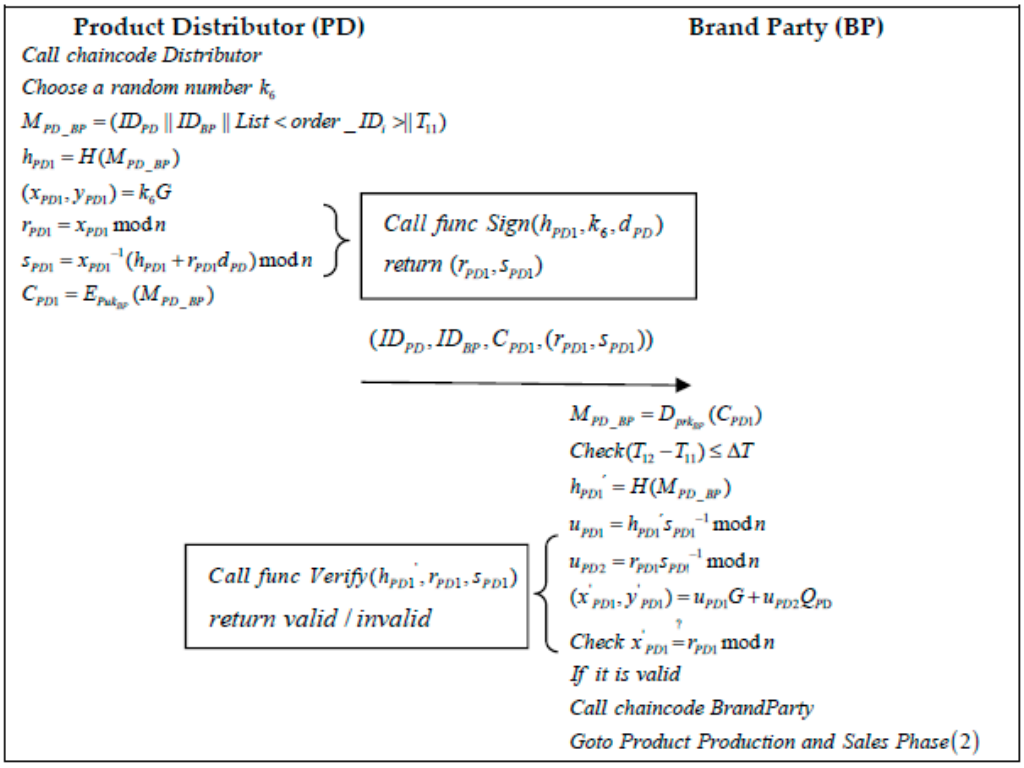

Figure 8. Product production and sales phase (1).

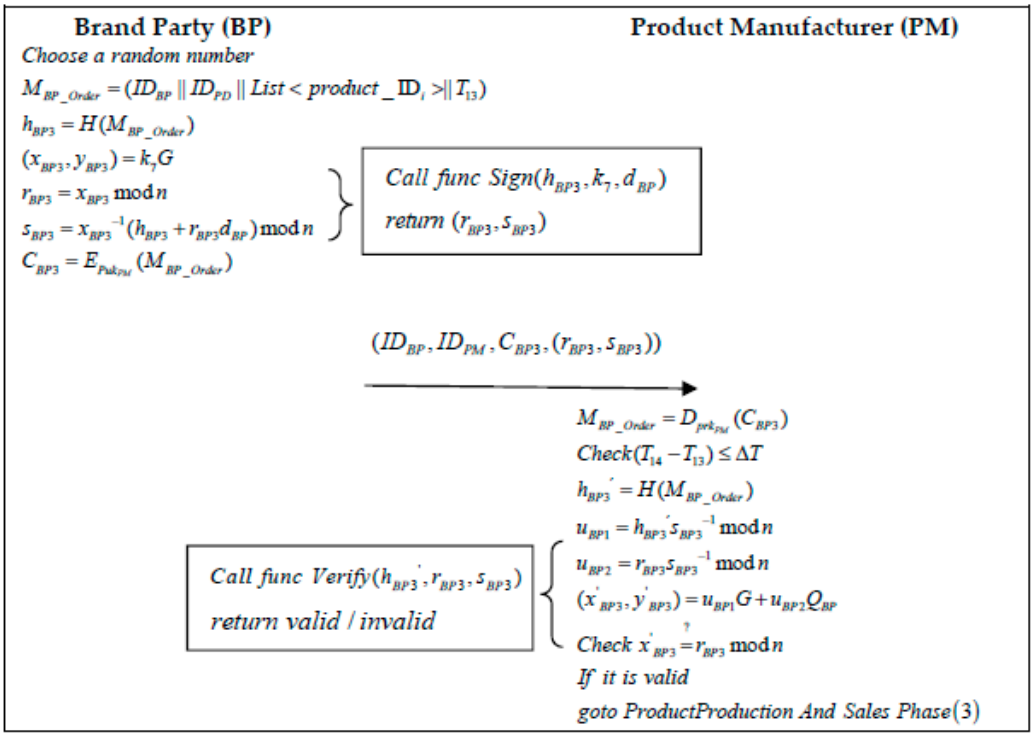

Figure 9. Product production and sales phase (2).

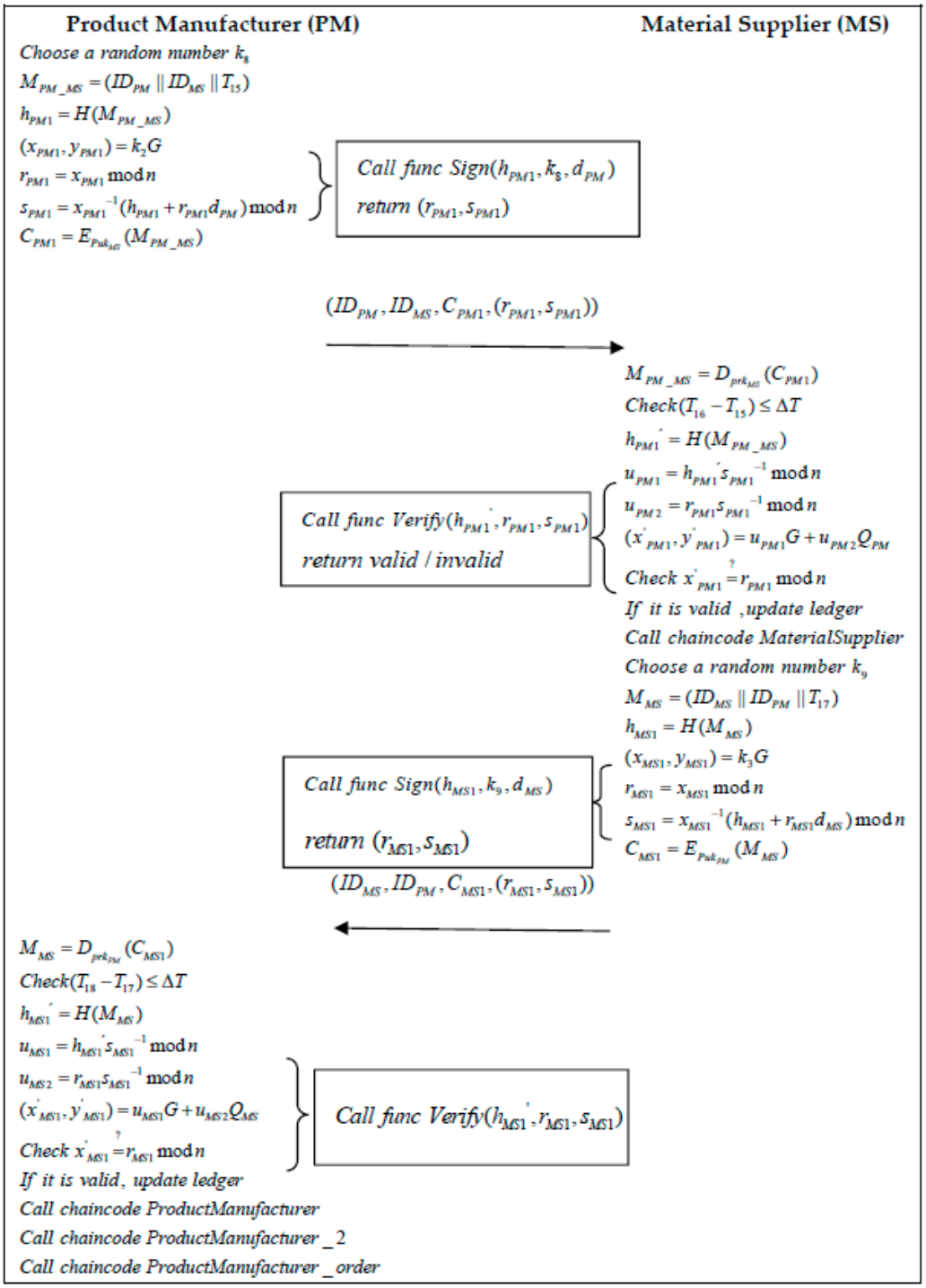

Figure 10. Product production and sales phase (3).

Step 1: When a PD wants to order a product, it submits an order request to the brand at moment T_{11} and generates an order for the product, executing the "Distributor" function, the algorithm of which is shown in Algorithm 5. This function signs the order contents and sends those to the BP. The PD selects a random number k_6 and generates the product order information M_{PD_BP}:

$$M_{PD\_BP} = (ID_{PD} || ID_{BP} || List < order\_\mathrm{ID}_i > || T_{11}) \quad (42)$$

The PD then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{PD1}, s_{PD1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{PD1} = H(M_{PD\_BP}) \tag{43}$$

$$(r_{PD1}, s_{PD1}) = Sign(h_{PD1}, k_6, d_{PD}) \tag{44}$$

The generated encrypted message is encrypted by the BP public key:

$$C_{PD1} = E_{Puk_{BP}}(M_{PD\_BP}) \tag{45}$$

The PD then sends the $(ID_{PD}, ID_{BP}, C_{PD1}, (r_{PD1}, s_{PD1}))$ information to the BP.

Step 2: When a BP receives a message from a PD at a moment T_{12}, it first decrypts the message using its private key:

$$M_{PD\_BP} = D_{prk_{BP}}(C_{PD1}) \tag{46}$$

Secondly, the BP verifies the validity of the timestamp:

$$Check(T_{12} - T_{11}) \leq \Delta T \tag{47}$$

Once the validity of the time has been verified, the brand uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{PD1}{}' = H(M_{PD\_BP}) \tag{48}$$

$$Verify(h_{PD1}{}', r_{PD1}, s_{PD1}) \tag{49}$$

If the signature verification is valid, the BP executes the "BrandParty" function at moment T_{13}; the algorithm of this function is shown in Algorithm 3. Then, the product order information is signed and sent to the PM, who chooses a random number k_7 and generates the product order information M_{BP_Order}.

$$M_{BP\_Order} = (ID_{BP}||ID_{PD}||List < product\_\mathrm{ID}_i >||T_{13}) \tag{50}$$

The BP then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{BP3}, s_{BP3}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{BP3} = H(M_{BP\_Order}) \tag{51}$$

$$(r_{BP3}, s_{BP3}) = Sign(h_{BP3}, k_7, d_{BP}) \tag{52}$$

The generated message is encrypted by the PM's public key:

$$C_{BP3} = E_{Puk_{PM}}(M_{BP\_Order}) \tag{53}$$

The BP then sends the $(ID_{BP}, ID_{PM}, C_{BP3}, (r_{BP3}, s_{BP3}))$ information to the PM.

Algorithm 5. Chain code of the product production and sales phase.

```
var PO []ProductOrder
func Distributor (ID_PD string, ID_BP string, kind string, Order_INF string, Order_IDs []string,
state string) {
for i:= 0; i < Order_IDs.Length; i++ {
index:= SearchOrder_ID(Order_IDs[i]);
   PO[index].Order_PD_idNumber = ID_PD
   PO[index]. Order_kind = kind
   PO[index].Order_brand_idNumber = ID_BP
   PO[index].Order_information = Order_INF
   PO[index].Order_createDate = time.Now()
   PO[index].Order_state = state
}
}
func ProductManufacture_order (ID_PM string, Order_IDs []string, state string) {
for i:= 0; i < Order_IDs.Length; i++ {
index:= SearchOrder_ID(Order_IDs[i]);
   PO[index].Order_PM_idNumber = ID_PM
   PO[index].Order_state = state
}
}
```

Step 3: When the PM receives a message from the BP at moment T_{14}, it first decrypts the message using its private key:

$$M_{BP\_Order} = D_{prk_{PM}}(C_{BP3}) \tag{54}$$

Secondly, the PM verifies the validity of the timestamp:

$$Check(T_{14} - T_{13}) \leq \Delta T \tag{55}$$

Once the validity of the time has been verified, the PM uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{BP3}{}' = H(M_{BP\_Order}) \tag{56}$$

$$Verify(h_{BP3}{}', r_{BP3}, s_{BP3}) \tag{57}$$

If the signature verification is valid, the PM signs the product material type and content requirements at moment T_{15} and delivers them to the MS. The PM chooses a random number k_8 and generates the product raw material requirement information M_{PM_MS}:

$$M_{PM\_MS} = (ID_{PM}||ID_{MS}||T_{15}) \tag{58}$$

The PM then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{PM1}, s_{PM1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{PM1} = H(M_{PM\_MS}) \tag{59}$$

$$(r_{PM1}, s_{PM1}) = Sign(h_{PM1}, k_2, d_{PM}) \tag{60}$$

The generated message is encrypted by the MS's public key:

$$C_{PM1} = E_{Puk_{MS}}(M_{PM\_MS}) \tag{61}$$

The PM then sends the $(ID_{PM}, ID_{MS}, C_{PM1}, (r_{PM1}, s_{PM1}))$ information to the MS.

Step 4: When the MS receives a message from the BP at a moment T_{16}, it first decrypts the message using its private key:

$$M_{PM\_MS} = D_{prk_{MS}}(C_{PM1}) \tag{62}$$

Secondly, the MS verifies the validity of the timestamp:

$$Check(T_{16} - T_{15}) \leq \Delta T \tag{63}$$

Once the validity of the time has been verified, the MS uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{PM1}{}' = H(M_{PM\_MS}) \tag{64}$$

$$Verify(h_{PM1}{}', r_{PM1}, s_{PM1}) \tag{65}$$

If the signature verification is valid, the MS executes the "Material Supplier" function at moment T_{17}. The algorithm of this function is shown in Algorithm 3 and provides the PM with the raw materials needed to produce the product according to the raw material requirements in its message, as well as the type, content, and source of the raw materials. The information is signed and given to the PM. The MS selects a random number k_9 and generates a message M_{MS}:

$$M_{MS} = (ID_{MS}||ID_{PM}||T_{17}) \tag{66}$$

The MS then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{MS1}, s_{MS1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{MS1} = H(M_{MS}) \tag{67}$$

$$(r_{MS1}, s_{MS1}) = Sign(h_{MS1}, k_9, d_{MS}) \tag{68}$$

The generated message is encrypted by the PM's public key:

$$C_{MS1} = E_{Puk_{PM}}(M_{MS}) \tag{69}$$

The MS then sends the $(ID_{MS}, ID_{PM}, C_{MS1}, (r_{MS1}, s_{MS1}))$ information to the PM.

Step 5: When the PM receives a message from the MS at moment T_{18}, it first decrypts the message using its private key:

$$M_{MS} = D_{prk_{PM}}(C_{MS1}) \tag{70}$$

Secondly, the PM verifies the validity of the timestamp:

$$Check(T_{18} - T_{17}) \leq \Delta T \tag{71}$$

Once the validity of the time has been verified, the MS uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{MS1}{}' = H(M_{MS}) \tag{72}$$

$$Verify(h_{MS1}{}', r_{MS1}, s_{MS1}) \tag{73}$$

If the signature verification is valid, the product ordered by the PD is manufactured, tested, and evaluated, then the "ProductManufacturer", "ProductManufacturer_2", and "ProductManufacturer_order" functions are executed with Algorithms 3, 4, and 5, respectively, and the local book is updated with the information to ship the product to the PD.

3.8. Product Purchase and Logistics Distribution Phase

In the product purchase and logistics delivery phase, when the CO has purchased the product, the PD will send the information related to the logistics order to the LP. The LP receives the relevant information to generate the logistics order, and the logistics information will be sent to the DM for delivery. The DM delivery may be updated with other DM information and product confirmations, or with the signatory for logistics information updates and confirmation of the product. The main participants in this phase are the PD, LP, DM, and CO; a flowchart for the phase is shown in Figures 11–14.

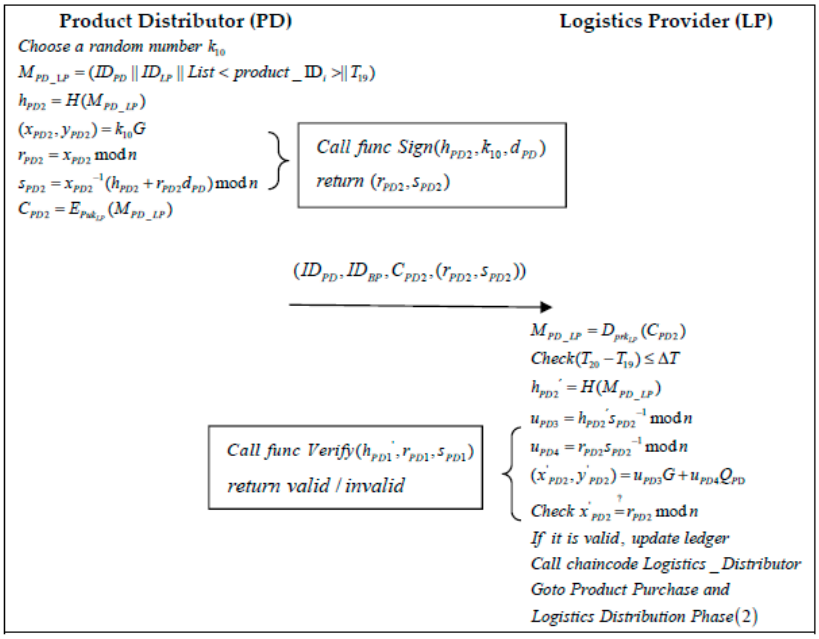

Figure 11. Product purchase and logistics distribution phase (1).

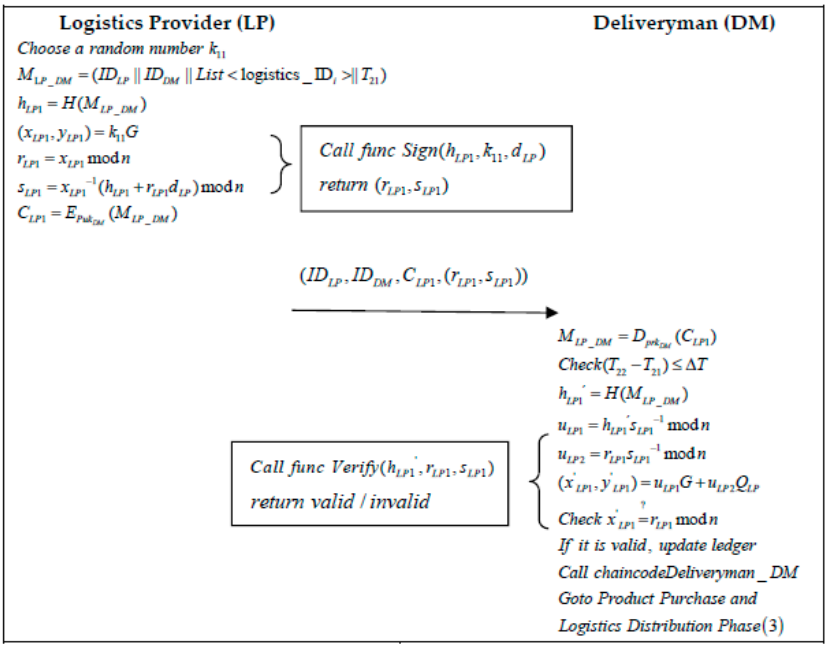

Figure 12. Product purchase and logistics distribution phase (2).

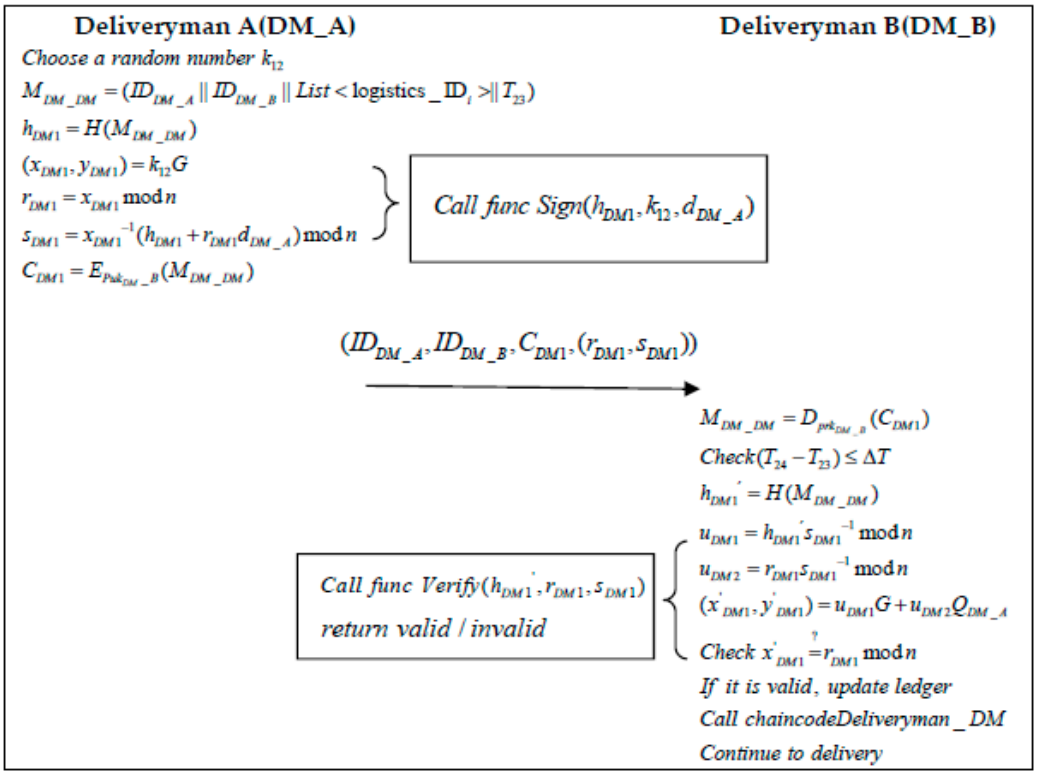

Figure 13. Product purchase and logistics distribution phase (3).

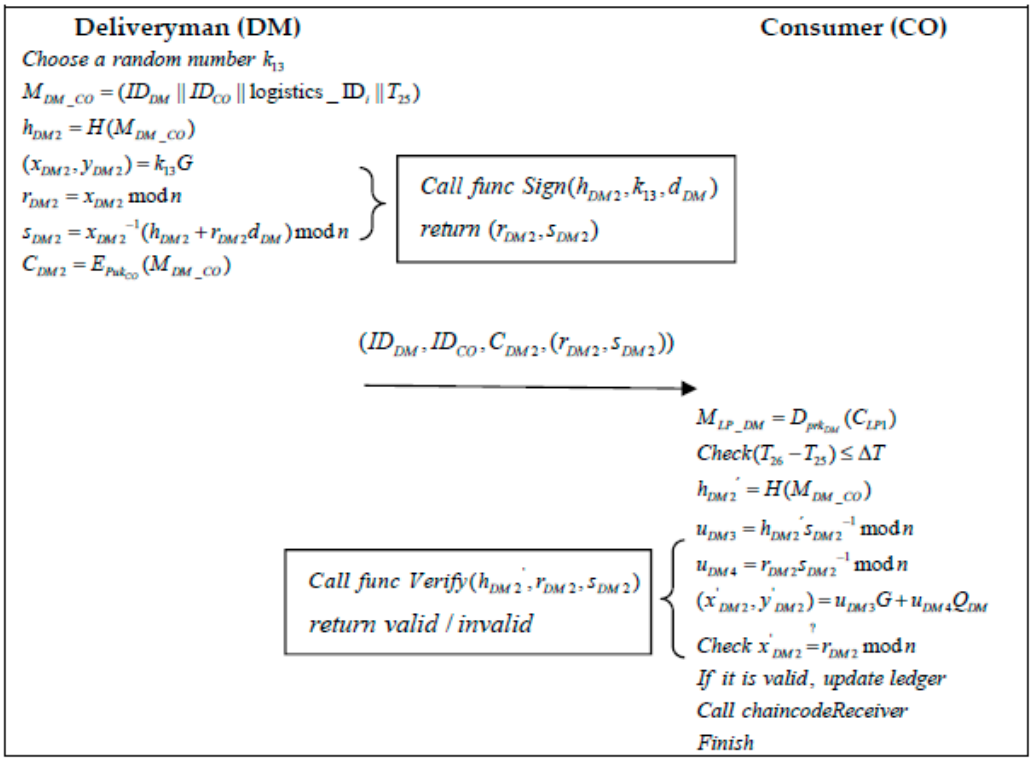

Figure 14. Product purchase and logistics distribution phase (4).

Step 1: When a CO wants to buy a product, they request that the PD buy it. The PD executes the "Distributor_sell" function, the algorithm of which is shown in Algorithm 6, and sends the PD UID, CO UID, and order type to the LP at moment T_{19}. The PD

randomly selects a number k_{10} and generates the information related to the logistics order M_{PD_LP}:

$$M_{PD\_LP} = (ID_{PD}||ID_{LP}||List < \text{product\_ID}_i >||T_{19}) \quad (74)$$

The PD then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{PD2}, s_{PD2}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{PD2} = H(M_{PD\_LP}) \quad (75)$$

$$(r_{PD2}, s_{PD2}) = Sign(h_{PD2}, k_{10}, d_{PD}) \quad (76)$$

The generated message is encrypted by the LP's public key:

$$C_{PD2} = E_{Puk_{LP}}(M_{PD\_LP}) \quad (77)$$

The PD then sends the $(ID_{PD}, ID_{BP}, C_{PD2}, (r_{PD2}, s_{PD2}))$ information to the LP.

Step 2: When an LP receives a message from a PD at a moment T_{20}, it first decrypts the message using its private key:

$$M_{PD\_LP} = D_{prk_{LP}}(C_{PD2}) \quad (78)$$

Secondly, the LP verifies the validity of the timestamp:

$$Check(T_{20} - T_{19}) \leq \Delta T \quad (79)$$

Once the validity of the time has been verified, the LP uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{PD2}{}' = H(M_{PD\_LP}) \quad (80)$$

$$Verify(h_{PD2}{}', r_{PD2}, s_{PD2}) \quad (81)$$

If the signature verification is valid, the LP generates the corresponding logistics order for each product at moment T_{21} and executes the "Logistics_Distributor" function, the algorithm of which is shown in Algorithm 6. At moment T_{21}, the logistics order is assigned to the DM, while the LP randomly selects a random number k_{11} and generates the logistics order UID information M_{LP_DM}:

$$M_{LP\_DM} = (ID_{LP}||ID_{DM}||List < l\text{ogistics\_ID}_i >||T_{21}) \quad (82)$$

The LP then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{LP1}, s_{LP1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{LP1} = H(M_{LP\_DM}) \quad (83)$$

$$(r_{LP1}, s_{LP1}) = Sign(h_{LP1}, k_{11}, d_{LP}) \quad (84)$$

The generated message is encrypted by the DM's public key:

$$C_{LP1} = E_{Puk_{DM}}(M_{LP\_DM}) \quad (85)$$

The LP then sends the $(ID_{LP}, ID_{DM}, C_{LP1}, (r_{LP1}, s_{LP1}))$ information to the DM.

Algorithm 6. Chain code of the product purchase and logistics distribution phase.

```
func Distributor_sell (ID_CO string, ID_PD string, Product_INF string, Product_IDs []string,
Signature string) {
for i:= 0; i < Product_IDs.Length; i++ {
index:= SearchProduct _ID(Product_IDs[i]);
   LP[index].Product_consumer_idNumber = ID_CO
   LP[index].Product_distributor_idNumber = ID_PD
   LP[index].Product_information = Product_INF
   LP[index].PD_Signature = Signature
}
}
var LO []LogisticsOrder
func Logistics_Distributor (ID_PD string, ID_CO string, ID_P []string, kind string, Product_IDs
[]string, state string, Signature string) {
for i:= 0; i < Product_IDs.Length; i++ {
LO[i].Logistics_idNumber = Generate_Logistics_id()
   LO[i].Logistics_distributor_idNumber = ID_PD
   LO[i].Logistics_company_idNumber = ID_LP
   LO[i].Logistics_consumer_idNumber = ID_CO
   LO[i].Logistics_product_idNumber = Product_IDs [i]
   LO[i].Logistics_kind = kind
   LO[i].Logistics_createDate = time.Now()
   LO[i].Logistics_state = state
}
for i:= 0; i < Product_IDs.Length; i++ {
index_P:= SearchProduct_ID(Product_IDs[i]);
index_L:= SearchLogistics_ID(Product_IDs[i]);
   LP[index_P].Product_courier_idNumber = LO[index_L].Logistics_idNumber
   LP[index_P].LP_Signature = Signature
}
}
```

Step 3: When a DM receives a message from an LP at moment T_{22}, it first decrypts the message using its private key:

$$M_{LP\_DM} = D_{prk_{DM}}(C_{LP1}) \tag{86}$$

Secondly, the DM verifies the validity of the timestamp:

$$Check(T_{22} - T_{21}) \leq \Delta T \tag{87}$$

Once the validity of the time has been verified, the DM uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{LP1}{}' = H(M_{LP\_DM}) \tag{88}$$

$$Verify(h_{LP1}{}', r_{LP1}, s_{LP1}) \tag{89}$$

If the signature verification is valid, the DM executes the "Deliveryman_DM" function, the algorithm of which is shown in Algorithm 7. The product is then delivered.

Algorithm 7. Chain code of the product purchase and logistics distribution phase.

```
DM_Num = 0
func Deliveryman_DM(ID_DM_1 string, ID_DM_2 string, Location string, Logistics_IDs []string,
state string, Signature string) {
for i:= 0; i < Logistics_IDs.Length; i++ {
index_L:= SearchLogistics_ID(Logistics_IDs[i]);
index_P:= SearchLogistics_Product_ID(Logistics_IDs[i])
   LO[index_L].Logistics_inf[DM_Num].Shipper_idNumber = ID_DM_1
   LO[index_L].Logistics_inf[DM_Num].Recipient_idNumber = ID_DM_2
   LO[index_L].Logistics_inf[DM_Num].Recipient_Location = Location
   LO[index_L].Logistics_inf[DM_Num].Recipient_time = time.Now()
   LO[index_L].Logistics_state = state
   LP[index_P].DM_Signature[DM_num] = Signature
}
DM_Num += 1
}
func Receiver (ID_DM string,ID_CO string, Location string, Logistics_ID string, state string,
Signature string) {
index_L:= SearchLogistics_ID(Logistics_ID);
index_P:= SearchLogistics_Product_ID(Logistics_IDs[i])
   LO[index_L].Logistics_inf[DM_Num].Shipper_idNumber = ID_DM
   LO[index_L].Logistics_inf[DM_Num].Recipient_idNumber = ID_CO
   LO[index_L].Logistics_inf[DM_Num].Recipient_Location = Location
   LO[index_L].Logistics_inf[DM_Num].Recipient_time = time.Now()
   LO[index_L].Logistics_finishDate = time.Now()
LO[index_L].Logistics_state = state
   LP[index_P].CO_Signature = Signature
}
```

Step 4: The distribution handover process of the products takes one of two kinds:

(a) When DM A delivers the product to DM B at moment T_{23}.

Step 1: DM_A randomly selects a number k_{12} and generates the information related to the logistics order $M_{\text{DM\_DM}}$:

$$M_{DM\_DM} = (ID_{DM\_1}||ID_{DM\_2}||List < \text{logistics\_ID}_i >||T_{23}) \quad (90)$$

DM_A then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{DM1}, s_{DM1}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{DM1} = H(M_{DM\_DM}) \quad (91)$$

$$(r_{DM1}, s_{DM1}) = Sign(h_{DM1}, k_{12}, d_{DM\_A}) \quad (92)$$

The generated message is encrypted by DM_B's public key:

$$C_{DM1} = E_{Puk_{DM\_B}}(M_{DM\_DM}) \quad (93)$$

DM_A then sends the $(ID_{DM\_A}, ID_{DM\_B}, C_{DM1}, (r_{DM1}, s_{DM1}))$ information to DM_B.

Step 2: When DM_B receives a message from DM_A at a moment T_{24}, it first decrypts the message using its private key:

$$M_{DM\_DM} = D_{prk_{DM\_B}}(C_{DM1}) \quad (94)$$

Secondly, DM_B verifies the validity of the timestamp:

$$Check(T_{24} - T_{23}) \leq \Delta T \tag{95}$$

Once the validity of the time has been verified, DM_B uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{DM1}{}' = H(M_{DM\_DM}) \tag{96}$$

$$Verify(h_{DM1}{}', r_{DM1}, s_{DM1}) \tag{97}$$

If the signature verification is valid, DM_B executes the "Deliveryman_DM" function using Algorithm 7 and proceeds with the delivery of the product.

(b) When the DM hands over the product he has delivered to the CO at moment T_{25}.

Step 1: The DM randomly selects a number k_{13} and generates the information related to the logistics order $M_{\text{DM\_CO}}$:

$$M_{\text{DM\_CO}} = (ID_{DM}||ID_{CO}||\text{logistics\_ID}_i||T_{25}) \tag{98}$$

The DM then calculates the message hash and executes the "Sign" algorithm to generate the signature (r_{DM2}, s_{DM2}). The "Sign" algorithm is shown in Algorithm 2:

$$h_{DM2} = H(M_{DM\_CO}) \tag{99}$$

$$(r_{DM2}, s_{DM2}) = Sign(h_{DM2}, k_{13}, d_{DM}) \tag{100}$$

The generated message is encrypted by the CO public key:

$$C_{DM2} = E_{Puk_{CO}}(M_{DM\_CO}) \tag{101}$$

The DM then sends the $(ID_{DM}, ID_{CO}, C_{DM2}, (r_{DM2}, s_{DM2}))$ information to the CO.

Step 2: When a CO receives a message from a DM at moment T_{26}, it first decrypts the message using its private key:

$$M_{DM\_CO} = D_{prk_{CO}}(C_{DM2}) \tag{102}$$

Secondly, the CO verifies the validity of the timestamp:

$$Check(T_{26} - T_{25}) \leq \Delta T \tag{103}$$

Once the validity of the time has been verified, the CO uses the "Verify" function in Algorithm 2 to calculate a hash value to verify the message:

$$h_{DM2}{}' = H(M_{DM\_CO}) \tag{104}$$

$$Verify(h_{DM2}{}', r_{DM2}, s_{DM2}) \tag{105}$$

If the signature verification is valid, the CO executes the "Receiver" function, the algorithm of which is shown in Algorithm 7, and updates the local ledger with the information to complete the product purchase and logistic distribution phase.

4. Analysis

In this study, we have carried out system characterization and also important security analysis to propose solutions to the problems of system vulnerabilities and system attacks.

4.1. Dispersive and Transparent

In this system, we have built a Hyperledger Fabric-based blockchain network. The roles in the system correspond to organizations and peer nodes in the consortium chain. These organizations and peer nodes must first register with the blockchain center and be approved by a certification authority before they can join the blockchain network and the corresponding channels. Where peer nodes within the same organization can trust each other, trust between different organizations is achieved by the certification of the Certificate Authority. Entities can join the same channel to share open and transparent information, as well as be able to segregate information through the channel. With this model, we have created a decentralized, transparent system where multiple organizations may trust one another.

4.2. Unforgeable, Traceable Data

First, we analyze the system's forgery prevention and traceability. In this system, we use Hyperledger Fabric-based blockchain technology, which raises the difficulty of forging the data stored in the blockchain compared to traditional database systems. At each stage of the system design, all participants must update the relevant data to the blockchain center via a chaincode. When a participant calls a function in the chaincode, the Hyperledger Fabric mechanism presents it to all peer nodes in the chain, and when the transaction is verified, each peer node signs and responds to the transaction. Afterward, the ledger of each peer node will be updated by sorting nodes. The above mechanism ensures that the data at the center of the blockchain cannot be falsified since the transactions and records stored in the ledger of all peer nodes in the blockchain are duplicated and the information in the ledger can only be updated through pre-written chain codes. In addition, each transaction record on the blockchain is stored as a "chain" in the ledger of each peer node; these records can be traced through the ledger to achieve the goal of traceability.

4.3. Data Integrity

Secondly, we analyze the integrity of the data. In this system, the ECDSA signature algorithm is used to ensure the integrity of messages passed between roles. When a role needs to transmit data to another role, a calculation must be performed on the data to be transmitted. A hash value is calculated, a set of signatures is generated, and the hash value is transmitted to the recipient along with the signature and the message. Once the receiver receives the message, they need to verify the validity of the message in terms of the hash and signature by using the "Verify" function in Algorithm 2.

For example, in the product design and development phase, the sender BP sends the message M_{BP_DES} to the receiver PM. The BP needs to generate h_{BP1}. The "Sign" function in Algorithm 2 is used to calculate (r_{BP1}, s_{BP1}). The BP then sends the message M_{BP_DES} with a signature (r_{BP1}, s_{BP1}) to the PM, who receives the message and decrypts it, generating $h_{BP1}{}'$ based on the message M_{BP_DES}. The "Verify" function in Algorithm 2 is used to verify the hash $h_{BP1}{}'$ with the signature (r_{BP1}, s_{BP1}). Table 3 shows all the details of each phase.

Table 3. Verification of the data integrity with the proposed scheme.

Phase	Party		Message	Hash Value	Verification
	Sender	Receiver			
Product Design and Development Phase	BP	PM	$M_{BP\_DES} = (ID_{BP}\|\|ID_{PM}\|\|List < product\_\text{ID}_i >\|\|T_1)$	$h_{BP1} = H(M_{BP\_DES})$	$Verify(h_{BP1}{}', r_{BP1}, s_{BP1})$
	PM	MS	$M_{PM\_MS} = (ID_{PM}\|\|ID_{MS}\|\|T_3)$	$h_{PM1} = H(M_{PM\_MS})$	$Verify(h_{PM1}{}', r_{PM1}, s_{PM1})$
	MS	PM	$M_{MS} = (ID_{MS}\|\|ID_{PM}\|\|T_5)$	$h_{MS1} = H(M_{MS})$	$Verify(h_{MS1}{}', r_{MS1}, s_{MS1})$
Product Evaluation and Approval Phase	PM	BP	$M_{PM\_BP} = (ID_{PM}\|\|ID_{BP}\|\|T_7)$	$h_{PM2} = H(M_{PM\_BP})$	$Verify(h_{PM2}{}', r_{PM2}, s_{PM2})$
	BP	PM	$M_{BP\_PM} = (ID_{MS}\|\|ID_{PM}\|\|T_9)$	$h_{BP2} = H(M_{BP\_PM})$	$Verify(h_{BP2}{}', r_{BP2}, s_{BP2})$
Product Ordering and Production Phase	PD	BP	$M_{PD\_BP} = (ID_{PD}\|\|ID_{BP}\|\|List < order\_\text{ID}_i >\|\|T_{11})$	$h_{PD1} = H(M_{PD\_BP})$	$Verify(h_{PD1}{}', r_{PD1}, s_{PD1})$
	BP	PM	$M_{BP\_Order} = (ID_{BP}\|\|ID_{PD}\|\|List < product\_\text{ID}_i >\|\|T_{13})$	$h_{BP3} = H(M_{BP\_Order})$	$Verify(h_{BP3}{}', r_{BP3}, s_{BP3})$
	PM	MS	$M_{PM\_MS} = (ID_{PM}\|\|ID_{MS}\|\|T_{15})$	$h_{PM1} = H(M_{PM\_MS})$	$Verify(h_{PM1}{}', r_{PM1}, s_{PM1})$
	MS	PM	$M_{MS} = (ID_{MS}\|\|ID_{PM}\|\|T_{17})$	$h_{MS1} = H(M_{MS})$	$Verify(h_{MS1}{}', r_{MS1}, s_{MS1})$
Product Purchase and Logistics Distribution Phase	PD	LP	$M_{PD\_LP} = (ID_{PD}\|\|ID_{LP}\|\|List < product\_\text{ID}_i >\|\|T_{19})$	$h_{PD2} = H(M_{PD\_LP})$	$Verify(h_{PD2}{}', r_{PD2}, s_{PD2})$
	LP	DM	$M_{LP\_DM} = (ID_{LP}\|\|ID_{DM}\|\|List < \text{logistics\_ID}_i >\|\|T_{21})$	$h_{LP1} = H(M_{LP\_DM})$	$Verify(h_{LP1}{}', r_{LP1}, s_{LP1})$
	DM_A	DM_B	$M_{DM\_DM} = (ID_{DM\_A}\|\|ID_{DM\_B}\|\|List < \text{logistics\_ID}_i >\|\|T_{23})$	$h_{DM1} = H(M_{DM\_DM})$	$Verify(h_{DM1}{}', r_{DM1}, s_{DM1})$
	DM	CO	$M_{DM\_CO} = (ID_{DM}\|\|ID_{CO}\|\|\text{logistics\_ID}_i\|\|T_{25})$	$h_{DM2} = H(M_{DM\_CO})$	$Verify(h_{DM2}{}', r_{DM2}, s_{DM2})$

4.4. Non-Repudiation

Next, we analyze the non-repudiation of the message. We use the ECDSA signature algorithm to ensure that the message comes from the correct sender. The sender needs to generate a signature based on the message before sending it, and the receiver verifies the signature by using the "Verify" function in Algorithm 2 after receiving the message. For example, in the product design and development phase, the sender uses the ECDSA "Sign" function in Algorithm 2 to generate a signature (r_{BP1}, s_{BP1}) for a random number k_1, a hash value h_{BP1}, and an ECDSA parameter d_{BP}, which is sent to the receiver. After receiving the message, the receiver calculates a hash value $h_{BP1}{}'$ for the message and then verifies the $h_{BP1}{}'$ and (r_{BP1}, s_{BP1}) signatures using the "Verify" function in Algorithm 2, which proves that the message has not been tampered with if it is correct. Table 4 lists all the signatures and verifications for each phase.

Table 4. Verifying the non-repudiation of the proposed scheme.

Phase	Party		Signature	Verification
	Sender	Receiver		
Product Design and Development Phase	BP	PM	$(r_{BP1}, s_{BP1}) = Sign(h_{BP1}, k_1, d_{BP})$	$Verify(h_{BP1}{}', r_{BP1}, s_{BP1})$
	PM	MS	$(r_{PM1}, s_{PM1}) = Sign(h_{PM1}, k_2, d_{PM})$	$Verify(h_{PM1}{}', r_{PM1}, s_{PM1})$
	MS	PM	$(r_{MS1}, s_{MS1}) = Sign(h_{MS1}, k_3, d_{MS})$	$Verify(h_{MS1}{}', r_{MS1}, s_{MS1})$
Product Evaluation and Approval Phase	PM	BP	$(r_{PM2}, s_{PM2}) = Sign(h_{PM2}, k_4, d_{PM})$	$Verify(h_{PM2}{}', r_{PM2}, s_{PM2})$
	BP	PM	$(r_{BP2}, s_{BP2}) = Sign(h_{BP2}, k_5, d_{BP})$	$Verify(h_{BP2}{}', r_{BP2}, s_{BP2})$
Product Ordering and Production Phase	PD	BP	$(r_{PD1}, s_{PD1}) = Sign(h_{PD1}, k_6, d_{PD})$	$Verify(h_{PD1}{}', r_{PD1}, s_{PD1})$
	BP	PM	$(r_{BP3}, s_{BP3}) = Sign(h_{BP3}, k_7, d_{BP})$	$Verify(h_{BP3}{}', r_{BP3}, s_{BP3})$
	PM	MS	$(r_{PM1}, s_{PM1}) = Sign(h_{PM1}, k_2, d_{PM})$	$Verify(h_{PM1}{}', r_{PM1}, s_{PM1})$
	MS	PM	$(r_{MS1}, s_{MS1}) = Sign(h_{MS1}, k_9, d_{MS})$	$Verify(h_{MS1}{}', r_{MS1}, s_{MS1})$
Product Purchase and Logistics Distribution Phase	PD	LP	$(r_{PD2}, s_{PD2}) = Sign(h_{PD2}, k_{10}, d_{PD})$	$Verify(h_{PD2}{}', r_{PD2}, s_{PD2})$
	LP	DM	$(r_{LP1}, s_{LP1}) = Sign(h_{LP1}, k_{11}, d_{LP})$	$Verify(h_{LP1}{}', r_{LP1}, s_{LP1})$
	DM_A	DM_B	$(r_{DM1}, s_{DM1}) = Sign(h_{DM1}, k_{12}, d_{DM\_A})$	$Verify(h_{DM1}{}', r_{DM1}, s_{DM1})$
	DM	CO	$(r_{DM2}, s_{DM2}) = Sign(h_{DM2}, k_{13}, d_{DM})$	$Verify(h_{DM2}{}', r_{DM2}, s_{DM2})$

4.5. Man-in-the-Middle Attack

For a man-in-the-middle attack, the system uses asymmetric encryption for defense. When the sender sends a message to the receiver, the message must be encrypted using the receiver's public key. When the receiver receives the message, they decrypt it with their private key to obtain the message to be transmitted. In our system, both communicating parties have access to each other's public keys in the blockchain network, meaning they do not need to send their public keys to each other. This prevents an attacker from intercepting the message and replacing the public key. So, even though an attacker may intercept the

message, they do not know the receiver's private key, so they cannot decrypt the message. Table 5 lists all the asymmetric encryptions and decryptions at each phase.

Table 5. Encryption and decryption to prevent a man-in-the-middle attack.

Phase	Party		Encryption	Decryption
	Sender	Receiver		
Product Design and Development Phase	BP	PM	$C_{BP1} = E_{Puk_{PM}}(M_{BP\_DES})$	$M_{BP\_DES} = D_{prk_{PM}}(C_{BC1})$
	PM	MS	$C_{PM1} = E_{Puk_{MS}}(M_{PM\_MS})$	$M_{PM\_MS} = D_{prk_{MS}}(C_{PM1})$
	MS	PM	$C_{MS1} = E_{Puk_{PM}}(M_{MS})$	$M_{MS} = D_{prk_{PM}}(C_{MS1})$
Product Evaluation and Approval Phase	PM	BP	$C_{PM2} = E_{Puk_{BP}}(M_{PM\_BP})$	$M_{PM\_BP} = D_{prk_{BP}}(C_{PM2})$
	BP	PM	$C_{BP2} = E_{Puk_{PM}}(M_{BP\_PM})$	$M_{BP\_PM} = D_{prk_{PM}}(C_{BP2})$
Product Ordering and Production Phase	PD	BP	$C_{PD1} = E_{Puk_{BP}}(M_{PD\_BP})$	$M_{PD\_BP} = D_{prk_{BP}}(C_{PD1})$
	BP	PM	$C_{BP3} = E_{Puk_{PM}}(M_{BP\_Order})$	$M_{BP\_Order} = D_{prk_{PM}}(C_{BP3})$
	PM	MS	$C_{PM1} = E_{Puk_{MS}}(M_{PM\_MS})$	$M_{PM\_MS} = D_{prk_{MS}}(C_{PM1})$
	MS	PM	$C_{MS1} = E_{Puk_{PM}}(M_{MS})$	$M_{MS} = D_{prk_{PM}}(C_{MS1})$
Product Purchase and Logistics Distribution Phase	PD	LP	$C_{PD2} = E_{Puk_{LP}}(M_{PD\_LP})$	$M_{PD\_LP} = D_{prk_{LP}}(C_{PD2})$
	LP	DM	$C_{LP1} = E_{Puk_{DM}}(M_{LP\_DM})$	$M_{LP\_DM} = D_{prk_{DM}}(C_{LP1})$
	DM_A	DM_B	$C_{DM1} = E_{Puk_{DM\_B}}(M_{DM\_DM})$	$M_{DM\_DM} = D_{prk_{DM\_B}}(C_{DM1})$
	DM	CO	$C_{DM2} = E_{Puk_{CO}}(M_{DM\_CO})$	$M_{DM\_CO} = D_{prk_{CO}}(C_{DM2})$

4.6. Replay Attack

During communication between two parties, a message may be captured by an attacker, who then pretends to be a legitimate sender and sends the same message to the recipient several times over. For this attack case, this system uses a mechanism of adding a timestamp between the two communicating parties for defense. The receiver needs to calculate the difference in the timestamp after receiving the message, and if the difference exceeds a threshold value, it means that a replay attack has been launched. Table 6 lists all the timestamp verifications in each phase.

Table 6. Timestamp validation to prevent replay attack.

Phase	Party		Sent Time	Received Time	Validation
	Sender	Receiver			
Product Design and Development Phase	BP	PM	T_1	T_2	$Check(T_2 - T_1) \leq \Delta T$
	PM	MS	T_3	T_4	$Check(T_4 - T_3) \leq \Delta T$
	MS	PM	T_5	T_6	$Check(T_6 - T_5) \leq \Delta T$
Product Evaluation and Approval Phase	PM	BP	T_7	T_8	$Check(T_8 - T_7) \leq \Delta T$
	BP	PM	T_9	T_{10}	$Check(T_{10} - T_9) \leq \Delta T$
Product Ordering and Production Phase	PD	BP	T_{11}	T_{12}	$Check(T_{12} - T_{11}) \leq \Delta T$
	BP	PM	T_{13}	T_{14}	$Check(T_{14} - T_{13}) \leq \Delta T$
	PM	MS	T_{15}	T_{16}	$Check(T_{16} - T_{15}) \leq \Delta T$
	MS	PM	T_{17}	T_{18}	$Check(T_{18} - T_{17}) \leq \Delta T$
Product Purchase and Logistics Distribution Phase	PD	LP	T_{19}	T_{20}	$Check(T_{20} - T_{19}) \leq \Delta T$
	LP	DM	T_{21}	T_{22}	$Check(T_{22} - T_{21}) \leq \Delta T$
	DM_A	DM_B	T_{23}	T_{24}	$Check(T_{24} - T_{23}) \leq \Delta T$
	DM	CO	T_{25}	T_{26}	$Check(T_{26} - T_{25}) \leq \Delta T$

5. Discussion

We tested the blockchain service's performance through experimental simulations. The experimental simulations of the described scenario are shown in Table 7.

Table 7. Experimental environment's configuration.

Configuration	Detail
CPU	Intel(R) Core(TM) i5-8300H CPU@2.30 GHz
Memory	8 G
Network	4 Gbit/s
SSD	60 GB

5.1. Throughput and Latency of Smart Contract Calling

Caliper is a blockchain performance-testing framework that allows users to test different blockchain solutions using customer use cases to obtain a set of performance test results. In this scenario, we used Caliper to test the performance of the chaincode in four phases, and the results are shown in Table 8.

Table 8. Summary of performance metrics.

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
test Product Design and Development Phase_100	492	1	98.5	2.70	0.28	0.82	64.8
test Product Evaluation and Approval Phase_100	983	1	192.3	7.27	0.74	4.00	95.6
test Product Ordering and Production Phase_100	976	1	195.4	7.79	0.34	4.03	96.2
test Product Purchase and Logistics Distribution Phase_100	994	0	197.9	8.45	0.20	3.10	102.6

We used the throughput and transaction latency as the key performance metrics in our benchmarking. Throughput is the rate at which transactions are committed to the ledger, measured in terms of the number of transactions executed per second (tps). Latency is the time it takes from the time the application sends a transaction proposal to the time the transaction is committed to the ledger. As can be seen in Table 8, the phases not only have a high success rate but also maintain an average latency of around 4 s, with a throughput rate of 95 TPS for each phase for the same transactions.

5.2. Resource Utilization

In Caliper, we also tested the utilization of the system. In the simulation experiments during the product design and development phase, we set up two organization nodes, each of which consisted of a peer node. At the same time, we set the order node. The resource utilization for the product design and development phase is shown in Table 9.

5.3. Computation Cost

In this section, we analyze the computational costs for each role in each phase of the study. We use asymmetric encryption/decryption, the ECDSA signature and verification functions, hashing operations, symmetric encryption operations, and multiplication and division operations as the basis for calculating the costs. The costs for each stage are shown in Table 10.

Table 9. Resource utilization for a test of the product design and development phase.

Name	CPU% (max)	CPU% (avg)	Memory(max) [MB]	Memory(avg) [MB]	Traffic In [MB]	Traffic Out [MB]	Disc Wirte [MB]	Disc Read [MB]
dev-peer0.org1.example.com-alcohol-supply-contract_1.0-035019925cd56f13a2148f77f6a81fe7d139997b666c1264978d0b99d97bd8d0	6.80	1.83	17.1	17.1	1.32	0.566	0.00	0.00
dev-peer0.org2.example.com-alcohol-supply-contract_1.0-035019925cd56f13a2148f77f6a81fe7d139997b666c1264978d0b99d97bd8d0	5.79	1.61	17.0	17.0	1.29	0.527	0.00	0.00
cli	0.00	0.00	15.2	15.2	0.00	0.00	0.00	0.00
Peer0.org1.example.com	24.06	9.61	123	122	4.04	2.36	3.89	0.00
orderer.example.com	12.75	4.48	110	109	2.57	5.13	5.53	0.00
Peer0.org2.example.com	22.40	8.88	146	145	3.98	2.24	3.89	0.00

Table 10. Computation costs of the proposed scheme.

Phase	Participant 1	Participant 2	Participant 3	Participant 4
Product Design and Development Phase	BP: $T_{hash} + T_{E/D} + T_{sig} + T_{fun}$	PM: $2T_{ver} + T_{sig} + 3T_{hash} + 2T_{cmp} + 3T_{E/D} + T_{fun} + T_{upload}$	MS: $T_{ver} + T_{sig} + 2T_{hash} + T_{cmp} + 2T_{E/D} + T_{upload} + T_{fun}$	N/A
Product Evaluation and Approval Phase	PM: $2T_{hash} + 2T_{E/D} + T_{sig} + T_{fun} + T_{upload} + T_{cmp} + T_{ver}$	BP: $2T_{hash} + 2T_{E/D} + T_{sig} + T_{fun} + T_{upload} + T_{cmp} + T_{ver}$	N/A	N/A
Product Ordering and Production Phase	PD: $T_{hash} + T_{E/D} + T_{sig}$	BP: $2T_{hash} + 2T_{E/D} + T_{sig} + T_{fun} + T_{cmp} + T_{ver}$	PM: $3T_{hash} + 3T_{E/D} + T_{sig} + 3T_{fun} + 2T_{cmp} + 2T_{ver} + T_{upload}$	MS: $2T_{hash} + 2T_{E/D} + T_{sig} + T_{cmp} + T_{ver}$
Product Purchase and Logistics Distribution Phase	PD: $T_{hash} + T_{E/D} + T_{sig}$	LP: $T_{ver} + T_{sig} + 2T_{hash} + T_{cmp} + 2T_{E/D} + T_{fun} + T_{upload}$	DM: $nT_{ver} + (n+1)T_{sig} + (2n+1)T_{hash} + nT_{cmp} + (2n+1)T_{E/D} + nT_{fun} + nT_{upload}$	CO: $T_{hash} + T_{E/D} + T_{cmp} + T_{ver} + T_{fun} + T_{upload}$

T_{sig}: Signature operation; T_{ver}: verify operation; $T_{E/D}$: encryption/decryption operation; T_{hash}: hash function operation; T_{cmp}: comparison operation; T_{fun}: call chaincode function; T_{upload}: upload data operation.

5.4. Communication Costs

In this section, we analyze the communication costs for each phase of the proposed scheme. For example, in the product design and development phase, the BP, PM, and MS communication data volume constitutes six ID messages, three asymmetric encryptions, and three signature messages. The communication cost for each phase is shown in Table 11.

Table 11. Communication costs of the proposed scheme.

Phase	Message Length	3.5 G (14 Mpbs)	4 G (100 Mpbs)	5 G (20 Gpbs)
Product Design and Development Phase	4032 bits	0.288 ms	0.040 ms	0.202 µs
Product Evaluation and Approval Phase	2688 bits	0.192 ms	0.027 ms	0.134 µs
Product Ordering and Production Phase	5376 bits	0.384 ms	0.054 ms	0.269 µs
Product Purchase and Logistics Distribution Phase	5376 bits	0.384 ms	0.054 ms	0.269 µs

5.5. Function Comparison

In this section, we compare some of the systems mentioned in the Related Work section regarding product anti-counterfeiting, as shown in Table 12.

Table 12. Comparison of product security systems.

Authors	Year	Objective	1	2	3	4	5	6
Dan et al. [2]	2012	Proposed a luxury anti-counterfeiting system architecture and hierarchy based on the EPC Internet of Things (IoT)	N	Y	Y	Y	N	Y
Hochholdinger et al. [15]	2019	Marks or traces, from a forensic intelligence perspective, can achieve a watch anti-counterfeiting effect	N	N	N	N	N	Y
Perez et al. [16]	2020	Introduced the latest traceability program and recommended a framework for garments	Y	Y	N	N	Y	Y
Agrawal et al. [17]	2021	Investigated and proposed a blockchain-based traceability framework for traceability in a multi-tier T&C supply chain	Y	N	Y	N	Y	Y
Our proposed method	2022	Proposed a blockchain-based framework for product traceability and documentation of logistics processes	Y	Y	Y	Y	Y	Y

Notes: 1: Blockchain-based architecture, 2: Internet of Things (IoT), 3: complete architecture or framework, 4: security analysis, 5: unforgeable, 6: traceable, Y: yes, N: no.

6. Conclusions

To combat the proliferation of counterfeit luxury products, protect the rights of consumers, and maintain the huge consumer market for luxury products, we propose a blockchain-based anti-counterfeiting management system for traceable luxury products. A consortium blockchain is built through Hyperledger Fabric to deploy and execute smart contracts. The information related to the production of raw materials, producers, consumers, product flow, and logistics of luxury products will all be uploaded to the chain. Combined with the centralized, tamper-evidencing, and traceable features of the blockchain, the system can achieve decentralized storage of data, thus ensuring that it does not rely on other regulatory bodies and hardware facilities to store the information on the chain. Its lack of such a need makes it a complete, traceable, and credible record. The system also makes use of the characteristics of smart contracts to strictly enforce the pre-agreed rules without human intervention, and can transparently disclose the current logistics flow of products in real-time according to the execution of smart contracts. To build a more secure system, we applied ECDSA to the communication protocol and analyzed the security of the system in terms of data integrity and evidence of tampering, distributed and member access, information transparency, and traceability. In addition, we demonstrated that the protocol is resistant to man-in-the-middle and replay attacks. The proposed scheme was then discussed in terms of both computational cost and communication cost and compared with other schemes. In summary, we made the following contributions:

(1). Anti-counterfeit traceability management of luxury products using Hyperledger Fabric technology;
(2). ECDSA signature algorithm used to ensure data integrity;
(3). Smart contracts designed in the process of ordering, production, sales, and logistics of luxury products, and relevant information updated in real-time;
(4). Calculation and communication cost analysis;

(5). Consumer or third-party verification of information about the product through the blockchain.

Author Contributions: Conceptualization, C.-L.C. and L.-H.G.; methodology, C.-L.C., L.-H.G. and M.Z.; validation, W.Z., W.-J.T., H.S., C.-T.L. and Y.-Y.D.; investigation, C.-L.C. and L.-H.G.; data analysis, W.Z., W.-J.T., Y.-Y.D. and H.S.; writing—original draft preparation, C.-L.C. and L.-H.G.; writing—review and editing, W.-J.T., Y.-Y.D., C.-T.L. and H.S.; supervision, C.-L.C. and M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Ministry of Science and Technology in Taiwan (nos. MOST 111-2218-E-305-001-MBK and MOST 110-2410-H-324-004-MY2), the Science and Technology Project of Jilin Provincial Department of Education (JJKH20210457KJ), the Jilin Province Science and Technology Development Plan Project (20220508038RC), the Undergraduate Training Programs for Innovation and Entrepreneurship Project of Jilin Province (J202210203JSJ02), the CERNET Innovation Project (NGII20180315), and the National Natural Science Foundation for Young Scientists of China (grant no. 51808474).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

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

References

1. Wang, Y.; Lin, J.; Choi, T.-M. Gray market and counterfeiting in supply chains: A review of the operations literature and implications to luxury industries. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *133*, 101823. [CrossRef]
2. Dan, A.I.; Ran, W.X.; Hu, D.; Xu, X.; Pan, S.E. Study on Optimization of Anti-counterfeiting Process of Luxury Goods Based on EPC IOT. *Logist. Technol.* **2012**, *31*, 4.
3. Chen, C.-L.; Lim, Z.-Y.; Liao, H.-C.; Deng, Y.-Y.; Chen, P. A Traceable and Verifiable Tobacco Products Logistics System with GPS and RFID Technologies. *Appl. Sci.* **2021**, *11*, 4939. [CrossRef]
4. Deepa, N.; Pham, Q.-V.; Nguyen, D.C.; Bhattacharya, S.; Prabadevi, B.; Gadekallu, T.R.; Maddikunta, P.K.R.; Fang, F.; Pathirana, P.N. A survey on blockchain for big data: Approaches, opportunities, and future directions. *Futur. Gener. Comput. Syst.* **2022**, *131*, 209–226. [CrossRef]
5. Stamatellis, C.; Papadopoulos, P.; Pitropakis, N.; Katsikas, S.; Buchanan, W.J. A Privacy-Preserving Healthcare Framework Using Hyperledger Fabric. *Sensors* **2020**, *20*, 6587. [CrossRef]
6. Chen, C.-L.; Wang, T.; Tsaur, W.-J.; Weng, W.; Deng, Y.-Y.; Cui, J. Based on Consortium Blockchain to Design a Credit Verifiable Cross University Course Learning System. *Secur. Commun. Netw.* **2021**, *2021*, 8241801. [CrossRef]
7. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 23 September 2022).
8. Hyperledger Fabric. Available online: https://wiki.hyperledger.org/display/Fabric (accessed on 1 June 2021).
9. Qi, M.; Chen, J.; Chen, Y. A secure biometrics-based authentication key exchange protocol for multi-server TMIS using ECC. *Comput. Methods Programs Biomed.* **2018**, *164*, 101–109. [CrossRef] [PubMed]
10. Puthal, D.; Ranjan, R.; Nanda, A.; Nanda, P.; Jayaraman, P.; Zomaya, A.Y. Secure authentication and load balancing of distributed edge datacenters. *J. Parallel Distrib. Comput.* **2019**, *124*, 60–69. [CrossRef]
11. Mohit, P.; Amin, R.; Biswas, G. Design of authentication protocol for wireless sensor network-based smart vehicular system. *Veh. Commun.* **2017**, *9*, 64–71. [CrossRef]
12. Wazid, M.; Das, A.K.; Kumari, S.; Li, X.; Wu, F. Design of an efficient and provably secure anonymity preserving three-factor user authentication and key agreement scheme for TMIS. *Secur. Commun. Netw.* **2016**, *9*, 1983–2001. [CrossRef]
13. Moon, A.H.; Iqbal, U.; Bhat, G.M. Implementation of Node Authentication for WSN Using Hash Chains. *Procedia Comput. Sci.* **2016**, *89*, 90–98. [CrossRef]
14. Internet Says, The Advantages and Disadvantages of RFID The Advantages of Electronic Tag Technology. Available online: https://www.maigoo.com/goomai/153745.html (accessed on 3 May 2022).
15. Hochholdinger, S.; Arnoux, M.; Delémont, O.; Esseiva, P. Forensic intelligence on illicit markets: The example of watch counterfeiting. *Forensic Sci. Int.* **2019**, *302*, 109868. [CrossRef]
16. Pérez, J.B.; Queiruga-Dios, A.; Martínez, V.G.; Del Rey, M. Traceability of Ready-to-Wear Clothing through Blockchain Technology. *Sustainability* **2020**, *12*, 7491. [CrossRef]
17. Agrawal, T.K.; Kumar, V.; Pal, R.; Wang, L.; Chen, Y. Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry. *Comput. Ind. Eng.* **2021**, *154*, 107130. [CrossRef]

18. Han, W.; Zhu, Z. An ID-based mutual authentication with key agreement protocol for multiserver environment on elliptic curve cryptosystem. *Int. J. Commun. Syst.* **2014**, *27*, 1173–1185. [CrossRef]
19. Chang, S.E.; Chen, Y.-C.; Lu, M.-F. Supply chain re-engineering using blockchain technology: A case of smart contract based tracking process. *Technol. Forecast. Soc. Chang.* **2019**, *144*, 1–11. [CrossRef]
20. Crosby, M.; Pattanayak, P.; Verma, S.; Kalyanaraman, V. Blockchain technology: Beyond bitcoin. *Appl. Innov.* **2016**, *2*, 71.
21. Chen, C.L.; Deng, Y.Y.; Li, C.T.; Zhu, S.; Chiu, Y.J.; Chen, P.Z. An IoT-based traceable drug anti-counterfeiting management system. *IEEE Access* **2020**, *8*, 224532–224548. [CrossRef]
22. Schmidt, C.G.; Wagner, S.M. Blockchain and supply chain relations: A transaction cost theory perspective. *J. Purch. Supply Manag.* **2019**, *25*, 100552. [CrossRef]
23. Androulaki, E.; Cachin, C.; De Caro, A.; Sorniotti, A.; Vukolic, M. Permissioned blockchains and hyperledger fabric. *Ercim News* **2017**, *110*, 9–10.
24. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, G.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the 13th EuroSys Conference, Porto, Portugal, 23–26 April 2018; ACM: New York, NY, USA, 2018; pp. 1–15.
25. Nasir, Q.; Qasse, I.A.; Abu Talib, M.; Nassif, A.B. Performance Analysis of Hyperledger Fabric Platforms. *Secur. Commun. Netw.* **2018**, *2018*, 3976093. [CrossRef]
26. A Blockchain Platform for the Enterprise-Hyperledger Fabric. Available online: https://hyperledger-fabric.readthedocs.io/en/release-2.2 (accessed on 23 September 2022).