Hindawi
Security and Communication Networks
Volume 2021, Article ID 5701258, 18 pages
https://doi.org/10.1155/2021/5701258

Research Article

A Fair and Privacy-Preserving Image Trading System Based on Blockchain and Group Signature

Le Wang,[1] **Xuefeng Liu**,[2] **and Xiaodong Lin**[1]

[1]*School of Computer Science, University of Guelph, Guelph, Ontario N1G2W1, Canada*
[2]*School of Cyber Engineering, Xidian University, Xi'an, Shaanxi 710071, China*

Correspondence should be addressed to Xiaodong Lin; xlin08@uoguelph.ca

Received 17 June 2021; Revised 20 September 2021; Accepted 5 October 2021; Published 31 October 2021

Academic Editor: A. S. M. Kayes

With the rise of digital images in our daily lives, there is a growing need to provide an image trading market where people can monetize their images and get desired images at prices that fit their budget. Those images are usually uploaded and stored onto centralized image trading service providers' servers and the transactions for image trading are processed by these providers. Unfortunately, transaction unfairness and users' privacy breaches have become major concerns since the service providers might be untrusted and able to manipulate image trading prices and infer users' private information. Recently, several approaches have been proposed to address the unfairness issue by using the decentralized ledger technique and smart contract, but users' privacy protection is not considered. In this paper, we propose a fair and privacy-preserving protocol that supports image fair exchange and protect user privacy. In particular, we exploit blockchain and Merkle tree to construct a fair image trading protocol with low communication overhead based on smart contract, which serves as an external judge that resolves disputes between buyers and sellers in image transactions. Moreover, we extend a popular short group signature scheme to protect users' identity privacy, prevent linkability of transactions from being inferred, and ensure traceability of malicious users who may sell fake images and/or refuse to pay. Finally, we design and build a practical and open-source image trading system to evaluate the performance of our proposed protocol. Experimental results demonstrate its effectiveness and efficiency in real-world applications.

1. Introduction

Digital imaging devices such as digital cameras are becoming more integrated in our lives. Before, they were simply used for entertainment purposes. However, people are now starting to use them for a profession, by sharing, selling, and trading pictures and images. As a result, an online market place for image trading is imperative to ensure the success of this profession. To satisfy the needs of image trading, there already exist many image trading service providers (ITSP), such as ShutterStock [1], iStockphoto [2], Fotolia [3], and Dreamstime [4]. They can offer users efficient and convenient image transaction services with a much lower marginal cost than traditional approaches.

The fairness of image transaction in the ITSP, however, is subject to skepticism and scrutiny, as user's images are stored in the ITSP's servers and all transactions are completed via the ITSP, who waits to receive money from buyers and images from sellers and only executes the exchange based on the hash computation [5]. Unfortunately, such fully third-party trusted ITSPs are often not available in real world. This is because ITSPs are viewed as a black box to users, thereby being able to manipulate image trading prices and volumes to chase higher profits, resulting in unfairness on users. For example, a seller sells an image for $2, but an ITSP could charge the price for $10 and save $8 for himself, which is obviously unfair to the seller.

In general, in order to complete image transactions more efficiently and conveniently, ITSPs usually requires users to submit their identity information, including phone numbers, e-mail addresses, bank card numbers, and home addresses, for various purposes, such as taxation. However, once these data are sent and stored on the ITSP's server, users will lose control of their data, leading their private

information to be vulnerable to the untrustworthy ITSPs, as well as intruders. Additionally, through analyzing users' transaction history, malicious ITSPs and intruders are able to infer linkability of transactions which occurred among different users. Even more, a particular user's transaction habit can be inferred, for example, what type of photos he prefers to buy.

Recent works have been focusing on solving the fairness issue incurring in an image transaction by using the blockchain technique [6–8]. The fairness is guaranteed by relying on a smart contract executed over decentralized cryptocurrencies, where the smart contract takes the role of an external judge that completes the exchange in case of disagreement [6]. In addition to smart contracts, perceptual hashing is leveraged to automatically detect and reject tampered images that are perceptually similar to images that are already present on the marketplace [7]. Besides, Zhao et al. designed an image network copyright transaction protection approach based on blockchain technology [8] so that the entire copyright transaction process is protected and the attribute identification of image content is identified. Unfortunately, these works only consider solving the fairness issue, while addressing the privacy issue at the same time remains to be an open problem.

In this paper, we propose to build a practical image trading system that can provide guarantee on achieving both fairness and privacy protection in an image trading process. The major contributions of this paper are summarized as follows:

(i) Fairness in trading is achieved by a fair image trading protocol that is constructed by utilizing blockchain and Merkle tree. Particularly, a smart contract deployed on the blockchain acts as a trusted and automatic judge to complete the image transaction in case of disagreement. Besides, compared to the image trading system that does not support digital goods preview before trading [6], our advantage lies in designing a distributed thumbnail preview mechanism to enhance user-friendliness and further guarantee the authenticity of trading images.

(ii) To address user privacy issue, we enhanced a popular short group signature BBS04 [9] to our proposed system so that a verifier is able to verify a signer's signature on transaction data, while the signer's identity is kept private from the verifier. In addition, users and intruders cannot infer any private information about a particular user by analyzing his or her image transaction history. Meanwhile, we redefined the role of the group manager [9] and replaced the sole manager with three independent managers to prevent them from maliciously deducing private information of the system. More importantly, we maintain the ability to track malicious users.

(iii) We design and build a practical image trading system that is developed from scratch, which can be adopted as a real-world application in the trading market. Moreover, we evaluate the performance of the proposed protocol in terms of the enhanced short group signature module computation cost, fabric network upload and download latency, image block encryption and decryption computation cost on mobile devices, and smart contract judgement cost. Experimental results show that the proposed protocol is efficient and effective in guaranteeing fairness and ensuring privacy protection.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce the building blocks, including Merkle tree, short group signature, IPInter Planetary File System (IPFS), blockchain, and smart contract, which are used in designing our image trading system. In Section 3, we present the system model, problem, and threat model. The detailed design of the proposed protocol is presented in Section 4 and Section 5. Then, the security and privacy of the proposed protocol will be analyzed in Section 6. In Section 7, we introduce the prototyping system that is developed from scratch and evaluate its performance. Finally, we briefly discuss related work in Section 8, and conclude this paper in Section 9.

2. Preliminaries

2.1. Merkle Tree. A *Merkle tree* is an authenticated data structure where every leaf node of the tree contains the cryptographic hash of a data block, and every nonleaf node contains the concatenated hashes of its child nodes [10].

A Merkle tree contains leaf nodes $x_1, \ldots, x_n \in \{0,1\}^*$, where n is an integer, is a tagged binary tree $MT = \text{Mtree}(x_1, x_2, \ldots, x_n)$ and the ith leaf node is labelled by x_i. Furthermore, a label y_j of nonleaf node Y_j is the hash value of the labels y_j^l and y_j^r, where y_j^l and y_j^r are labels for child nodes Y_j^l and Y_j^r, respectively (i.e., $y_j := H(y_j^l, y_j^r)$, where H stands for hash function). We call y_j the parent of y_j^l and y_j^r. We name y_j^l as the slibing of y_j^r and vice versa. A Merkle tree of n elements $x_1, x_2, \ldots, x_n$ is created by Algorithm 1.

The root node of a Merkle tree MT is labelled as $\text{root}(MT)$. To prove that an element x_i is one of leaf nodes of the Merkle tree, a Merkle proof p is used, which is a vector composed of labels on all the siblings of elements on a path from the ith leaf node to the root node. We denote Algorithm 2 for generating a Merkle proof by Merkle Tree Proof Generation, which takes a Merkle tree MT and an index i as inputs and outputs a Merkle proof p that x_i is the ith leaf of MT.

At last, the algorithm Merkle tree verification (Algorithm 3) takes as input an element x_i, a Merkle proof p, and a root of Merkle tree $\text{root}(MT)$. This algorithm is used to verify whether x_i is a leaf node of the Merkle tree MT with root r using proof p. If the verification is successful, the algorithm outputs 1, otherwise, the algorithm outputs 0.

2.2. Short Group Signature. The concept of short group signature was first proposed by Boneh et al. [9] in 2004. With short group signature, any member of the group can sign

Input: $(x_1, x_2, \ldots, x_n)$
set Y = root node
if $n = 1$ **then**
  label $(Y) = x_1$
else
  $y_0^l = \text{Mtree}(x_1, x_2, \ldots, x_{\lceil n/2 \rceil})$
  $y_0^r = \text{Mtree}(x_{\lceil n/2 \rceil}, \ldots, x_n)$
  label $(Y) = H(\text{root}(y_0^l) \| \text{root}(y_0^r))$
end if
Output: Merkle tree MT with root Y

Algorithm 1: Merkle tree creation.

Input: Merkle Tree MT, index i
$V = MT[i]$
for $k \in [\log_2(n)]$ **do**
  set y_k = label (sibling of v)
  set v = parent of v
end for
Output: Merkle Proof $p = (y_1, \ldots, y_d)$

Algorithm 2: Merkle tree proof generation.

Input: $i \in [n], x \in \{0,1\}^\lambda, p = (l_1, \ldots, l_d), r \in \{0,1\}^\mu$
for each $l_k \in p$ do
if $i/2^k \text{mod} 2 = 0$ **then**
  $x = H(l_m \| x)$
  if $H(\text{is Prgrmd}(l_m \| x))$ **then**
    Terminate and Output
  end if
else
  $x = H(x \| l_k)$
  if $H(\text{is Prgrmd}(x \| l_l))$ **then**
    Terminate and Output
  end if
end if
if $x = r$ **then**
  Output 1
else
  Output 0
end if

Algorithm 3: Merkle tree verification.

message, but the resulting signature keeps the identity of the signer confidential. More concretely, given a short group signature and a group of n users, a verifier cannot distinguish the signer's identity. This property can be used to preserve the identity of the signer. Compared to the group signatures based on Strong-RSA [11], the length of short group signatures is shorter, about 200 bytes, so it is more suitable for practical applications.

The short group signature scheme is constructed on the Strong Diffie–Hellman (SDH) assumption in groups with a bilinear map. In this paper, we will improve it to construct a privacy-preserving protocol with conditional accountability in our decentralized image trading system.

2.3. IPFS. The InterPlanetary File System (IPFS) [12] is a hypermedia protocol and peer-to-peer network for storing and sharing data in a distributed file system. In a nutshell, IPFS is similar to a network, but it can be viewed as a single BitTorrent swarm that exchanges objects within a single Git

repository. Specially, IPFS maintains a high-throughput content-addressable block storage model with content-addressable hyperlinks. This results in a generalized Merkle DAG, a data structure on which people can build blockchains, versioned file systems, and even a persistent network. Importantly, IPFS has no single point of failure and the nodes do not need to trust each other. Distributed content delivery can save bandwidth and prevent DDoS attacks that HTTP schemes may encounter [13]. In this paper, we deployed a four-node IPFS system that stores thumbnail images users plan to trade.

2.4. Blockchain and Smart Contract. Cryptocurrencies are gaining increasing attention from academia and industry in recent years. Cryptocurrencies are based on a public digital ledger that stores all current and historical transactions and is maintained by decentralized miners [14]. The digital ledger is stored in the form of blockchain, and all miners agree on its state through a consensus protocol. Due to its special data structure, the blockchain enables decentralization, transaction immutability, traceability, and transparency. Briefly speaking, the blockchain technique is a decentralized and, oftentimes, public digital ledger, which consists of records called blocks that are used to record transactions across miners, and has drawn much attention from academia recently [15–19].

The smart contract is built on top of cryptocurrencies and allows users to define and execute contracts on the blockchain [20]. In particular, a smart contract is a self-executing computer program and consists of functions (executable units of code within the contract) and data (the state of the smart contract). The program code captures the logical contract terms between multiple parties and predefines trigger conditions and response actions. The execution of functions in a smart contract is triggered by time or events. Ideally, smart contracts can be regarded as executed by a distributed and trusted global machine that will faithfully execute every instruction. With the help of smart contracts, the rules of fair exchange can be enforced without trusted third parties.

The Ethereum is one of the most popular public permissionless blockchain platforms [20]. In short, the Ethereum is a public decentralized network, open to anyone, where participants interact anonymously. There exist two types of accounts in the Ethereum network: externally owned accounts and smart contract accounts. An externally owned account is associated with a unique public-private key pair, owned by someone (e.g., seller, buyer, and committee) who has an Ether balance, and the private key can sign transactions from that account. Contract accounts do not have key pairs. A smart contract account maintains an Ether balance and stores a contract code that determines the flow of Ether in the account. A smart contract account must be activated by an externally owned account. In order to execute a smart contract used to transfer cryptocurrencies, a user-controlled account must pay a certain amount of gas using Ether. The gas fee is actually a transaction fee that encourages miners to incorporate the code execution of the smart contract into the blockchain. Gas can be seen as an indicator of the cost of a regulated smart contract, with each assembly operation having a fixed gas cost based on its expected execution time. However, as more and more Ethereum-enabled applications become available, transaction activities become more frequent and active, leading to ever higher transaction gas fees and lower transaction consensus efficiency.

Hyperledger fabric [21] is another type of blockchain, the permissioned blockhead, and it is designed for use in an enterprise context and offers some key differentiating features compared to the public permissionless blockchain. One of the most important different factors is the support for pluggable consensus protocols, allowing the platform to be more effectively customized to fit specific use cases and trust models. Additionally, fabric can use consensus protocols that do not require native cryptocurrency to incentivize expensive mining or drive smart contract execution. Absence of cryptographic mining operations means that the platform can be deployed at roughly the same operating cost as other distributed systems, meaning that users do not have to afford high gas costs for transactions. Finally, miner nodes are mutually known and not anonymous in fabric network, this means that while miner nodes may not fully trust each other, the network can operate under a governance model that builds on the trust that does exist between miner nodes, such as a legal agreement or a framework for handling disputes. The combination of these differentiated design features makes fabric one of the better performing platforms today in terms of transaction processing and transaction confirmation latency.

3. Problem Statement

3.1. System Model. As illustrated in Figure 1, the system model in this paper involves six parties: a seller, a buyer, a committee, an IPFS system, the Ethereum platform, and a consortium blockchain with smart contract. The seller and buyer upload and download thumbnails from the IPFS via their smart devices. In addition, they exchange data (e.g., Merkle root hash r of a traded image) through the smart contract deployed on the consortium blockchain (data flow blockchain). At the same time, the buyer, seller, and committee transfer cryptocurrencies among them on the Ethereum platform. The committee is host-and-curious and made up of three members, which is responsible for generating group public key, issuing group private keys for users and recovering malicious user's real identity.

When sellers plan to put their images for sale, first of all, they must locally compute and upload thumbnails of images to the IPFS, and then, buyers can view them on their own devices (e.g., smartphones) and decide whether to buy. If yes, the smart contract deployed on the consortium blockchain is used to guarantee the integrity and authenticity of traded images between sellers and buyers. Meanwhile, both sides of the transaction complete a reliable transfer of cryptocurrency corresponding to the value of images based on Ethereum. If disputes arise during the above processes, for

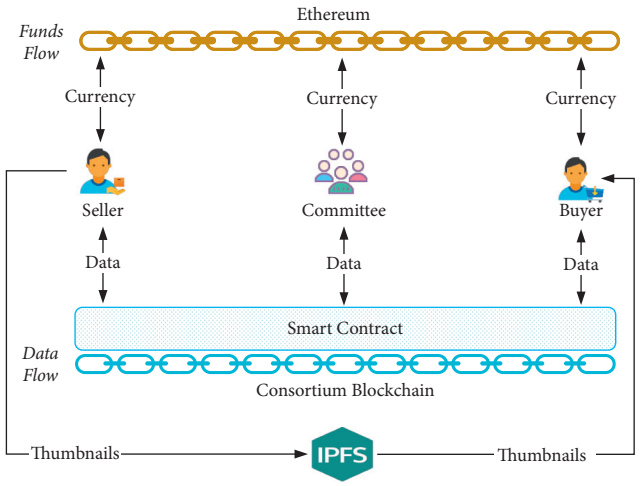

Figure 1: System model.

instance, the buyer finds that the traded images are fake or the seller does not receive the money, the smart contract will intervene to prevent sellers from selling fake photos and buyers from refusing to pay.

3.2. Research Problem. In this system model, there exist two major issues to be addressed, which are summarized as the following two research problems.

Unfairness: in our system model, unfairness incurs in both buyers and sellers. For an honest buyer, unfairness means not receiving the real photos from sellers after paying for them; for an honest seller, not receiving the money after the real images are delivered to buyers.

Privacy leakage: as mentioned in Section 1, in the traditional image trading system, two types of private information can be compromised. The first one is the user's identity information, such as name, telephone number, e-mail address, and bank card number, which may be disclosed by ITSPs for profit. The second one is user's linkability of transactions. Because the fact that the linkability of transactions might reflect the user relationships, user's consumption habits, and preferences, and once this information is obtained and inferred by a malicious person, it will lead to terrible consequences.

3.3. Threat Model. Fairness threats: based on the system model, fairness threats are likely to derive from all participants in the image trading process, such as sellers, buyers, and committee. Therefore, three types of threats related to the fairness of image trading are possible, which are summarized as follows:

(i) Threat1: the seller might use fake images to transact with the buyer and earn illegal profits, which will result in a loss to the buyer

(ii) Threat2: the buyer might refuse to pay money to the seller after receiving the traded image, which will cause a loss of seller earnings

(iii) Threat3: the committee may embezzle the deposits which are used to complete image transactions between buyers and sellers

Privacy threats: the user's identity and linkability of image transaction are private and confidential to the committee members, users (buyers and sellers), and intruders. However, a malicious user or an external attacker may want to detect user's identity information and analyze user's linkability of image transactions, as, based on these data, they can snoop on the user's image preference and other more intimate information. In addition, in most cases, the committee members are honest, but some of them still have the incentive to unlawfully reveal the user's identity. In order to clearly, the privacy threats are classified into the following three items.

(i) Threat4: the attacker detects the user's identity information

(ii) Threat5: the attacker infers the user's linkability of image transactions

(iii) Threat6: the attacker analyzes user's image preference

4. Short Group Signature Scheme with Conditional Accountability

4.1. Overview. As introduced in previous sections, we intent to utilize short group signature to protect user's identity information, user's linkability of transactions, and even user's image preference so that private or sensitive information of users is not disclosed to malicious users and external attackers. However, traditional short group signature (BBS04) [9] cannot be directly used into protecting users' identity privacy in our protocol. This is because the normal group signature scheme sets up a group manager to simultaneously issue group private key to users and reveal their true identity once they have misbehavior. This is very scary; if the only one manager goes corrupt or is compromised by an attacker, then all the users' real identity information will be leaked, along with their linkability and image preferences.

Therefore, we design an enhanced short group signature (short for ESGS) scheme with conditional accountability, which is extended from a classic short group signature scheme [9]. This new scheme is not only able to preserve user's identity privacy but also able to prevent the leakage of user privacy in case of the group manager failure. We will show how to build this enhanced short group signature for protecting user privacy in the Section 4.2.

4.2. Construction of ESGS. ESGS contains four algorithms: Key Gen, Sign, Verify, and Open. In Key Gen, a committee member (named as Issuer) is responsible for generating the group public key gpk, keeping the secret parameter γ and issuing private key gsk$[i]$ for user i. Moreover, each of the other two committee members (named as Revealer$_1$ and Revealer$_2$) holds a private key, ξ_1 and ξ_2, respectively, which are used to jointly recover malicious user's identity. In Sign, a group member i signs the transaction data with his/her

private key gsk$[i]$. A verifier (any group member) is able to check whether the signature of a transaction signed by a group member is correct in Verify. In Open, if a user is identified by the committee for misbehaviour, such as selling fake photos, then Revealer$_1$ and Revealer$_2$ work together to reveal the true identity of the user using ξ_1 and ξ_2. Details of this scheme are described in Figure 2.

Consider bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with respective generators g_1 and g_2. Suppose further that the SDH assumption holds on $(\mathbb{G}_1, \mathbb{G}_2)$, and the linear assumption holds on $\mathbb{G}_1$. The enhanced short group signature scheme employs a hash function H: $\{0,1\}^* \longrightarrow \mathbb{Z}_p$, treated as a random oracle in the proof of security. The total number of users in the group is n.

KeyGen: Issuer selects $h \xleftarrow{R} \mathbb{G}_1 / \{1_{\mathbb{G}_1}\}$ and $\gamma \xleftarrow{R} \mathbb{Z}_p{}^*$ and sets $\omega = g_2^{\gamma}$. Using γ generates, for each user i, $1 \le i \le n$, and SDH tuple (A_i, x_i) selects $x_i \xleftarrow{R} \mathbb{Z}_p{}^*$ and sets $A_i \longleftarrow g_1{}^{1/(\gamma + x_i)}$. At the same time, Issuer sends h to Revealer$_1$ and Revealer$_2$ via the secure channel. Revealer$_1$ selects $\xi_1 \xleftarrow{R} \mathbb{Z}_p{}^*$ and sets $u \in \mathbb{G}_1$ such that $u^{\xi_1} = h$. Then, Revealer$_1$ sends back u to Issuer via the secure channel. In a similar way, Revealer$_2$ selects $\xi_2 \xleftarrow{R} \mathbb{Z}_p{}^*$ and sets $v \in \mathbb{G}_1$ such that $v^{\xi_1} = h$. Then, Revealer$_1$ sends back v to Issuer via the secure channel. Therefore, the group public key of ESGS is gpk $= (g_1, g_2, h, u, v, \omega)$. The private key of Revealer$_1$ and Revealer$_2$ is ξ_1 and ξ_2, respectively. Every user's private key is his or her tuple gsk$[i] = (A_i, x_i)$. The secret parameter γ is only known to the private key Issuer.

Sign: the user i holds his or her private key gsk$[i] = (A_i, x_i)$, invokes the smart contract to get the group public key gpk $= (g_1, g_2, h, u, v, \omega)$ from the consortium blockchain and an image block $M \in \{0,1\}^*$, and computes and outputs a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ for the image block M.

Verify: given a group public key gpk $= (g_1, g_2, h, u, v, \omega)$, an image block M, and a group signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. A verifier first computes R_1, R_2, R_3, R_4, and R_5 with formula (1)–(5).

$$R_1 = \frac{u^{s_\alpha}}{T_1^{c}}, \tag{1}$$

$$R_2 = \frac{v^{s_\beta}}{T_2^{c}}, \tag{2}$$

$$R_3 = e(T_3, g_2)^{s_x} * e(h, \omega)^{-s_\alpha - s_\beta} * e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} * \left(\frac{e(T_3, \omega)}{e(g_1, g_2)}\right)^{C}, \tag{3}$$

$$R_4 = \frac{T_1^{s_x}}{u^{s_{\delta_1}}}, \tag{4}$$

$$R_5 = \frac{T_2^{s_x}}{v^{s_{\delta_2}}}. \tag{5}$$

Then, it checks

$$c \stackrel{?}{=} H(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5). \tag{6}$$

If equation (6) holds, then the given image block M is signed by one of these n users in the group. Otherwise, it is not.

Open: this algorithm is used for recover any malicious user's identity based their signatures. It takes as input a group public key gpk $= (g_1, g_2, h, u, v, \omega)$ and the corresponding private keys, Revealer$_1$'s ξ_1 and Revealer$_2$'s ξ_2, together with a message M and a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. It follows the following steps to execute. Firstly, verify that σ is a valid signature on M. Second, consider the first three elements (T_1, T_2, T_3) as a linear encryption. Third, Revealer$_1$ computes $\lambda \longleftarrow T_1{}^{\xi_1}$ and then sends λ to Revealer$_2$ via the secure channel. Revealer$_2$ recovers the user's identity A as $A \longleftarrow T_3 / \lambda \cdot T_2{}^{\xi_2}$, and vice versa. At last, Revealer$_2$ looks up the user index corresponding to the identity A recovered from the signature.

4.3. Security Analysis of ESGS. As described above, our enhanced short group signature scheme inherits from the traditional short group signature scheme BBS04. Both of them are composed of four algorithms, namely, Key Gen, Sign, Verify, and Open. More specifically, the first three algorithms remain the same in our scheme and BBS04 scheme, so their security has been proved by BBS04. As for the Open algorithm, we enhance its security through utilizing multiple group managers (Revealer$_1$ and Revealer$_1$) to replace the only one group manager of the BBS04. Suppose that these two Revealers are honest-and-curious and non-colluding. Compared to the original Open algorithm in BBS04, in order to successfully recover user's identity, Revealer$_1$ and Revealer$_2$, must work together to reveal the malicious user's identity. Moreover, none of them have the ability to recover the malicious user's identity. Now, the security of the Open algorithm of ESGS can be proved as follows.

Theorem 1. *For any adversary $\mathcal{A}$ who has the private key of one of the two Revealers (i.e., Revealer$_1$'s private key ξ_1), we can create a simulator which has the ability to construct parameters using the same methods as the ESGS based on a DDH (Decisional Diffie–Hellman) triple. Afterward, the simulator sends parameters to the adversary; if he/she can calculate the identity A of the user, then it means that the simulator has the ability to solve the DDH problem.*

Simulator: there are two main operations which should be completed by the simulator. One of them is to construct DDH triple, the other is to generate simulated parameters for the adversary based on the methods of the ESGS. The details are shown below:

(i) DDH triple: consider a (multiplicative) cyclic group $\mathbb{G}$ of order p, with generator v. The DDH assumption states that, given v^a and v^b for randomly chosen

Consider bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with respective generators g_1 and g_2. Suppose further that the SDH assumption holds on $(\mathbb{G}_1, \mathbb{G}_2)$, and the Linear assumption holds on $\mathbb{G}_1$. The enhanced short group signature scheme employs a hash function H: $\{0,1\}^* \to \mathbb{Z}_p$, treated as a random oracle in the proof of security. The total number of users in the group is n.

KeyGen. The $Issuer$ selects $h \xleftarrow{R} \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$ and $\gamma \xleftarrow{R} {\mathbb{Z}_p}^*$ and sets $\omega = g_2^{\gamma}$. Using γ, generates for each user i, $1 \le i \le n$, and SDH tuple (A_i, x_i): selects $x_i \xleftarrow{R} {\mathbb{Z}_p}^*$, and sets $A_i \leftarrow g_1^{1/(\gamma + x_i)}$. At the same time, the $Issuer$ sends h to the $Revealer_1$ and $Revealer_2$ via the secure channel. $Revealer_1$ selects $\xi_1 \xleftarrow{R} {\mathbb{Z}_p}^*$, and sets $u \in \mathbb{G}_1$ such that $u^{\xi_1} = h$. Then $Revealer_1$ sends back u to the $Issuer$ via the secure channel. In a similar way, $Revealer_2$ selects $\xi_2 \xleftarrow{R} {\mathbb{Z}_p}^*$, and sets $v \in \mathbb{G}_1$ such that $v^{\xi_1} = h$. Then $Revealer_1$ sends back v to the $Issuer$ via the secure channel. Therefore, the group public key of ESGS is $gpk = (g_1, g_2, h, u, v, \omega)$. The private key of the $Revealer_1$ and $Revealer_2$ is ξ_1 and ξ_2 respectively. Every user's private key is his or her tuple $gsk[i] = (A_i, x_i)$. The secret parameter γ is only known to the private-key $Issuer$.

Sign. The user i holds his or her private key $gsk[i] = (A_i, x_i)$, invokes the smart contract to get the group public key $gpk = (g_1, g_2, h, u, v, \omega)$ from the consortium blockchain, and an image block $M \in \{0,1\}^*$, computes and outputs a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ for the image block M.

Verify. Given a group public key $gpk = (g_1, g_2, h, u, v, \omega)$, an image block M, and a group signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. A verifier first computes R_1, R_2, R_3, R_4 and R_5 with formula (1), (2), (3), (4) and (5).

$$R_1 = u^{s_\alpha} / T_1^{c}. \tag{1}$$

$$R_2 = v^{s_\beta} / T_2^{c}. \tag{2}$$

$$\begin{aligned} R_3 =& e(T_3, g_2)^{s_x} * e(h, \omega)^{-s_\alpha - s_\beta} \\ & * e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} * (e(T_3, \omega) / e(g_1, g_2))^{c}. \end{aligned} \tag{3}$$

$$R_4 = T_1^{s_x} / u^{s_{\delta_1}}. \tag{4}$$

$$R_5 = T_2^{s_x} / v^{s_{\delta_2}}. \tag{5}$$

then checks

$$c \stackrel{?}{=} H(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5). \tag{6}$$

If the equation (6) holds, then the given image block M is signed by one of these n users in the group. Otherwise, it is not.

Open. This algorithm is used for recover any malicious user's identity based their signatures. It takes as input a group public key $gpk = (g_1, g_2, h, u, v, \omega)$, the corresponding private keys: $Revealer_1$'s ξ_1 and $Revealer_2$'s ξ_2, together with a message M and a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. It follow the following steps to execute. Firstly, verify that σ is a valid signature on M. Second, consider the first three elements (T_1, T_2, T_3) as a Linear encryption. Third, the $Revealer_1$ computes $\lambda \leftarrow T_1^{\xi_1}$, and then send λ to $Revealer_2$ via the secure channel, $Revealer_2$ recovers the user's identity A as $A \leftarrow T_3 / \lambda \cdot T_2^{\xi_2}$, and vice versa. At last, the $Revealer_2$ looks up the user index corresponding to the identity A recovered from the signature.

Figure 2: Details of ESGS.

$a, b \in \mathbb{Z}_p$, find $y \in \mathbb{G}$ and make $y = v^{ab}$. So, the DDH triple is (v^a, v^b, y).

(ii) Simulated parameters: construct two (multiplicative) cyclic group $\mathbb{G}_1$ and $\mathbb{G}_2$; their prime order is p, g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, and e is a computable map e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Select $h \xleftarrow{R} \mathbb{G}_1 / \{1_{\mathbb{G}_1}\}$, and set $u, v \in \mathbb{G}_1$ such that $h = u^{\xi_1} = v^a$. Select $\gamma \xleftarrow{R} {\mathbb{Z}_p}^*$. Moreover, construct the first three elements T_1, T_2, and T_3 of a user's ESGS signature σ and make $T_1 = u^\alpha$, $T_2 = v^b$, and $T_3 = A \cdot h^\alpha \cdot y$. Until now, the simulator constructs the simulated parameters SP = $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, u, v, h, \gamma, T_1, T_2, T_3)$ and then sends it to the adversary.

Proof. When the adversary receives the simulated parameters SP, he or she follows the Open algorithm to calculate the user's identity A, and the calculation process is as follows:

$$\begin{aligned} a &= \frac{T_3}{{T_1}^{\xi_1} \cdot {T_2}^{\xi_2}} = \frac{A \cdot h^\alpha y}{\left(u^\alpha\right)^{\xi_1} \cdot \left(v^b\right)^{\xi_2}}, \\ &= \frac{A \cdot h^\alpha y}{\left(u_1^{\xi}\right)^{\alpha} \cdot \left(v^b\right)^{\xi_2}} = \frac{A \cdot h^\alpha y}{h^\alpha \cdot \left(v^b\right)^{\xi_2}} = A \cdot \frac{y}{v^{b\xi_2}}. \end{aligned} \tag{7}$$

Based on the Key Gen algorithm of the ESGS, we know that $\xi_2 \in \mathbb{Z}_p$. In addition, in the DDH assumption, a has the same property and $a \in \mathbb{Z}_p$. So, he/she can get the following equation:

$$A = A \cdot \frac{y}{v^{ab}}. \tag{8}$$

Therefore, with the advantage of knowing Releaver$_1$'s private key ξ_1, the adversary needs to find y and makes $y = v^{ab}$ so that he could get the correct identity A. This means that the simulator has the same probability of being able to solve the DDH problem, but it is computationally infeasible.

Above all, we can see that Open algorithm is secure in the case that the attacker knows a private key of one of the two Revealers.

5. Protocol Design

Using the Ethereum, a consortium blockchain with the smart contract, an IPFS system, and the enhanced short group signature, we now construct our proposed protocol, which is the foundation of a fair and privacy-preserving image trading system. Within this protocol, the seller and the buyer can complete image transactions fairly and without worrying about privacy disclosure, including identity privacy, transaction linkability, and image preferences. Meanwhile, in each transaction, the seller and the buyer can verify the authenticity of each other's identity through Sign – Verify mechanism and integrity of the transaction data under privacy protection.

In order to describe briefly and clearly, we divide this protocol into four stages, including preparation stage,

transaction stage, complaint stage, and revelation stage. Details of each stage are shown below.

5.1. Preparation Stage. As we mentioned in Section 3, the committee consists of three members, who are denoted by Issuer, Revealer_1, and Revealer_2. Issuer needs to generate initialization parameters and group public key of ESGS and store the secret parameter γ which is used to generate private key $\text{gsk}[i] = (A_i, x_i)$ for every user. Additionally, Revealer_1 and Revealer_2, respectively, generate private key ξ_1 and ξ_2 based on the received parameter h from Issuer. In particular, when the committee receives multiple complaints about a specific user, Revealer_1 and Revealer_2 work together to recover this user based on previous signatures, open his/her identity, and add it to the blacklist. It is required that Revealer_1 and Revealer_2 jointly disclose the users identity using ξ_1 and ξ_2; this protocol ensures that any individual on the committee does not have the ability to reveal identity of any user. The detailed process is shown in Figure 3.

During this phase, three preparatory tasks need to be completed. Let gpk denotes group signature public key, $\text{gsk}[i]$ and $\text{gsk}[j]$ denote the seller's private key and buyer's private key, respectively, Eth_Comm denotes the joint Ethereum account of the committee members, Eth_Buyer denotes the buyer's Ethereum account, and Eth_Seller denotes the seller's Ethereum account.

(i) Registering Ethereum address: in order to complete the cryptocurrency exchange, the buyer, the seller, and the committee are required to register the Ethereum account which is represented by the Ethereum address. Especially, the committee's Ethereum account is jointly generated by the committee members based on the outstanding work of [22]. Because the committee's Ethereum account is used to store the transaction deposits of users, this mechanism can ensure that the deposits can only be handled with the consent of all committee members.

(ii) Generating public and private keys of ESGS: Issuer generates group public key gpk and issues private key $\text{gsk}[k]$ for user k. In addition, he or she invokes the smart contract to upload gpk to the consortium blockchain, so users can get it to verify each other's signature on the transaction data. When a seller i applies for his/her private key from Issuer, Issuer generates private key $\text{gsk}[i]$ with the secret parameter γ and sends $gsk[i]$ to the seller. For the buyer j, the same approach is used to generate the private key $gsk[j]$. Specially, $i \le n$, $j \le n$, and $i \ne j$, where n is the total user number of the group.

(iii) Generating and sharing thumbnails: the seller generates thumbnails for all the photos for sale based on the Android compress function [23] and then uploads them to the IPFS. The buyer's application automatically downloads thumbnails from the IPFS, so the buyer can view them anytime and decide whether to buy.

5.2. Transaction Stage. This stage is very important in the entire protocol because it can help users (buyers and sellers) transact images in a fair and privacy-preserved way. As shown in Figure 4, the transaction procedures are completed by four entities, the buyer, the seller, the smart contract, and the committee's Ethereum account Eth_Comm. Among these entities, there are three kinds of data exchange channels, as shown in the Notes in Figure 4. The first one is called offchain secure channel, which is used to transfer data between the buyer and seller via the peer-to-peer secure channel, such as buyer's purchase message and seller's encrypted image blocks and the signature. The second is presented by consortium blockchain, which means the buyer and seller exchange transaction data with the smart contract deployed on the consortium blockchain. Ethereum are the last one, which denotes that deposit d is transferred among the buyer, the seller, and the committee.

During this stage, Let $\text{img\_block}[k]$ denote the kth leaf node of Merkle tree ($1 \le k \le n$, where n is the total number of leaf nodes), ρ_k denote the Merkle proof of the leaf node k based on Algorithm 2 described in Section 2, sk denote the encryption key of image blocks which is generated by the seller, S denote the set of encrypted image blocks and the signature, r denote the root hash of the Merkle tree, and d denote the transaction deposit.

Now, we discuss the details of the transaction stage. We suppose that the buyer selects a satisfying image via their smart device and then initiates a purchase request to the seller via offchain secure channel.

Once the seller receives the purchase request, he or she begins to compute the Merkle tree of the traded image, generates the encryption key sk, and calculates the kth encrypted image block and it proof $E_{sk}(\text{img\_block}[k], \rho_k)$, $1 \le k \le n$. Each encrypted image block can be independently verified with Algorithm 3 mentioned in Section 2. Afterward, the seller's private key $\text{gsk}[i]$ is used to sign for every encrypted image block, and then, they are packed as a set S. Lastly, the seller sends the set S and his or her Ethereum addresses Eth_Seller to the buyer. Simultaneously, the seller sends the Merkle tree root hash r of the traded image to the smart contract.

After the buyer receives S and Eth_Seller via the offchain secure channel, he/she invokes the smart contract to get the group public key gpk and uses it to verify the signature of each encrypted image block. If verification passes, the buyer transfers the deposit d to the committee's Ethereum address Eth_Comm and notifies the smart contract and the seller. Additionally, the seller can confirm this notification by viewing the transaction history of Eth_Comm because of the transparency property of the Ethereum. If it does not, the transaction will be terminated.

After the seller confirms that the buyer has stored deposit d in Eth_Comm, he or she reveals the secret key sk to the smart contract. The buyer gets it from the smart contract and uses it to decode encrypted image blocks stored in S. So, the buyer can get a new set S_{decode} that consists of $(\text{img\_block}[k], \rho_k)$ $1 \le k \le n$. The buyer next verifies the integrity of each image block via Algorithm 3. If the result is correct, the buyer notifies the seller, smart contract, and

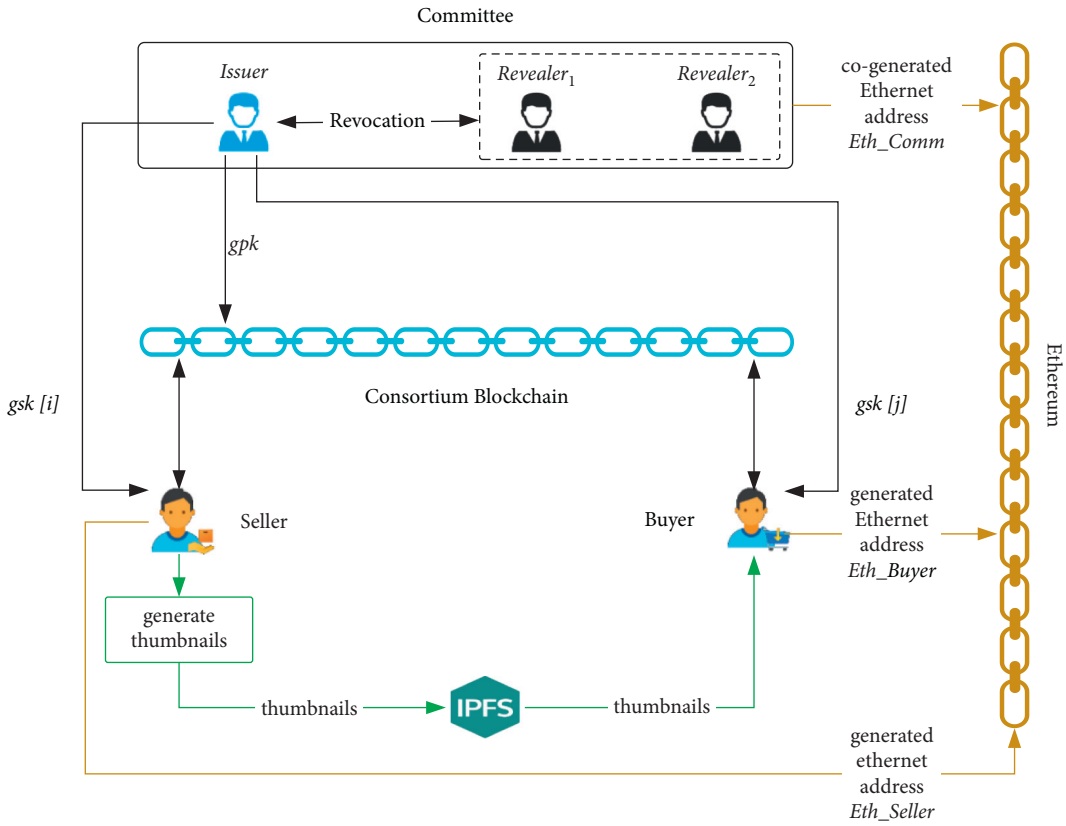

Figure 3: The details of the preparation stage.

Eth_Comm, after the deadline, the committee transfers the deposit d to the seller's Ethereum address Eth_Seller. Otherwise, the protocol moves to the complaint stage.

As described above, we use the Merkle tree to construct the block-verifiable image blocks. Now, we will depict how to use this technology.

Suppose the seller has an image, as shown in Figure 5(a), he or she pursues the following steps to build the Mekle Tree for this image. When the seller receives the transaction request, he or she divides the image into blocks, such as 4 blocks in this example. Then, the hash value of each block is used as a leaf node of the Merkle tree, such as $H(1)$, $H(2)$, $H(3)$, and $H(4)$. Whereas every nonleaf node is labelled with the cryptographic hash of its child nodes, for example, $H(1,2)$ is the hash value of $H(1)$ and $H(2)$. Eventually, the root hash r will be computed which is mentioned in Figure 4. As a result, the image data blocks with self-verification capability are constructed successfully, and each one is composed of the leaf node image block img_block$[k]$ and its proof ρ_k, $1 \le k \le 4$. Finally, the seller encodes the self-verifiable image data blocks with the secret key sk and signs on them with the seller's private key. Until now, the set S is constructed completely; then, the seller sends it to the buyer.

After the set $S = E_{sk}(\text{img\_block}[k], \rho_k), \text{Sig}_{\text{gsk}[i]}$, $1 \le k \le 4$, is received by the seller through the offchain secure channel. Through signature verification and decryption described in Figure 4, the buyer can get self-verifiable image blocks, which are shown on the left side of Figure 5(b). In order to close the transaction, the integrity of each block should be verified carefully. For example, when block2 is verified, the buyer should firstly compute a new hash $H(2)'$ based on block2 and combine it with $H(1)$ to generate a new hash $H(1,2)'$. Finally, the buyer can link $H(1,2)'$ and $H(3,4)$ to compute a new root hash $r\prime$. If $r\prime$ is equal to the original root hash r originally uploaded to the smart contract by seller, this means there is no problem with block2. Otherwise, the buyer should launch the complaint process. It is noticed that only if all the data blocks (such as block1, block2, block3, and block4) pass the verification, the buyer can confirm that the traded image is fine and stitch image blocks into a complete image, and the transaction will be completed properly.

5.3. Complaint Stage. At the complaint stage, the smart contract acts as a self-executing arbiter, based on the data submitted by the buyer, to determine whether the seller has misbehavior. If yes, it refunds the deposit to the buyer. Otherwise, it transfers the deposit to the seller. Additionally, in order to allow enough complaint time for users and further ensure the fairness of the transaction, we set an expiration date for the complaint operation, such as two days or three days. Therefore, the user should launch the complaint operation before the deadline.

When the buyer launches the complaint, as shown in Figure 6, he/she needs to send the encrypted image blocks with failed integrity verification and the corresponding signature to the smart contract. This mechanism can

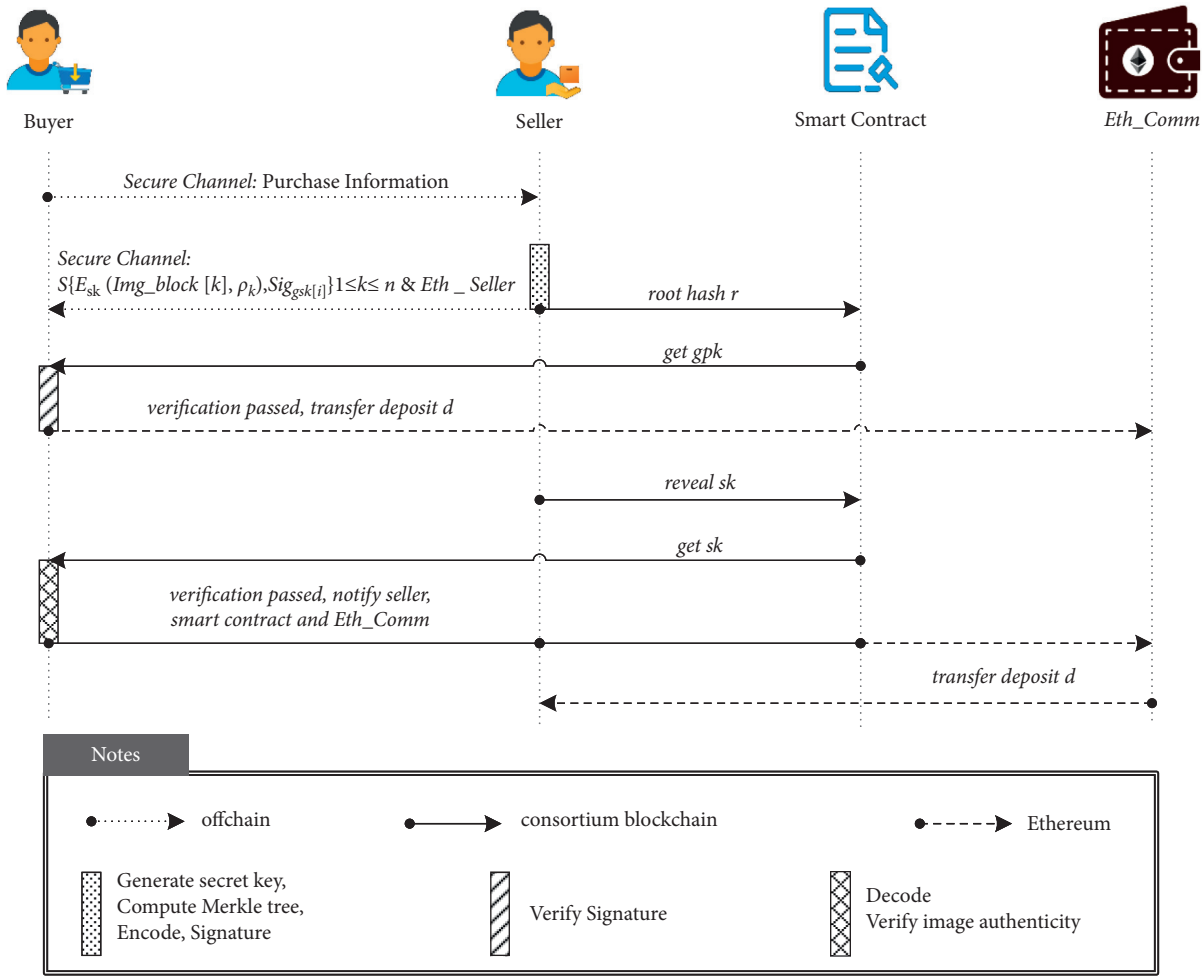

Figure 4: The details of the transaction stage.

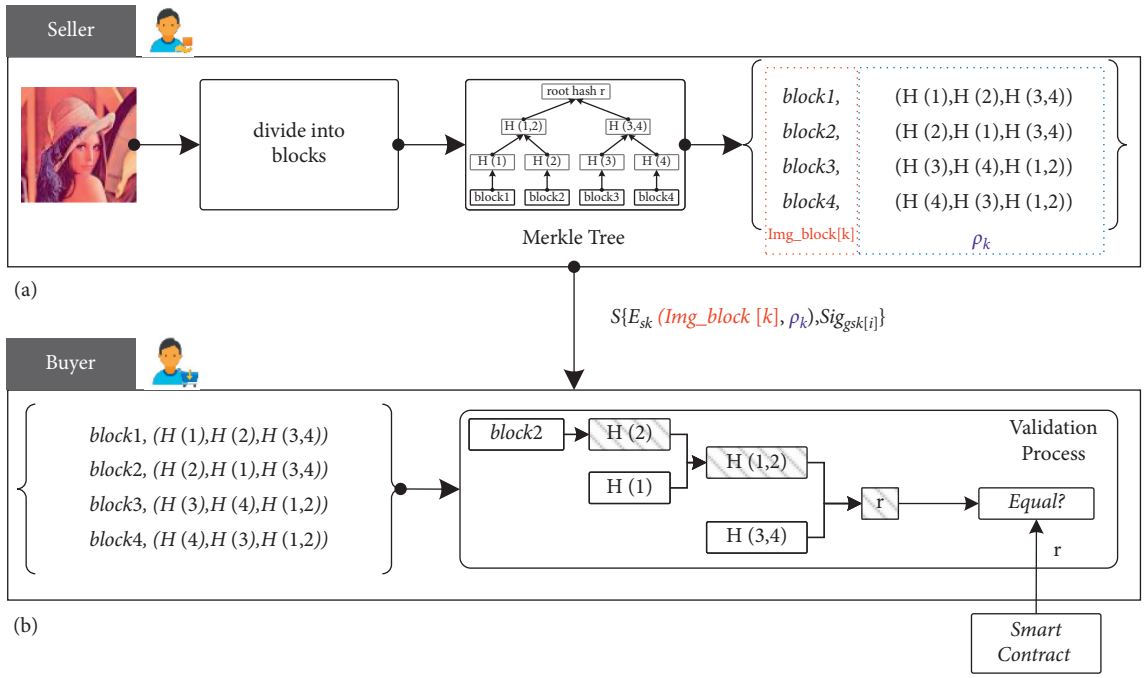

Figure 5: (a) The construction method of set S and (b) the verification process of block-verifiable image blocks.

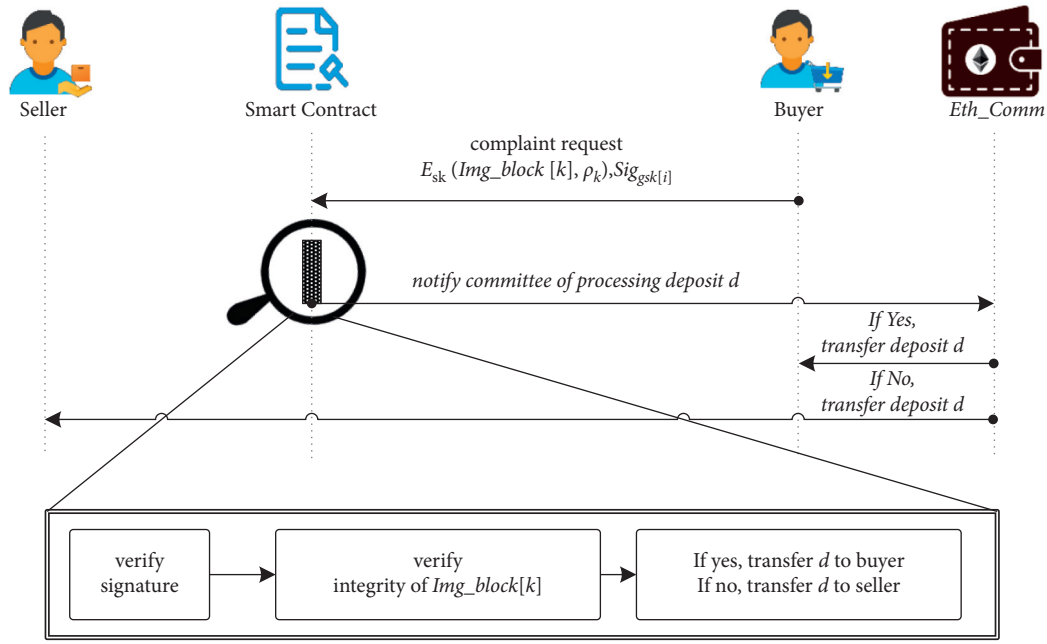

Figure 6: The details of the complaint stage.

effectively reduce the computational overhead of smart contract because the buyer only needs to submit the problematic encrypted image blocks, rather than all of encrypted image blocks. To describe clearly, we assume that the Img_block[3] does not pass the validation, so the buyer should send $E_{sk}(\text{img\_block}[3], \rho_3), \text{Sig}_{\text{gsk}[i]}$, to the smart contract.

The next issue is that the smart contract should double-check the buyer's submitted encrypted image block which is sent by the seller via offchain secure channel. Therefore, the smart contract should verify the signature firstly, then verify the integrity of img_block[3], and output the verification result lastly. The details of the above procedure are shown below.

(i) Verifying signature: the group public key gpk is used to verify the correctness of $\text{Sig}_{\text{gsk}[i]}$. If yes, this means that the complaint request message $E_{sk}(\text{img\_block}[3], \rho_3), \text{Sig}_{\text{gsk}[i]}$, came from the seller and has not been tampered by the buyer; if no, it means the seller has misbehavior; the committee is notified by smart contract to transfer the deposit d to the buyer after the expiration date.

(ii) Verifying encrypted image block(s): the verification method described in Figure 5(b) is used again. The new hash value $H(3)'$ of img_block[3] is computed; then, the new $H(3,4)'$ is calculated based on the connected $H(3)'$ and $H(4)$. The smart contract computes the new root hash $r\prime$ of the connection of $H(3,4)'$ and $H(1,2)$ and then compares r' with r primitively coming from the seller. If the result is equal, it proves that the seller is honest, so the smart contract notifies the committee to transfer the deposit d to the seller after expiration. Otherwise, it means that the seller has misbehaviour, and d should be refunded to the buyer after expiry.

During this stage, in order to reduce the computation cost of smart contract, the protocol divides a traded image into blocks and the number of blocks is decided by the seller. The self-verification image block can be constructed using the Merkle tree, and these self-verification image blocks are encrypted with the secret key *sk* and signed by the seller's private key gsk[*i*]. As a result, when the buyer finds a self-verification image block is wrong, he/she immediately initiates a complaint request to the smart contract. The smart contract just needs to verify this submitted image block(s), not the entire image, which can greatly reduce the volume of computation required for the smart contract and improve the efficiency of smart contract validation.

5.4. Revelation Stage. The main role of this phase is to trace and open the malicious user's identity because it is possible that there are some corrupt users who want to cheat their counterparties. For example, the seller tries to deceive the buyer with unreal images and the buyer does not want to pay after they received real images.

However, based on the properties of the complaint stage, the committee can identify malicious users. For maintaining the user-friendliness of the proposed protocol while resisting user's misbehavior, the committee keeps periodic statistics on the malicious behavior of each user and sets thresholds for misbehavior. If a user's malicious behavior exceeds the threshold during the period, like three times per week, the committee will reveal his or her identity. For example, if a seller sells fake images three times in a week, then Revealer_1 and Revealer_2 will work together to recover his or her identity for punitive purposes.

We follow the Open algorithm of the ESGS described in Figure 2 to construct the revelation subprotocol. In this algorithm, Revealer_1 and Revealer_2, respectively, holds the private key ξ_1 and ξ_2, which are used to jointly reveal the identity of the user. Hence, the revealing procedure is different from the Open algorithm of the BBS04, but they have the same security capabilities that have been proved in Section 4. Details of the revelation protocol are shown below. The two Revealers get the group public key gpk from the consortium blockchain and obtain the encrypted image block(s) and its signature $E_{\text{sk}}(\text{img\_block}[k], \rho_k), \text{Sig}_{\text{gsk}[i]}$ from the complained buyer. The proceeds are as follows. First, both of the Revealers verify whether $\text{Sig}_{\text{gsk}[i]}$ is a valid signature on $E_{sk}(\text{img\_block}[k], \rho_k)$. If this is the case, Revealer_2 calculates ${T_2}^{\xi_2}$ and sends it to Revealer_1. Then, Revealer_1 considers the first three elements (T_1, T_2, T_3) of $\text{Sig}_{\text{gsk}[i]}$ as a linear encryption and recovers the user's A as $A = T_3/({T_1}^{\xi_1} \cdot {T_2}^{\xi_2})$. At last, Revealer_1 looks up the user index corresponding to the identity A recovered from the signature.

6. Security and Privacy Analysis

In this section, we present an analysis of the proposed protocol to show that it effectively addresses the fairness and privacy threats described in Section 3.3.

6.1. Fairness Analysis. As mentioned in Section 3.3, there are three kinds of threats that can threaten the fairness of the protocol. Now, let us discuss how the protocol addresses these threats.

For Threat1, the seller has no ability to cheat the buyer with the unreal images. This is guaranteed due to the transparency property of blockchain and the image block authenticity verification property of the Merkle tree. Particularly, when a buyer wants to buy a special image from the seller, he or she needs to initiate a purchase request and send it to the seller; then, the seller begins to divide the traded image into blocks and calculate the Merkle tree and invokes the smart contract to upload the Merkle tree root hash r of the traded image to the consortium blockchain. Based on the transparency property of blockchain, all users (including the buyer) can view and download the root hash r from the consortium blockchain; this mechanism guarantees that the root hash r of the traded image cannot be modified once it is stored in one of the blocks of the consortium blockchain.

Afterward, each image block and its proof will be encrypted with a symmetric key chosen by the seller and signed with the seller's private key. All encrypted and signed image blocks and their proofs will be packaged into a set S. Then, the set S and the seller's Ethereum address Eth_Seller is sent to the buyer via the offchain secure channel. The buyer verifies the correctness of signatures firstly using the group public key gpk which is downloaded from the consortium blockchain platform. If the result is wrong, the transaction is terminated. Otherwise, the authenticity of the image block will be verified with its corresponding proof, and the verification method is described in Section 5.2. If any problem is found, the transaction will be terminated and the complaint will be launched by the buyer.

Additionally, the proposed protocol sets aside an expiration date for complaint operations, such as two days or three days, which can be set flexibly. The deadline gives enough time for the buyer to validate the image block. The buyer's deposit will not be transferred to the seller until the expiration date and the buyer has not initiated a complaint operation. In conclusion, all of the above operations and mechanisms ensure that the seller has no ability to cheat the buyer with the unreal images and earns illegal income.

For Threat2, in our design, the buyer cannot refuse to pay money after receiving the real images from the seller. This is a guarantee due to the deposit prepayment mechanism and the signature forgery resistance property of group signature schemes. Especially, when the buyer receives the Merkle tree root hash r of the traded image, he or she must transfer the deposit d to the committee's Ethereum address via the Ethereum blockchain. The transaction will only continue if the seller and committee recognize this transfer operation, otherwise, the transaction will be terminated.

If the transaction is successfully completed, the buyer receives the real image, the signature of the seller is valid and each image block is authentic. The deposit will be transferred to the seller's account at the end of the expiration date. To sum up, the deposit prepayment mechanism can prevent the buyer from refusing to pay in normal image transactions.

If the buyer discovers any problem in the data sent by the seller by the deadline, he or she will initiate a complaint request to the smart contract deployed on the consortium blockchain and submit the corresponding evidence. The smart contract strictly checks whether the evidence is credible, including verifying the seller's signature on the evidence to prevent buyers from forging evidence and verifying the authenticity and integrity of every image block to identify if the problematic image block is the same as the one reported by the buyer. Only if both of the above are met can the smart contract determine that the buyer's complaint is valid; then, the committee's Ethereum account will be notified to refund the deposit d to the buyer; the seller informed that he or she has committed a violation in the transaction with the buyer and that the behavior will be blacklisted. In conclusion, due to the signature forgery resistance property of group signature schemes, it is very difficult for a buyer to forge the seller's signature, so the credibility of the evidence submitted by the buyer to the smart contract can be guaranteed. Moreover, the deposit d will only be returned to the buyer if the smart contract identifies the problematic image block as the same as the one reported by the buyer. As a result, the protocol can guarantee that the buyer cannot refuse to pay money after receiving the real images from the seller.

For Threat3, the group committee cannot maliciously embezzle the deposits of transactions between buyers and sellers. This is guaranteed because the committee's Ethereum account Eth_Comm is jointly generated and controlled by all

members based on [22]. More specially, the deposits in the Eth_Comm can only be handled with the consent of all committee members. Moreover, when developing the practical system based on the proposed protocol, the committee members come from different interests, such as user representatives, consortium blockchain operators, and regulators, so that the possibility of collusion among committee members can be minimized.

6.2. Privacy Analysis. In the proposed protocol, privacy protection means to prevent user's identity information, user's linkability of transactions, and even user's image preferences from being leaked to attackers, including malicious users, intruders, and dishonest committee members. As described in Section 3.3, we next discuss how the protocol addresses these threats.

With respect to Threat4, attackers, including malicious users, intruders, and corrupt committee members, cannot detect the user's identity information based on their signatures. This is guaranteed by reason of the full-anonymity property of [9], which proves the security of this property through defining a corresponding experiment based on the experiment CCA of [24], CPA-full-anonymity, in which the adversary cannot query the opening oracle and the security and privacy is proved slightly.

With respect to Threat5, attackers have no ability to infer the user's linkability of transactions due to the anonymity property of the group signature. As we know, a group signature scheme is a special type of signature, each group member is able to sign message individually, but the public cannot know who signs it. In the proposed protocol, buyers and sellers sign on transaction data with their private key generated by the enhanced short group signature scheme, through consensus of miner nodes within the consortium blockchain; all transaction data will be stored in the blocks. Therefore, when malicious users, intruders, and corrupt committee members want to infer the user's linkability of transactions based on the data stored in the consortium blockchain, for a specific transaction, they cannot identify which user signed it, so it is not feasible to infer the transaction relationship between users.

With respect to Threat6, as mentioned above, attackers cannot identify which user signs a specific transaction, so they are unable to establish the relationship between the traded images and users and thus cannot analyze user behavior, such as user's image preference, based on image transaction history stored in the consortium blockchain. While some of the attackers may be able to analyze the transfer relationship of users' Ethereum addresses, this is also very difficult. This is because of the following two reasons. First, a user's Ethereum account is not related to his or her account of the consortium blockchain, and it is impossible for an attacker to correlate these two accounts of the same user. Second, all users' transfer operations are required to go through the committee account, which is helpful to mix up the transfer relationship between users and makes analysis more difficult.

7. Evaluation

In this section, we implemented the proposed protocol by developing an experimental prototyping system and evaluated its performance in multiple dimensions. Through extensive studies and tests, we chose the Hyperledger fabric [25] as the consortium blockchain, due to the fact that it is the most widely used open-source permissioned distributed ledger technology (DLT) platform. We utilized Pairing-Based Cryptography (PBC) library [26] to implement our enhanced short group signature protocol with conditional accountability described in Section 4. The algorithms Merkle Tree Creation, Merkle Tree Proof Generation, and Merkle Tree Verification are implemented based on an open-source java library [27]. In addition, the Ethereum Testnets [28] is used to support the deposit transfer among buyers, sellers, and the committee. Compared to the Ethereum Mainnet [29], the Testnets are used by protocol developers or smart contract developers to test both protocol upgrades as well as potential smart contracts in a production-like environment before deployment to Mainnet. When developers firstly apply to join the Testnets, it will give some Ether coins for testing so that developers can save money.

7.1. Prototyping Experiment. In this prototyping system, to meet user habits, we developed mobile applications for both sellers and buyers, and independent Web applications for Issuer, Revealer_1, and Revealer_2. More specifically, to prevent collusion of committee members, Issuer is set as the operator of fabric, like the association of photographers. Revealer_1 and Revealer_2 can be elected by all users in the system and updated regularly. The structure of the prototyping system is shown in Figure 7 and the operating environment for each component is shown in Table 1. For details about transaction procedures in the prototyping system, see Appendix A.

Mobile application: mobile application should be installed on both the buyer's and seller's smart devices. It is used to interact transaction data with the Hyperledger fabric, upload and download thumbnail with the IPFS, and transfer Ether with the Ethereum. In particular, this application implements thumbnail generation, upload and download preview, Merkle tree calculation, image block proof computation and verification, 128bit-AES key generation, encryption and decryption, group signature and verification, Ether transfer, etc.

Web application: web applications are mainly developed for the committee members Issuer, Revealer_1, and Revealer_2, which can make them manage the group easier and faster. For Issuer, web client primarily provides functions of ESGS parameter initialization, secret parameter γ generation and storage, public key generation, user's private key generation and issuance, interaction parameters h, u, and v with Revealer_1 and Revealer_2, and visual operation. For Revealer_1 and Revealer_2, the client mainly completes visual interaction,

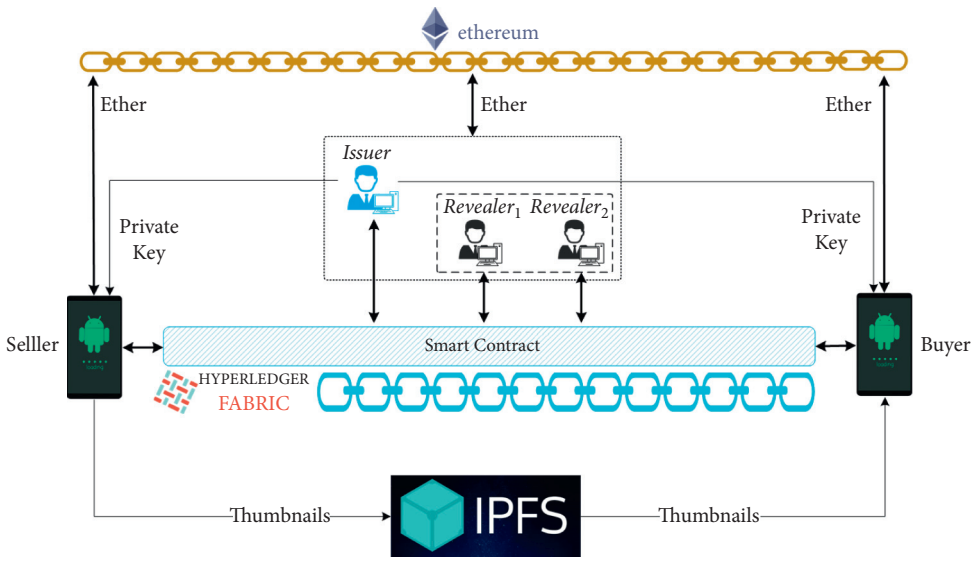

Figure 7: The structure of the prototyping system.

Table 1: Operating environment for the demo system.

Component	Type	CPU	RAM (G)	Bandwidth (Mbps)
Buyer	Android smart phone	MediaTekimensity	8	150
Seller	Android smart phone	MediaTekimensity	8	150
Issuer	Virtual machine	Intel(R)eon(R) 2 GHz	4	1
Revealer_1	Virtual machine	Intel(R)eon(R) 2 GHz	4	1
Revealer_2	Virtual machine	Intel(R) Xeon(R) 2 GHz	4	1
Fabric node	Virtual machine	Intel(R)Xeon(R) 2 GHz	4	1
IPFS node	Virtual machine	Intel(R) Xeon(R) 2 GHz	4	1

interaction with Issuer, and private key ξ_1 or ξ_2 generation and jointly responds and revealing the malicious user real identity.

Hyperledger fabric: we deployed a four-node fabric network based on its open-source code [30] v2.2 LTS release version. Every node is configured as the full node, which means it has a complete copy of the ledger. First, this mechanism can guarantee all transaction data are open and transparent to every user. Moreover, the rights of each node are the same, and the destruction of any node will not affect the security of the whole system or cause data loss. Lastly, the ledger data of each node is exactly the same, which means that data tampering of a single node is meaningless. All of these nodes communicate with each other on a peer-to-peer network.

Smart contract: the smart contract is implemented using the program language Solidity [31]. Through invoking the corresponding interface, the Issuer can upload group public key gpk, the seller can upload root hash r and 128bit-AES key sk, and the buyer can download gpk, r, and sk. Additionally, users can upload questionable encrypted image block(s) and the corresponding signature(s) to the smart contract. Once the smart contract receives this type of data, it will follow the steps of Section 5.3 to automatically determination who has misbehavior in the seller or the buyer and handle accordingly based on the result.

Ethereum Testnets: there are two kinds of accounts on the Ethereum test network. The first is the committee member co-generated account, which is used to keep deposits for users. The second is user's account and used to send and receive Ether.

7.2. Performance Evaluation. In this section, we evaluated metrics in the ESGS modules computation cost, fabric network upload and download latency, image block encryption and decryption computation cost, and smart contract judgment cost. As shown in Table 2, after 1000 tests, we calculated the average computational overhead for Issuer to generate secret parameter γ, user private key gsk and public key gpk, Revealer_1 to generate private key ξ_1, Revealer_2 to generate private key ξ_2, the user to sign image block, the verifier to verify signature, and Revealer_1 and Revealer_2 to jointly recover user's real identity. We can see that the average computational overhead of most of the modules is within milliseconds, and this means our enhanced short group signature can perfectly meet the practical requirements.

Moreover, as shown in Table 3, we calculated the maximum, minimum, and average upload and download latency of the network through 1000 tests. Similarly, the four-node fabric network can respond to the user's

Table 2: ESGS module performance.

	γ	gpk	ξ_2	ξ_1
Average value	27.878 μs	2.751 ms	5.008 ms	4.235 ms
	gsk	sign	verify	open
Average value	29.442 μs	8.091 ms	7.186 ms	1.796 ms

Table 3: Fabric network upload and download latency.

Latency	Maximum (ms)	Minimum (ms)	Average (ms)
Upload	711.00	373.00	511.60
Download	68.00	33.44	52.00

uploading and downloading operations in milliseconds, with a very good user interaction experience.

In addition, in order to evaluate the encryption and decryption computation cost of image block(s) on the mobile device, as shown in Figure 8, we tested the encryption and decryption overheads for the different number of image blocks. It is noticed that the encryption time is a little longer than decryption time, but they can all be done in tens of milliseconds. To evaluate the efficiency of smart contract, as shown in Figure 9, the computation cost of smart contract for the different number of image blocks which are complained by the buyers. We can see that the smart contract has the ability to complete the judgment task in tens of milliseconds. However, along with the number of block increases, the computation overhead increases as well.

8. Related Work

Fair exchange is a well-studied research problem. It has been shown that fair exchange is not possible without further assumptions about a trusted third party [32–34]. To circumvent this impossibility, researchers have studied weaker security optimistic models, in which the third trusted party (TTP for short) is only consulted if one party deviates from the expected behavior [35, 36]. We can consider the smart contract-based solution as a variant of the optimistic protocol, where the smart contract plays the role of the TTP. In particular, KüpçüLysyanskaya [5] consider a similar use case, file sharing, but it implements very different security than what we have done. (1) The arbiter uses a cut-and-choose approach so that the probability of not finding a cheater is nonnegligible for a corrupted file. (2) The workload of the arbiter is high due to the cut-and-choose, leading to high costs in the smart contract setup. In contrast, our solution has only a negligible error rate and a small financial cost. We also emphasize that the cost models for arbitrators and smart contracts are very different.

Recently, there has been a significant amount of work on the use of cryptocurrencies such as Bitcoin and Ethereum to achieve fairness in blockchain-based protocols [6, 8, 37–39]. As already discussed, fairness is guaranteed by relying on the smart contract executed over decentralized cryptocurrencies, where the contract takes the role of an external judge that completes the exchange in case of disagreement [6]. In addition to the smart contract, perceptual hashing is

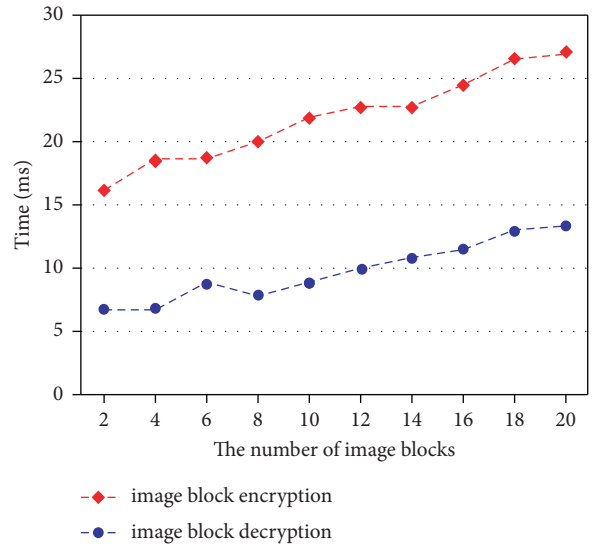

Figure 8: Image block encryption and decryption computation cost.

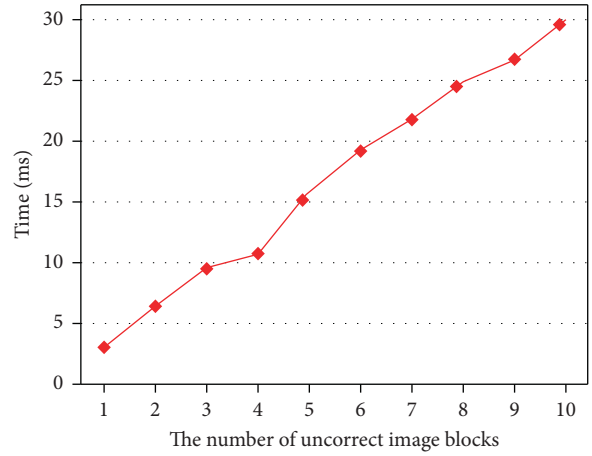

Figure 9: Smart contract judgment computation cost.

leveraged to automatically detect and reject tampered images that are perceptually similar to images that are already present on the marketplace [7]. In addition, Zhao et al. designed an image network copyright transaction protection approach based on blockchain technology [8] so that the entire copyright transaction process is protected and the attribute identification of image content is identified. Unfortunately, these works only consider solving the fairness issue, while addressing the privacy issue at the same time remains to be an open problem.

Finally, we propose to build a practical image trading system that can provide a guarantee on achieving both fairness and privacy protection in an image trading system. In terms of protecting privacy, we extend a short group signature scheme [9] to protect the user's identity, linkability of transactions, and image preference. At the same time, group signature is used in many blockchain-based social areas. Qiao utilized the group signature to prevent user's privacy disclosure in blockchain supply chain transactions

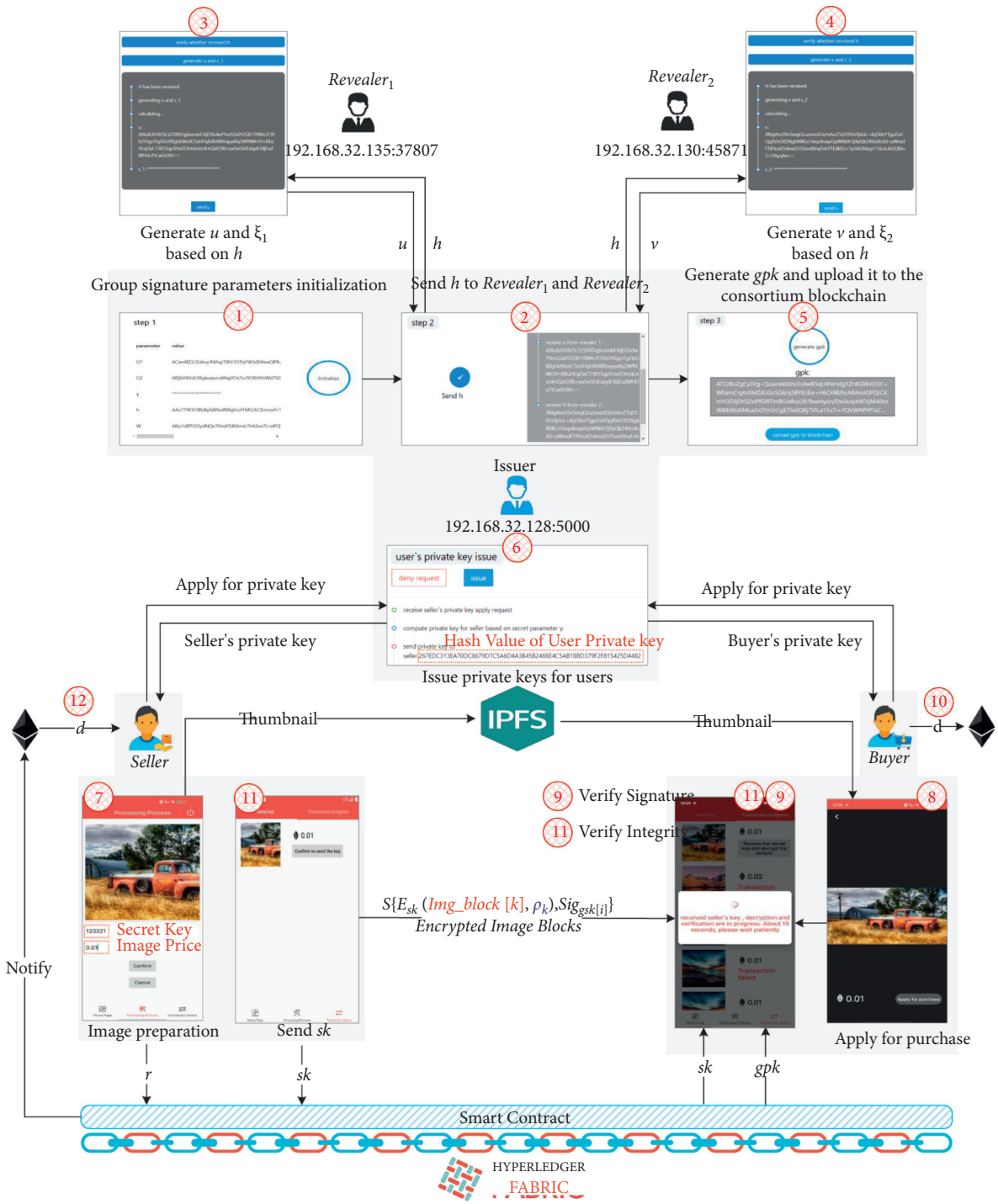

Figure 10: Image transaction procedures with screenshots.

[40]. In smart grids, Chen et al. proposed a privacy-preserving scheme based on blockchain and group signature to protect users' identities, behavior, and preferences [41]. Zhang et al. designed a scheme called BFSP, which is a fair, reliable, and privacy-preserving blockchain-based smart parking. Particularly, group signatures, bloom filters, and vector-based encryption are used to protect the user's privacy, the smart contract is utilized to make sure fairness, and blockchain is leveraged to guarantee reliability.

9. Conclusion and Future Work

In this paper, we propose a fair and privacy-preserving image trading system based on blockchain and group signature. We extend the short group signature to our proposed system so that a verifier is able to verify a signer's signature on transaction data, while the singer's identity is kept private from the verifier. Moreover, the proposed protocol can protect malicious users and intruders from inferring any

private information about a particular user. In addition, fairness in trading is achieved by a fair image trading protocol that is constructed by utilizing blockchain and the Merkle tree. To evaluate the efficiency of the proposal, we design and build a practical image trading system to test its multifaceted performance.

There are two interesting problems we will continue to study for our future work. One of them is the applicability of the proposed protocol to other digital goods, which means we need to analyze the characteristics of each type of digital goods and optimize our scheme according to their characteristics so that it can be both secure and practical. Another problem for our future work is to explore the feasibility of decentralized stablecoins and NFT in the proposed scheme because Ether is a nonstable cryptocurrency and its price fluctuates too much, which can cause unnecessary losses to users. Moreover, NFT [42] is a very popular new technology used for copyright validation and trading of digital artworks.

Appendix

A. Image Transaction Procedures with Screenshots

In this section, we combine transaction procedures and the screenshots of the Web applications (used by committee members) and the Android applications (used by the seller and the buyer) to depict the entire image transaction in Figure 10. Particularly, the circular icons with numbers and light red grid lines represent the specific trading steps. The text above, below, or in the upper left corner of each screenshot identifies the work completed in this step. The Ethereum Testnets is represented by the Ethereum icon.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Le Wang and Xiaodong Lin were supported by Natural Sciences and Engineering Research Council of Canada (NSERC). Xuefeng Liu was supported by the Key Research and Development Programs of Shaanxi, under Grant 2019ZDLGY13-04.

References

[1] Shutterstock, 2003, https://www.shutterstock.com/zh/home/.
[2] iStockphoto, 1999, https://www.istockphoto.com/.
[3] Fotolia, 2004, https://www.fotolia.com/.
[4] Dreamstime, 2000, https://www.dreamstime.com/.
[5] A. Küpçü and A. Lysyanskaya, "Usable optimistic fair exchange," in *Proceedings of the Cryptographers' Track at the RSA Conference*, pp. 252–267, Springer, San Francisco, CA, USA, March 2010.
[6] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: how to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 967–984, Toronto, Canada, October 2018.
[7] R. Mehta, N. Kapoor, S. Sourav, and R. Shorey, "Decentralised image sharing and copyright protection using blockchain and perceptual hashes," in *Proceedings of the 2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, pp. 1–6, IEEE, Bangalore, India, January 2019.
[8] C. Zhao, M. Liu, Y. Yang, F. Zhao, and S. Chen, "Toward a blockchain based image network copyright transaction protection approach," in *Proceedings of the International Conference on Security with Intelligent Computing and Big-Data Services*, pp. 17–28, Springer, Guilin, China, 2018.
[9] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proceedings of the Annual International Cryptology Conference*, pp. 41–55, Springer, Santa Barbara, CA, USA, August 2004.
[10] Merkle tree: 2021, https://en.wikipedia.org/wiki/Merkle_tree.
[11] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Proceedings of the Annual International Cryptology Conference*, pp. 255–270, Springer, Kyoto, Japan, December 2000.
[12] J. Benet, "Ipfs-content addressed, versioned, p2p file system," 2014, https://arxiv.org/abs/1407.3561.
[13] P. Labs, 2016, https://ipfs.io/#how-it-works.
[14] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2019, https://git.dhimmel.com/bitcoin-whitepaper/.
[15] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: a survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.
[16] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Business & Information Systems Engineering*, vol. 59, no. 3, pp. 183–187, 2017.
[17] C. Dannen, *Introducing Ethereum and Solidity*, Springer, New York, NY, USA, 2017.
[18] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: a deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, 2019.
[19] L. Liu, J. Feng, Q. Pei et al., "Blockchain-enabled secure data sharing scheme in mobile-edge computing: an asynchronous advantage actor–critic learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2342–2353, 2020.
[20] V. Buterin, "A next-generation smart contract and decentralized application platform," *White Paper*, vol. 3, p. 37, 2014.
[21] T. L. Foundation, "The fabric tutorials," 2021, https://hyperledger-fabric.readthedocs.io/en/latest/tutorials.html.
[22] J. Doerner, Y. Kondi, E. Lee, and A. Shelat, "Threshold ECDSA from ECDSA assumptions: the multiparty case," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1051–1066, IEEE, San Francisco, CA, USA, May 2019.
[23] Android, 2021, https://developer.android.google.cn/reference/android/graphics/Bitmap?hl=en#compress(android.graphics.Bitmap.CompressFormat,%20int,%20java.io.OutputStream.
[24] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions," in *Proceedings of the International Conference on the Theory and*

Applications of Cryptographic Techniques, pp. 614–629, Springer, Warsaw, Poland, May 2003.
[25] T. L. F. Project, 2020, https://www.hyperledger.org/use/fabric.
[26] B. Lynn, 2006, https://crypto.stanford.edu/pbc/.
[27] https://github.com/quux00/merkle-tree.
[28] T. E. Community, 2021, https://ethereum.org/en/developers/docs/networks/#testnets.
[29] https://ethereum.org/en/developers/docs/networks/#mainnet.
[30] T. L. Foundation, 2021, https://github.com/hyperledger/fabric#releases.
[31] Solidity, 2021, https://docs.soliditylang.org/en/v0.8.4.
[32] H. Pagnia and F. C. Gärtner, "On the impossibility of fair exchange without a trusted third party," Technical Report D, Citeseer, Princeton, NJ, USA, 1999.
[33] A. C.-C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (Sfcs 1986)*, pp. 162–167, IEEE, Toronto, Canada, October 1986.
[34] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the 19th Symposium on Theory of Computing (STOC)*, pp. 218–229, ACM, New York, NY, USA, 1987.
[35] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 591–606, Springer, Espoo, Finland, June 1998.
[36] C. Cachin and J. Camenisch, "Optimistic fair secure computation," in *Proceedings of the Annual International Cryptology Conference*, pp. 93–111, Springer, Kyoto, Japan, December 2000.
[37] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, pp. 443–458, IEEE, San Jose, CA, USA, May 2014.
[38] R. Kumaresan and I. Bentov, "Amortizing secure computation with penalties," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 418–429, Vienna, Austria, October 2016.
[39] R. Kumaresan, V. Vaikuntanathan, and P. N. Vasudevan, "Improvements to secure computation with penalties," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 406–417, Vienna, Austria, October 2016.
[40] S. Qiao, *Group Signatures for Preserving Anonymity in Blockchain Supply Chain Transactions*, North Carolina State University, Raleigh, NC, USA, 2021.
[41] X. Chen, J. Shen, Z. Cao, and X. Dong, "A blockchain-based privacy-preserving scheme for smart grids," in *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, pp. 120–124, Shanghai, China, March 2020.
[42] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (Nft): overview, evaluation, opportunities and challenges," 2021, https://arxiv.org/abs/2105.07447.