Received January 17, 2022, accepted February 8, 2022, date of publication February 18, 2022, date of current version March 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3152765

A Secure E-Coupon Service Based on Blockchain Systems

JONGBEEN HAN[1], **YONGSEOK SON**[2], **AND HYEONSANG EOM**[1]
[1]Department of Computer Science and Engineering, Seoul National University, Seoul 08826, Republic of Korea
[2]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea

Corresponding author: Yongseok Son (sysganda@cau.ac.kr)

This work was supported in part by the BK21 FOUR Intelligence Computing (Department of Computer Science and Engineering, SNU) funded by the Ministry of Education (MOE, South Korea); in part by the National Research Foundation of Korea (NRF) under Grant 4199990214639; and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government through MSIT under Grant NRF-2021R1F1A1063438, Grant 2021R1C1C1010861, and Grant KIAT-P0012724.

ABSTRACT As the popularity of e-commerce grows, an electronic coupon (e-coupon) is widely used due to its convenience and portability. In most e-coupon services, the information of e-coupons is managed on a centralized server. However, e-coupon services are often vulnerable to security issues because of centralization. For example, when the e-coupon information which is stored in a centralized e-coupon server is forged, it becomes difficult to match the user and the e-coupon's owner, and an expired e-coupon can be used repetitively (i.e., double-spending). To handle this issue, we propose a new e-coupon service by exploiting a blockchain system to improve the security of the service. To do this, we first design a server to enable the e-coupon service and communicate with the blockchain system. Second, we devise a smart contract on the blockchain system to provide integrity of the e-coupon business logic and the e-coupon's information. We implemented the proposed service on an Ethereum-based blockchain system. The experimental results show that our proposed service improves higher security with a minor performance overhead compared with an existing e-coupon service.

INDEX TERMS E-coupon, blockchain, smart contract, security.

I. INTRODUCTION

With the growth of the electronic commerce market, electronic coupons (e-coupons) are being adapted as an effective marketing tool [1], [2]. The electronic nature of e-coupons not only provides coupon providers, such as sellers and marketers, with an efficient way of management but is also convenient for customers. For example, since an e-coupon is provided by digital code, e-coupon providers can distribute the e-coupon to the customers online and easily collect statistics such as downloading and using e-coupons. Also, customers can easily manage the e-coupons via their mobile devices or PCs. Because of these advantages of e-coupons, Global Mobile Coupons Market 2016-2020 reports that the global mobile coupon market will grow to a compound annual growth rate (CAGR) of 73.14% over 2016-2020 [3].

Although the e-coupon market evolves and an e-coupon provides several benefits, there are some challenges. For easy management, most e-coupon services manage e-coupon information in a centralized system. When an e-coupon is used, the e-coupon is validated by using the information in the centralized database system. However, the information can be easily manipulated by an administrator due to the centralization nature so that there can be a forgery and fraudulent usage of an e-coupon. For example, an e-coupon may be redeemed multiple times (double spending), or a malicious attacker may manipulate the discount rate. In the United States, PennLive estimates real e-coupon crime costs to be around \$300-\$600 million dollars per year [4].

To enhance the security of e-coupons, Hsueh *et al.* [5] propose an e-coupon system using a hash chain which is combined with blockchain technology. Our study is in line with the work in terms of providing the integrity of e-coupon information via blockchain technology. In contrast, furthermore, we provide the integrity of operations (e.g., managing e-coupons, etc.) as well as the integrity of e-coupon information by devising a secure smart contract.

In this paper, we propose an e-coupon service based on a blockchain system to improve the security of the service. To do this, we first design a server to enable e-coupon service

The associate editor coordinating the review of this manuscript and approving it for publication was Zhangbing Zhou.

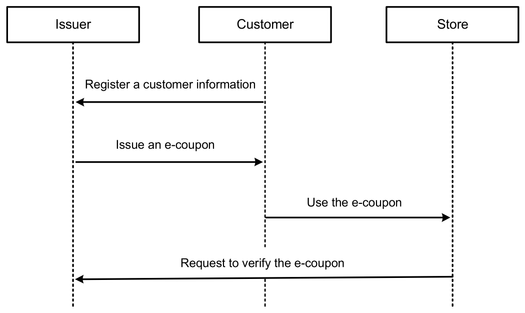

Issuer
Customer
Store
Register a customer information
Issue an e-coupon
Use the e-coupon
Request to verify the e-coupon

FIGURE 1. Example of centralized e-coupon service.

and communicate with the blockchain system. Second, we devise an e-coupon smart contract in the blockchain system to provide the integrity of the operations (i.e., business logic code [6]) and e-coupon information. In addition, we deploy an e-coupon smart contract to the blockchain automatically for user convenience.

We apply and implement the proposed service on the Quorum blockchain system [7] for the security of e-coupon information and business logic code (i.e., downloading, giving, and using an e-coupon). Experimental results demonstrate that the proposed service improves security and has a minor performance overhead compared with existing services. The contributions of our work are as follows:

- We investigate the existing e-coupon processing mechanism in terms of security and e-coupon trading.
- We propose a new service that enables secure e-coupon trading via an e-coupon smart contract on a blockchain system and deploys the e-coupon smart contract automatically.
- We demonstrate that the proposed e-coupon service is more secure compared with the existing services.

The rest of this paper is organized as follows. Section II describes the background and motivation. Section III pr-esents the design and implementation of the proposed service. Section IV shows the experimental results. Section V discusses the related work. Section VI concludes this paper.

II. BACKGROUND AND MOTIVATION

A. SERVING E-COUPONS ON A CENTRALIZED SERVER

With the expansion of smartphones and the development of e-commerce, the usage of e-coupon is increasing [8]–[10]. Unlike traditional paper coupons, the e-coupons allow coupon providers to collect and manage the coupon information easily (e.g., the number of coupons, the number of downloads, lists of customers, or whether coupons have been used). In addition, e-coupons provide customers to use and manage the e-coupons via website or smartphone [11]. As shown in Figure 1, most e-coupons are provided by a centralized server for managing the e-coupon information since the information on the centralized server can be managed and collected efficiently. The e-coupon services have the following process to redeem an e-coupon:

1) To download an e-coupon, a customer registers the customer information to an e-coupon issuer.
2) The customer downloads an e-coupon from the issuer via a mobile device or PC.
3) When a customer uses the e-coupon, the customer sends the e-coupon to the store (i.e., e-coupon provider).
4) The store requests the issuer to verify the e-coupon. And the issuer verifies the validity of this e-coupon according to the database.

In the process of e-coupon services, verifying an e-coupon is the most important task because the forged or manipulated e-coupons by malicious attacks lead to a financial problem. To prevent this forgery of e-coupons, previous works [2], [12]–[15] propose mechanisms to validate the e-coupons via message-digest algorithm 5 (MD5), message authentication code (MAC), and one-way hash function. However, they do not provide the techniques to prevent the falsification of the information on a centralized server. In other words, forgery of e-coupons does not occur during data transmission, meanwhile, forgery of e-coupon information stored in the e-coupon database can occur when using the above techniques. In addition, an administrator of the e-coupon server can modify any e-coupon information for his/her own benefits. Therefore, our study aims to introduce a new e-coupon service that dose not allow unauthorized forging of e-coupons and manipulation of information on the e-coupon server. To this end, we devise an e-coupon service based on a blockchain system.

B. BLOCKCHAIN

The blockchain technology [16], [17] is an attractive solution to address security issues (e.g., data integrity) in distributed systems. To address the issues, most blockchain systems maintain a time-stamped chain of the blocks with every participating user. The block consists of a block header and block body. The block body includes transactions. The block header includes a previous block hash and the root of the Merkle tree [18] generated with the transactions of the block body, etc. The blocks are chained together by the previous block hash and a new block can only be appended to the end of the chain. With these features, the transactions stored in the blockchain can not be updated or deleted due to the chain of historical transactions. Thus, a blockchain system can provide the Byzantine fault tolerance (BFT) [19] and inter-individual transfers without an intermediate entity.

Among the blockchain systems, Ethereum is a one of popular blockchain-based platform that provides smart contracts. A smart contract is a set of promises in a digital form which users perform [20]. A smart contract can run consistently on all the Ethereum nodes without the arbitration of a trusted entity because the business logic code and status value (which is the result of a smart contract) of the smart contracts are stored in the blockchain [21], [22]. With these features, users can construct distributed applications (DApps) with anonymity, transparency, immediacy, and high-level security via the smart contract. Although smart contracts enhance

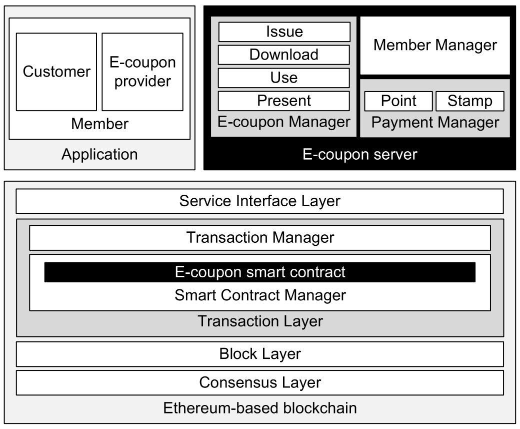


FIGURE 2. Overall architecture of the e-coupon service.

security, there are shortcomings. For example, it is difficult for users to manually build a smart contract, which can decrease the usability of the smart contract. Thus, we devise a secure and highly usable e-coupon service by exploiting the high-level security of blockchain and deploying an e-coupon smart contract automatically.

III. DESIGN AND IMPLEMENTATION

To achieve higher security and usability of an e-coupon service, we propose a new secure e-coupon service by exploiting blockchain and smart contracts. In our service, we provide the integrity of the business logic and e-coupon's information by adopting blockchain.

A. OVERVIEW

Figure 2 shows the overall architecture of the proposed e-coupon service. The e-coupon service consists of three layers: an application, e-coupon server, and Ethereum-based blockchain. The application is similar to the existing applications except for signing transactions and sending the transactions to the blockchain. The e-coupon server is a broker that delivers member information and e-coupon information stored in the blockchain to the application. The Ethereum-based blockchain validates e-coupon transactions and stores the data into the blockchain. Also, e-coupon smart contracts operate via the Ethereum virtual machine (EVM), a sandboxed virtual machine implicitly enclosed within each complete Ethereum node, capable of executing the contract bytecode.

We consider blockchain architecture for improving the performance of the blockchain in e-coupon service. For example, Ethereum blockchain stores all transaction states to a smart contract by using a tree structure (i.e., account storage trie). Therefore, when the size of stored states increases, the tree size also increases. This result can increase the tree search time to store and retrieve the state information. Therefore, this scheme may show performance degradation in storing or retrieving the e-coupon state information. On the other hand, we provide each smart contract for each e-coupon provider, and each tree in each smart contract manages its own e-coupon state information. This scheme reduces the tree depth and so that it improves performance in storing and retrieving e-coupon state information.

In addition, we enable to easily manage the e-coupon and reduce the cost of development by making and deploying e-coupon smart contracts automatically. To do this, the proposed e-coupon service provides a smart contract template to e-coupon providers. With this template, the e-coupon providers can easily create a coupon smart contract and automatically deploy the smart contract to the blockchain without writing a new smart contract by configuring the e-coupon information (i.e, the quantity of the coupon, coupon validity period, coupon type, discount amount, etc.). Therefore, it can provide convenience to e-coupon providers and reduce the cost of building the smart contract.

B. E-COUPON SERVER

1) E-COUPON MANAGER

The e-coupon manager provides an interface to deploy an e-coupon smart contract, get an e-coupon list, download an e-coupon, use the e-coupon, and provide the e-coupon to customers. Furthermore, the manager communicates with the blockchain to obtain and store e-coupon information. For example, when an e-coupon provider issues an e-coupon, the e-coupon provider requests to deploy an e-coupon smart contract to the e-coupon manager. Then, the e-coupon manager generates the transaction that deploys the e-coupon smart contract on the blockchain. After then, it stores the e-coupon information and the smart contract address in the server's database. By using the information stored in the database, the e-coupon manager provides e-coupon information to customers. Note that all the e-coupon data stored in the server is only used for displaying to the application. The data modification must be performed via transaction processing based on the data in blockchain.

We classify e-coupons into two types which our service supports. The first type is a *discount coupon*, which e-coupon providers use to attract new customers or provide a discount on a product or service. This coupon can be free, depending on the choice of the e-coupon provider. The second type is a *reserve coupon*, which is used to increase the loyalty of existing customers. Furthermore, the *reserve coupon* can be a point or stamp. The point is used as cash, and the stamp is used when the quantity set by an e-coupon provider is satisfied for redeeming goods or services. The reserve coupon is related to payments because customers can obtain the coupon when they purchase goods or services.

2) MEMBER MANAGER

The member manager manages user information for communicating between the application and the blockchain. For example, the manager maps the wallet address in the Ethereum-based blockchain to the user's ID in the applications (e.g., e-coupon provider or customer). This is because applications perform the transactions based on the wallet

Algorithm 1 An Example of an E-Coupon Smart Contract

```
 1: function downloadCoupon(mgs)
 2:     /* require() is a verification function for a given
        condition */
 3:     require(remain_coupons > 0)
 4:     require(coupons[msg.sender].downloaded == false)
 5:     require(expirationDate >= now)
 6:
 7:     /* Set a customer and modify the number of coupons
        */
 8:     coupons[msg.sender] = Coupon({
 9:         downloaded: true,
10:         pendding: false
11:     });
12:     remain_coupons = remain_coupons.sub(1);
13:
14:     DownloadCouponEvent(msg.sender)
15: end function
16:
17: function requestCoupon(msg)
18:     require(coupons[msg.sender].downloaded == true)
19:     require(coupons[msg.sender].pendding == false)
20:     require(expirationDate >= now)
21:     require(startDate <= now)
22:
23:     /* Modify a state to use the e-coupon from the cus-
        tomer */
24:     coupons[msg.sender].pendding = true;
25:
26:     RequestCouponEvent(msg.sender)
27: end function
28:
29: function confirmCoupon(msg, customer)
30:     require(msg.sender == owner)
31:     require(coupons[customer].downloaded == true)
32:     require(coupons[customer].pendding == true)
33:
34:     /* Confirm the use of e-coupon from the customer */
35:     coupons[customer].pendding = false
36:     coupons[customer].downloaded = false
37:
38:     ConfirmCouponEvent(customer, now)
39: end function
```

address in the blockchain, as well as the user's ID on the server. In addition, the manager maps the wallet addresses of the e-coupon provider and customer to the smart contract addresses of the e-coupon. To provide the privacy of the members, we do not upload member information on the blockchain. Instead, we use the member information mapped between the blockchain and the e-coupon server.

3) PAYMENT MANAGER

The payment manager provides an interface to save information paid by a customer in the blockchain and search the information. Also, the manager manages the history of reserve e-coupons created when customers pay for goods/services or redeem e-coupons to purchase corresponding goods or services. For instance, when a customer pays for a product or service, the payment manager creates a transaction related to the payment to the e-coupon smart contract for saving reserve e-coupons to the blockchain. And the manager stores payment information in a database and serves the history to customers.

C. E-COUPON SMART CONTRACT IN ETHEREUM-BASED BLOCKCHAIN

We exploit the blockchain to prevent the forgery of e-coupon information via a consensus algorithm. Also, the smart contract stored in the blockchain does not allow falsification because all nodes participating in the blockchain network perform the smart contract's business logic whether the logic is correct or not. By exploiting this feature of the smart contract, we guarantee the integrity of the e-coupon business logic. The business logic of an e-coupon includes e-coupon operations (e.g., issue, download, redeem, gift, etc.).

Algorithm 1 shows an example of how we guarantee integrity using a smart contract for e-coupons. Specifically, this algorithm describes the main business logic of downloading and using a discount e-coupon. `downloadCoupon()` is a function in the smart contract to download an e-coupon according to a customer request based on the transaction information (Algorithm 1, lines 1-15). To download an e-coupon, `downloadCoupon()` first validates whether the e-coupon exists or not (line 3). If the e-coupon exists, `downloadCoupon()` validates whether the customer already has the e-coupon or not (line 4). If the customer does not have the e-coupon, `downloadCoupon()` finally validates whether the e-coupon is expired or not (line 5). When the transaction satisfies all the above conditions, `downloadCoupon()` generates a new state for the e-coupon which identifies the customer who has downloaded the e-coupon (lines 8-11). Also, it reduces the number of e-coupons remaining (line 12). Subsequently, `downloadCoupon()` generates an event of downloading e-coupon used by the e-coupon server to track changes in the state of the e-coupon (line 14). After calling `downloadCoupon()`, the changed states are stored in the blockchain, which can not be falsified.

To redeem an e-coupon, there are two functions which are `requestCoupon()` and `confirmCoupon()` (Algorithm 1, lines 17-39). `requestCoupon()` is a function to use an e-coupon from a customer request (lines 17-27). `requestCoupon()` first validates whether the customer has the e-coupon or not (line 18). If the customer has the e-coupon, `requestCoupon()` validates whether the e-coupon has been used or not (line 19). If the e-coupon has not been used, `requestCoupon()` finally validates whether the e-coupon is available or not (lines 20-21). The transaction is rejected even if a single condition is not satisfied. Otherwise, `requestCoupon()` modifies the state to

TABLE 1. Notation in our e-coupon service.

Notion	Description
UID	Identifier of a registered user (i.e., an e-coupon provider or customer)
C	E-coupon information (e.g., price, the number of e-coupons, start date, expiration date, etc.)
Addr_c	Customer's wallet address
Addr_{ecp}	E-coupon provider's wallet address
Addr_{ecsc}	E-coupon smart contract address
Key_c	Customer's private key
Key_{ecp}	E-coupon provider's private key
Pub_c	Customer's public key
Pub_{ecp}	E-coupon provider's public key
Tx_u	Unsigned transaction
Tx_{s_c}	Signed transaction by the customer
Tx_{s_ecp}	Signed transaction by the e-coupon provider

use the e-coupon based on the customer requests (line 24). Next, the `requestCoupon()` generates an request event for the customer to use the e-coupon (line 26). Subsequently, the event is used by the e-coupon server to notify the e-coupon provider that there is a request for the use of e-coupons (line 26). `confirmCoupon()` is a function that approves the customer's request to use the e-coupon (lines 29-39). First, `confirmCoupon()` validates whether the message sender (who sent a transaction) is the owner of the e-coupon smart contract and the customer has requested to use the e-coupon (lines 30-32). After this verification, `confirmCoupon()` confirms the use of the e-coupon (lines 35-36). Finally, `confirmCoupon()` generates an event that confirms the use of the e-coupon. The event is used by the e-coupon server to notify the e-coupon provider and the customer that the e-coupon has been applied (line 38). Note that all statements in Algorithm 1 are performed among nodes participating in the blockchain network. Thus, whenever each statement is executed, the nodes reach a consensus to modify or check the state. This can guarantee the integrity of the business operations.

The process for reserve e-coupons is similar to that for the discount e-coupon. Exceptionally, for the reserve e-coupons, the smart contract manages the payment, the reserve e-coupon (i.e., point or stamp) related to an e-coupon provider, redeemable goods or services with the reserve e-coupon, etc. For example, when a customer pays for a product or service, the smart contract calculates the number of coupons provided based on the configuration of the e-coupon provider. In addition, the smart contract enables all types of e-coupons to be transferred between customers.

D. PROCESSING E-COUPON OPERATIONS

In this section, we explain how to process each e-coupon operation between application, e-coupon server, and Ethereum-based blockchain. Table 1 lists the notations we use in our e-coupon service. As shown in the table, `UID` is a registered user identifier stored in the e-coupon server to identify a member. It is used when the e-coupon server provides the e-coupon information to a corresponding member.

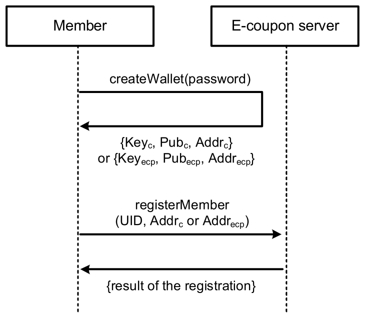

Member
E-coupon server
createWallet(password)
{Key$_c$, Pub$_c$, Addr$_c$} or {Key$_{ecp}$, Pub$_{ecp}$, Addr$_{ecp}$}
registerMember (UID, Addr$_c$ or Addr$_{ecp}$)
{result of the registration}

FIGURE 3. Registration of a new member (i.e., creating a wallet).

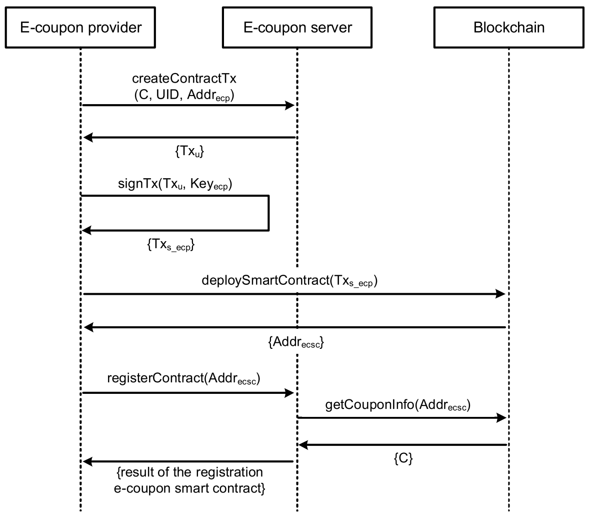

E-coupon provider
E-coupon server
Blockchain
createContractTx (C, UID, Addr$_{ecp}$)
{Tx$_u$}
signTx(Tx$_u$, Key$_{ecp}$)
{Tx$_{s\_ecp}$}
deploySmartContract(Tx$_{s\_ecp}$)
{Addr$_{ecsc}$}
registerContract(Addr$_{ecsc}$)
getCouponInfo(Addr$_{ecsc}$)
{C}
{result of the registration e-coupon smart contract}

FIGURE 4. Issuing an e-coupon smart contract.

`C` represents the e-coupon information stored in the smart contract of the blockchain. The initial e-coupon information is determined by an e-coupon provider. This information is updated when the e-coupon is downloaded, used, or transferred as gift. Addr_c and Addr_{ecp} are the external owned address (EOA) of the customer and the e-coupon provider, respectively. They are the wallet's addresses used in the blockchain when e-coupons are distributed, downloaded, used, or given by e-coupon providers or customers. Addr_{ecsc} is a contract address (CA) about an e-coupon smart contract and a target address to execute the business logic of the e-coupon. Key_c and Key_{ecp} are the private keys of the customer and the e-coupon provider, respectively. The keys are used to sign an unsigned transaction (Tx_u) including the e-coupon information (`C`) without a signature when an e-coupon smart contract is deployed, downloaded, or used. Tx_{s_c} and Tx_{s_ecp} refer to the transaction with a signature by an e-coupon provider or customer. The Blockchain verifies Tx_{s_c} and Tx_{s_ecp} using Pub_c or Pub_{ecp} which is a public key related to Addr_c and Addr_{ecp}. The process of the e-coupon service consists of five steps: (1) registration of a new member, (2) issuing an e-coupon smart contract, (3) downloading an e-coupon, (4) gifting an e-coupon, and (5) using an e-coupon. We will explain each step as follow.

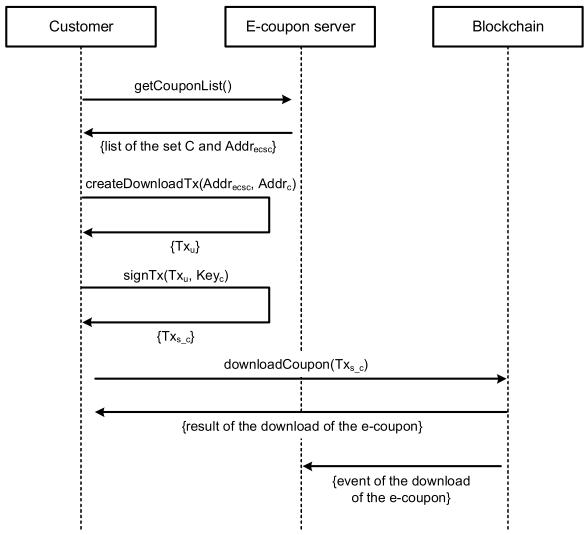

Customer
E-coupon server
Blockchain
getCouponList()
{list of the set C and $Addr_{ecsc}$}
createDownloadTx($Addr_{ecsc}$, $Addr_c$)
{Tx_u}
signTx(Tx_u, Key_c)
{Tx_{s_c}}
downloadCoupon(Tx_{s_c})
{result of the download of the e-coupon}
{event of the download of the e-coupon}

FIGURE 5. **Downloading an e-coupon.**

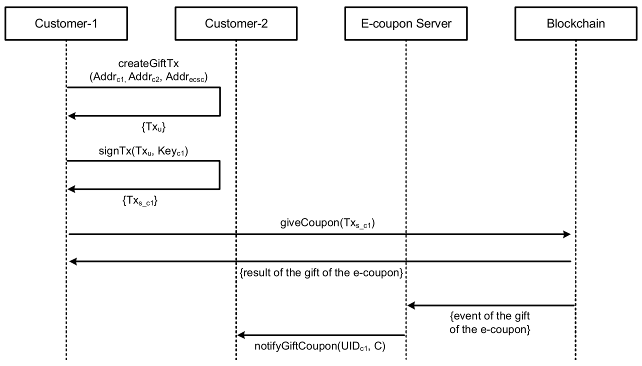

Customer-1
Customer-2
E-coupon Server
Blockchain
createGiftTx ($Addr_{c1}$, $Addr_{c2}$, $Addr_{ecsc}$)
{Tx_u}
signTx(Tx_u, Key_{c1})
{Tx_{s_c1}}
giveCoupon(Tx_{s_c1})
{result of the gift of the e-coupon}
{event of the gift of the e-coupon}
notifyGiftCoupon(UID_{c1}, C)

FIGURE 6. **Gifting an e-coupon.**

1) REGISTRATION OF A NEW MEMBER

Figure 3 shows the process of registering a new member (i.e., an e-coupon provider or a customer). To perform a business logic operation through a smart contract, each member needs to create a wallet via `createWallet()`. The wallet stores pairs of public and private keys and is configured to interact with the blockchain. For example, an e-coupon provider uses the wallet when deploying e-coupon smart contracts, paying for a product or service, or confirming usage of an e-coupon. Also, a customer uses the wallet when downloading, giving, or using an e-coupon.

To create a pair of private and public key in the wallet, we use a one-way encryption algorithm (i.e., public key infrastructure (PKI) [23]) that generates a pair of private and public keys such as ($\mathtt{Key}_c$, $\mathtt{Pub}_c$, $\mathtt{Addr}_c$) or ($\mathtt{Key}_{ecp}$, $\mathtt{Pub}_{ecp}$, $\mathtt{Addr}_{ecp}$) as shown in Figure 3. $\mathtt{Key}_c$ and $\mathtt{Key}_{ecp}$ are random numbers and the key size is 256 bits (32 bytes). Generated $\mathtt{Key}_c$ and $\mathtt{Key}_{ecp}$ are encrypted using the password transmitted by a member for high-level security. $\mathtt{Pub}_c$ and $\mathtt{Pub}_{ecp}$ are derived from $\mathtt{Key}_c$ and $\mathtt{Key}_{ecp}$ via elliptic curve cryptography (ECC) [24] and the key size is 512 bits (64 bytes). $\mathtt{Pub}_c$ and $\mathtt{Pub}_{ecp}$ are again hashed into a SHA-3 (i.e., Keccak-256), resulting in 256 bits (32 bytes), and the last 20 bytes are used as the wallet address (i.e., $\mathtt{Addr}_c$ or $\mathtt{Addr}_{ecp}$), which is target address of a transaction. After creating the wallet, the member requests to register the wallet address on the e-coupon server, and the e-coupon server stores the wallet address with `UID` via `registerMember()`. Finally, the e-coupon server transfers the result of the registration.

2) ISSUING AN E-COUPON SMART CONTRACT

Figure 4 shows the procedure of issuing an e-coupon smart contract. When an e-coupon provider issues an e-coupon (e.g., tickets, gift certificates, discount coupons, etc.), the e-coupon provider sets the e-coupon information (`C`), including the price, the number of e-coupons, start date, expiration date, etc. The price of an e-coupon is what a customer has to pay when downloading the e-coupon. When the price is set to 0, the e-coupon is classified as a free coupon. The number of e-coupons indicates the total count of remaining e-coupons. This means that customers cannot download the e-coupon anymore if the number is 0. The start date shows the starting date when a customer can download the e-coupon. The expiration date shows the duration of an e-coupon. When an e-coupon expires, all related operations will be disabled (such as downloading, giving, and using the e-coupon).

After setting the e-coupon information (`C`), the e-coupon provider requests the e-coupon server to create $\mathtt{Tx}_u$, which is a transaction to generate the e-coupon smart contract via `createContractTx()`. The e-coupon server creates $\mathtt{Tx}_u$ with `C`, `UID`, $\mathtt{Addr}_{ecp}$ and returns $\mathtt{Tx}_u$ to the e-coupon provider. The e-coupon provider checks $\mathtt{Tx}_u$ and signs $\mathtt{Tx}_u$ with $\mathtt{Key}_{ecp}$ via `signTx()`. And then, the e-coupon provider transmits $\mathtt{Tx}_{s\_ecp}$ to the blockchain via `deploySmartContract()`. If the transaction of the e-coupon smart contract ($\mathtt{Tx}_{s\_ecp}$) is valid, the transaction is processed to issue the contract and the e-coupon information is stored in the blockchain. After then, the e-coupon provider requests to register the smart contract address ($\mathtt{Addr}_{ecsc}$) on the e-coupon server via `registerContract()`. At this time, to synchronize the e-coupon server and the blockchain, the e-coupon server gets the e-coupon information (`C`) of the blockchain and stores it in its database via `getCouponInfo()`. Furthermore, the e-coupon server transfers the result of the smart contract registration for the e-coupon smart contract to the e-coupon provider. Finally, the e-coupon information can be provided to customers so that they obtain the e-coupon list to download e-coupons.

3) DOWNLOADING AN E-COUPON

As shown in Figure 5, the first step of downloading an e-coupon, customers receive a list of e-coupon information (`C`) and e-coupon smart contract addresses ($\mathtt{Addr}_{ecsc}$) from the e-coupon server via `getCouponList()`. The customer can use a filter to receive all e-coupons or specific e-coupons according to the e-coupon provider's `UID`. Next, the customer creates an e-coupon download transaction ($\mathtt{Tx}_u$) of the desired e-coupon with the e-coupon smart

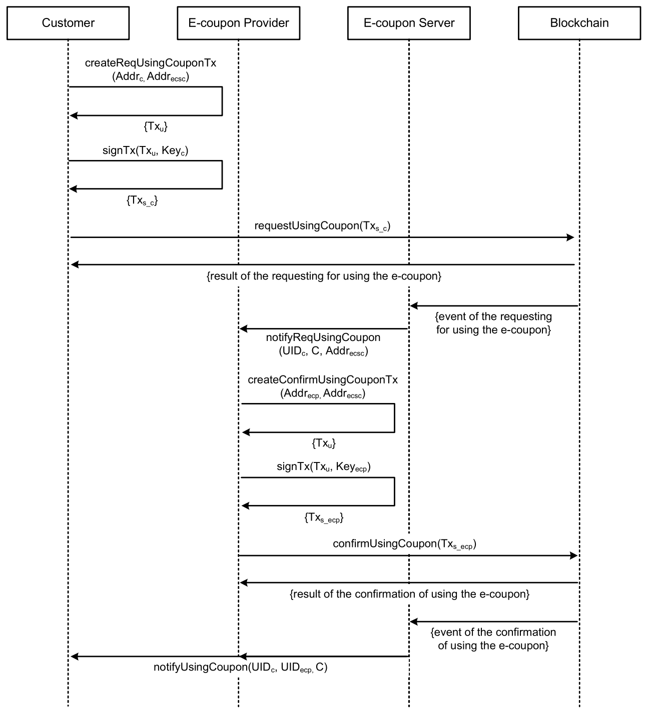

Customer
E-coupon Provider
E-coupon Server
Blockchain
createReqUsingCouponTx ($Addr_c$, $Addr_{ecsc}$)
{Tx_u}
signTx(Tx_u, Key_c)
{Tx_{s_c}}
requestUsingCoupon(Tx_{s_c})
{result of the requesting for using the e-coupon}
{event of the requesting for using the e-coupon}
notifyReqUsingCoupon (UID_c, C, $Addr_{ecsc}$)
createConfirmUsingCouponTx ($Addr_{ecp}$, $Addr_{ecsc}$)
{Tx_u}
signTx(Tx_u, Key_{ecp})
{Tx_{s_ecp}}
confirmUsingCoupon(Tx_{s_ecp})
{result of the confirmation of using the e-coupon}
{event of the confirmation of using the e-coupon}
notifyUsingCoupon(UID_c, UID_{ecp}, C)

FIGURE 7. **Using an e-coupon.**

contract address ($\mathtt{Addr}_{ecsc}$) and the wallet address ($\mathtt{Addr}_c$) via `createDownloadTx()`.

After then, the customer signs the transaction with own private key ($\mathtt{Key}_c$) via `signTx()`. The transaction (Tx_{s_c}) is propagated to the blockchain via `downloadCoupon()` and the corresponding e-coupon smart contract verifies the validity of the downloadable e-coupon by checking different parameters, such as the validity of the transaction signature, the validity period, and the quantity of the coupon. If the transaction is valid (i.e., the e-coupon is downloaded), the e-coupon smart contract in the blockchain updates the state with the customer having downloaded the e-coupon and propagates the e-coupon download transaction to the blockchain network. Otherwise, the blockchain returns a failed result. When a block containing the transaction is generated, the e-coupon server obtains and stores an event of downloading e-coupon from the blockchain to its database, to synchronize between the e-coupon server and blockchain.

4) GIFTING AN E-COUPON

Figure 6 shows the process that a customer gifts an e-coupon to another customer. First, a customer-1 creates a transaction to give an e-coupon to customer-2 via `createGiftTx()` with three parameters such as customer-1's wallet address ($\mathtt{Addr}_{c1}$), customer-2's wallet address ($\mathtt{Addr}_{c2}$), and the e-coupon smart contract address ($\mathtt{Addr}_{ecsc}$) associated with the e-coupon. After creating the transaction, the customer-1 signs the transaction ($\mathtt{Tx}_u$) with the own private key ($\mathtt{Key}_{c1}$) via `signTx()`, and passes the signed transaction (Tx_{s_c1}) to the blockchain via `giveCoupon()`. The e-coupon smart contract, according to the address ($\mathtt{Addr}_{ecsc}$), executes a gift operation to validate the transaction (Tx_{s_c1}) whether the customer-1 has enough coupons, the e-coupon has not expired, etc. After then, the blockchain returns the execution result such as success or failure of the transaction. Also, the e-coupon server obtains and stores the event of e-coupon for gift, and transfers the notification (an event with which the e-coupon for gift is now available) to customer-2 via `notifyGiftCoupon()`.

5) USING AN E-COUPON

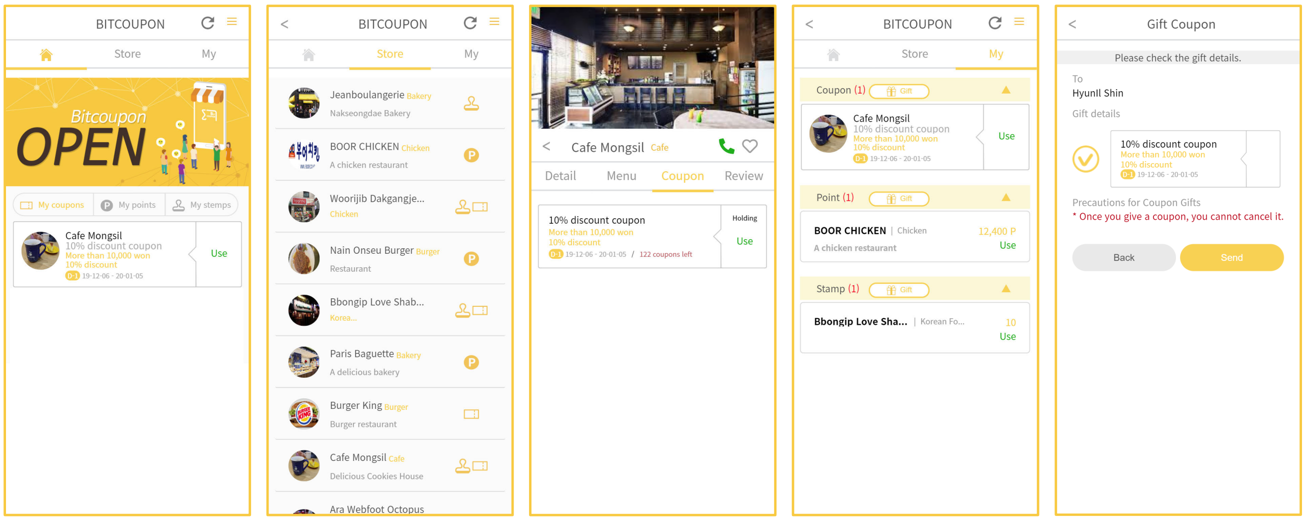
Figure 7 shows the usage of an e-coupon. As shown in the figure, two types of transactions are required when using an e-coupon. One transaction is performed when the e-coupon is used by a customer. Another transaction is to confirm the use by the e-coupon provider. This is to prevent the e-coupon provider from using the e-coupon without the customer's permission. This guarantees that the provider can use the e-coupon only based on the user's request.

To use an e-coupon (i.e., a discount or reserve e-coupon), a customer creates a transaction ($\mathtt{Tx}_u$) with the customer wallet address ($\mathtt{Addr}_c$) and the e-coupon contract address ($\mathtt{Addr}_{ecsc}$) via `createReqUsingCouponTx()`. Then, the customer signs the transaction with their private key ($\mathtt{Key}_c$) via `signTx()` and sends the signed transaction ($\mathtt{Tx}_{s\_c}$) to the blockchain via `requestUsingCoupon()`. On the blockchain, the e-coupon smart contract performs the transaction. When the transaction is valid, the smart contract modifies the state of the e-coupon with the transaction. Then, the blockchain returns the result of the requesting for using the e-coupon to the customer. After then, the e-coupon server gets the event of the requesting for using the e-coupon and synchronizes the e-coupon state. The e-coupon server notifies the e-coupon provider about the request for the use of the e-coupon via `notifyReqUsingCoupon()`.

Subsequently, to get an approval to use the e-coupon, the e-coupon provider generates another transaction ($\mathtt{Tx}_U$) to confirm use of the e-coupon via `createConfirmUsingCouponTx()` and signs the transaction with the private key of the e-coupon provider ($\mathtt{Key}_{ecp}$) via `signTx()`. After then, the e-coupon provider sends the signed transaction ($\mathtt{Tx}_{s\_ecp}$) to the blockchain via `confirmUsingCoupon()`. When the blockchain receives the transaction ($\mathtt{Tx}_{s\_c}$), the e-coupon smart contract carries out the transaction and returns the result regarding the confirmation of using the e-coupon. Once the use of the e-coupon is approved, the e-coupon server takes the event of confirming usage of the e-coupon and notifies the customer and the e-coupon provider via `notifyUsingCoupon()`.

E. DEMONSTRATION OF THE PROPOSED E-COUPON SERVICE

Based on the proposed smart contract mechanisms, we develop a proof of concept (PoC) service, as shown in Figure 8. As shown in the figure, there are screens of the main, list of stores, detail of a store, list of e-coupon, and e-coupon for gift. Figures 8(a) shows the home screen, which



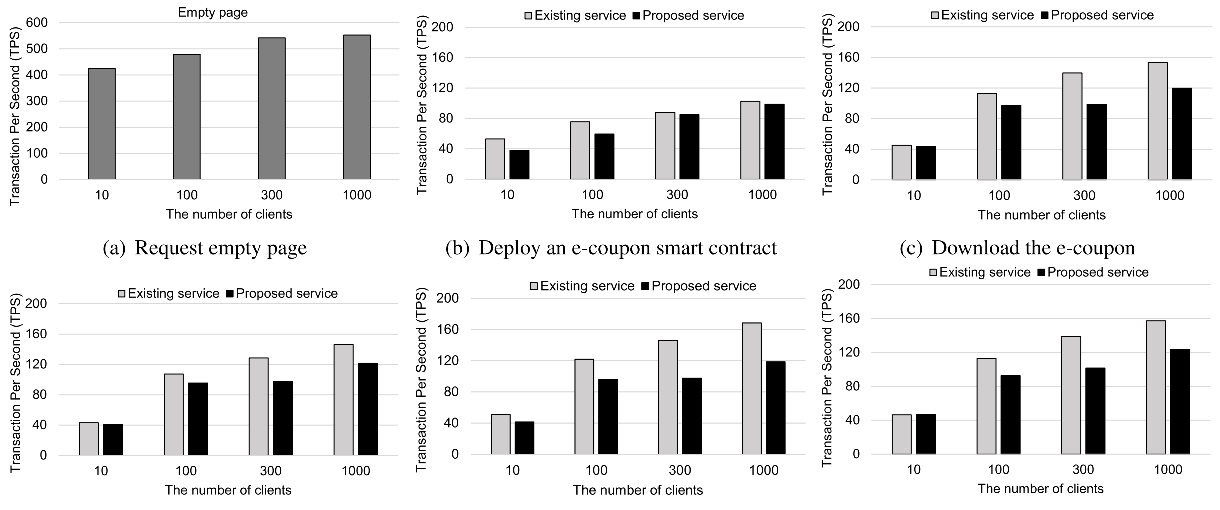
Please check the gift details.

Precautions for Coupon Gifts

* Once you give a coupon, you cannot cancel it.

(a) Main screen (b) Store list screen (c) Store detail screen (d) My list screen (e) Gift screen

FIGURE 8. **Proposed e-coupon service.**



(a) Request empty page (b) Deploy an e-coupon smart contract (c) Download the e-coupon

(d) Give the e-coupon (e) Request using the e-coupon (f) Confirm using the e-coupon

FIGURE 9. **Performance results.**

includes the e-coupon information. The menu of *My coupons* shows the discount e-coupon. The menus of *My points* and *My stamps* show the reserve e-coupon. To download a discount e-coupon, first, the customer gets a list of stores (i.e., e-coupon provider) as shown in Figure 8(b). After then, the customer can download the e-coupon from the store detail screen as shown in the Figure 8(c).

Figure 8(d) shows the e-coupon information owned by a customer. The customer can also give the e-coupon to other customers using a gift button. Figure 8(e) shows a gift screen, where a customer confirms giving an e-coupon to another customer. Once the customer gives an e-coupon to another customer, the customer cannot cancel it. As well as these features, our service has additional functionalities which we omit in Figure 8. For example, there are customer registration, request of using e-coupon, issuing an e-coupon, issuing points or stamps, and confirming the use of e-coupon. Also, by using smart contracts, users can maximize efficiency by allowing users to exchange e-coupons for coupons that they want.

IV. EVALUATION

A. EXPERIMENTAL SETUP

We perform all the experiments on private blockchain which is built with five nodes. Each node includes two Intel Xeon E5-2683 processors (total 32 cores), 64 GiB DRAM, and runs Ubuntu 16.04.5 LTS distribution with Linux kernel 4.4.0. In our service, we construct an e-coupon server using the web3.js and an Express framework. Also, we use Quorum, an Ethereum-based distributed ledger protocol, which provides new consensus mechanisms for private blockchain [7]. In the consensus mechanisms, we use Istanbul BFT (Byzantine Fault Tolerance) consensus algorithm. To build this BFT-based blockchain, we need more than 3f+1 nodes are required (f is the number of faulty nodes). The minimum number of nodes is five when the number of fault nodes is one. Thus, we build the BFT-based blockchain with five nodes. In addition, in the case of Ethereum-based public blockchain, the gas is used to prevent spam on the Ethereum networks. Meanwhile, in our scheme, we use an Ethereum-based private blockchain (i.e., Quorum). In the case of the Ethereum-based private blockchain, the gas is not required since only authorized nodes can access the blockchain. Thus, we do not consider gas limit and gas price.

We empirically evaluate our service by using a synthetic benchmark. The smart contract scenario of the synthetic benchmark consists of four operations: deploying an e-coupon smart contract, downloading, giving, and using the e-coupon. To compare the performance differences between blockchain e-coupon service and blockchain-free service, we evaluate the proposed service using the blockchain and an existing service without using the blockchain. We measure the performance with Jmeter [25] which is used as a load testing tool for analyzing and measuring the performance of a variety of web application services.

B. PERFORMANCE RESULTS

Figure 9 shows the performance results for each operation in the existing and proposed services. For experimental parameters, we set the number of clients as 10, 100, 300, and 1000 at each experiment. Each client generates a transaction, and the number of client is same as the number of transactions. And, the performance metric is transaction per second (TPS). Note that Figure 9(a) shows a baseline performance through requesting empty page to e-coupon server. It shows the performance from 425 TPS to 532 TPS. However, other experimental results show the performance of 38 TPS to 168 TPS in both existing and proposed schemes as shown in Figure 9(b), 9(c), 9(d), 9(e), and 9(f). This is because each operation has an overhead to store data into a database or execute various business logic, etc.

Overall, Figure 9 shows that the performance increases when a large number of requests is issued. This is due to the increase of parallelism as the number of threads processed in the server increases. As shown in Figure 9(b), in the case of the deploy operation, the performance of the proposed service is reduced by up to 28%, 21%, 3%, and 3% compared with the existing service (without blockchain) when the number of clients is 10, 100, 300, and 1000, respectively. This is because the proposed service uses the e-coupon smart contract in the blockchain to improve security. Especially, the result of deploying an e-coupon smart contract as depicted in Figure 9(b) shows a few lower throughput than other operations such as downloading, giving, requesting, and Confirming of an e-coupon. This is because the deploy operation stores more data and requires more steps such as validation of all the e-coupon information. Except for the result of the deploy operation, other operations show similar performance.

In the case of downloading, giving, requesting, and confirming e-coupon, as shown in Figures 9(b), 9(c), 9(d), 9(e), and 9(f), the proposed service shows the performance degradation by up to 18%, 21%, 33%, and 33% compared with an existing service when the number of clients is 10, 100, 300 and, 1000, respectively. The results show that this performance degradation depends on the use of blockchain. Usually, there is a trade-off between performance and security [26]. We sacrifice the performance of the e-coupon service.

Instead, we focus on improving the security level by guaranteeing the integrity of e-coupon information. For example, the existing e-coupon service uses a database system. In this system, since an administrator can easily obtain the authority, the administrator rather easily modifies the data maliciously. Meanwhile, in our provided service which uses a blockchain system, the administrator cannot easily obtain the authority since the authority should be obtained by the consensus of all users. Thus, it is hard to take over the authority, and we can prevent the malicious modification. This means that our proposed scheme increases the security level. We explain in detail in sub-section IV-C. Also, note that many studies [27], [28] are underway to improve the performance of the blockchain. Therefore, performance issues with blockchain will be mitigated in the future and we also leave the improvement for the performance of blockchain as future work.

C. DISCUSSION OF SECURITY

Blockchain is an append-only database, so it cryptographically links each block added to the blockchain and does not provide modify and delete operations. For example, to change a block's contents, the hash value of all subsequent blocks from the block must be modified, which also has to be agreed upon by other nodes via a consensus algorithm. Therefore, a malicious attacker must hack into multiple nodes in the blockchain network. This means that if the attacker tries to modify or delete data, the attacker must have overwhelming power compared with the others. Through these techniques, the blockchain provides a high level of integrity and security for data (i.e., e-coupon information). In addition, the smart contract on the blockchain can ensure the integrity of the business logic (i.e., operations for e-coupon) because it also is stored in the blockchain. Consequently, our proposed e-coupon service exploiting blockchain and smart contract can provide a more secure while maintaining the performance with a minor overhead.

In an e-coupon service, there are security requirements which are (1) non-repudiation and (2) unique usage of e-coupon [29].

1) NON-REPUDIATION

Non-repudiation in an e-coupon service is the assurance that users (i.e., e-coupon provider and customer) cannot deny the validity of associated transactions (i.e., issue, use, and give an e-coupon). To provide the non-repudiation, we use digital signature and blockchain. To perform an e-coupon operation, an user generates a transaction and signs the transaction. Smart contract checks if the signed transaction is valid or not. If the signed transaction is valid, the signed transaction is stored in the blockchain. With the signed transaction, the user can identify who issues, uses, or gives an e-coupon. In addition, the blockchain can prevent modifying the stored transaction by using a consensus algorithm with multiple nodes. Therefore, when an attacker tries to falsify a transaction, the attacker must have overwhelming computing power compared with the other nodes, which is unrealistic. Consequently, we can identify who performs a transaction with the signature and preserve the transaction invariant with blockchain. Thus, it allows that the user signing the transaction cannot deny the performed transaction.

2) UNIQUE USAGE

An unique usage of e-coupon means that a customer cannot use the already used e-coupon again. To guarantee the unique usage, we devise an e-coupon smart contract based on the blockchain. The smart contract is performed according to the e-coupon state information (e.g., whether to use coupons, the coupon price, the number of e-coupons, start date, expiration date, etc.). For example, when a customer uses an e-coupon, the e-coupon smart contract allows the use of the e-coupon and changes the e-coupon state as ''used''. After then, when the customer tries to use the e-coupon again, the e-coupon smart contract rejects the use of the e-coupon since the e-coupon state is already changed ''used'' by the smart contract. Thus, we can provide the unique usage of an e-coupon by exploiting the e-coupon smart contract.

V. RELATED WORK

There are previous studies [2], [12], [13], [15] for providing secure e-coupon. Blundo *et al.* [2] propose new e-coupon models and e-coupon protocols using message authentication code (MAC) for e-coupon security. Agarwal *et al.* [12] propose a solution based on a third-party centralized coupon mint, which checks for double-spending. Hsueh *et al.* [13] sign the e-coupon with digital signatures (i.e., PKI) and use hash functions to check the consistency of the information and verify all digital signatures of the e-coupon. Chang *et al.* [15] use one-way hash function and MAC, allowing e-coupon providers to prevent e-coupon from being double-redeemed by customers without any additional computation cost on mobile devices.

With these techniques [2], [12], [13], [15], a user can detect whether an e-coupon is modified or not by a malicious attacker. Therefore, they prevent the forgery and falsification of an e-coupon and effectively manage e-coupon issue and use of e-coupons. However, these approaches are not suitable when e-coupon information can be modified by a malicious attacker in the e-coupon server database. In addition, these approaches cannot prevent the malicious behavior of an administrator. Our study is in inline with these works [2], [12], [13], [15] in terms of enhancing security of e-coupons. In contrast, we focus on improving the security of e-coupon service stored in its database as well as preventing forgery of e-coupon itself.

Hsueh *et al.* [5] provide a hash chain which is combined with the blockchain technology to verify the forgery of e-coupons. They guarantee the integrity of e-coupon information by using blockchain technology. Our study is in inline with the work [5] in terms of using blockchain technology for integrity of e-coupon. In contrast, we exploit a smart contract to provide the integrity of the e-coupon business logic such as downloading, using, and gifting an e-coupon.

Podda *et al.* [29] analyze and compare several blockchain-based coupon systems. It also proposes a general schema of digital coupons and points out the basic properties that a coupon system should guarantee. Hsu *et al.* [30] analyze to prove that the security requirements of the e-voucher system and explore how to apply blockchain technology and cryptography to build a secure e-voucher system. And, They propose a feasible application model that integrates blockchain technology in the context of vouchers to support the field of the campus welfare meal voucher system. Our study is in line with this approaches [29], [30] in terms of providing the e-coupon security features (non-repudiation, unique usage, decentralized verification, etc.) by using a blockchain system and start contract. In contrast, we focus on investigating the performance and cost of development by using e-coupon smart contract template. In addition, we consider a general-purpose e-coupon system rather than a specific use case (i.e., campus welfare meal voucher system) with e-coupon smart contract template.

VI. CONCLUSION

We have investigated e-coupon services that store e-coupon information on a centralized server. We found that the e-coupon information stored in the server can be manipulated by a malicious attacker or administrator. To handle this issue, we present a new e-coupon service that improves security by exploiting e-coupon smart contracts in a blockchain system. We have implemented the proposed service on the Quorum blockchain and evaluated the service using a synthetic benchmark. According to our experimental results, the proposed service prevents the manipulation of e-coupon information with higher security and minor performance overhead. In the future, we will focus on improving blockchain performance.

REFERENCES

[1] (2019). *Wikipedia: E-coupon*. [Online]. Available: https://en.wikipedia.org/wiki/E-coupon

[2] C. Blundo, S. Cimato, and A. De Bonis, ''Secure E-coupons,'' *Electron. Commerce Res.*, vol. 5, no. 1, pp. 117–139, Jan. 2005.

[3] (2016). *World Mobile Coupons Market to Grow at 73.1% CAGR to 2020*. [Online]. Available: https://www.prnewswire.com/news-releases/world-mobile-coupons-market-to% -grow-at-7314-cagr-to-2020-603320306.html
[4] (2017). *Coupon Fraud is Crime, Even if it Feels Harmless: Coupon Counselor*. [Online]. Available: https://goo.gl/2emab1.
[5] S.-C. Hsueh and J.-H. Zeng, "Mobile coupons using blockchain technology," in *Proc. Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.* Springer, 2018, pp. 249–255.
[6] A. Knight and N. Dai, "Objects and the web," *IEEE Softw.*, vol. 19, no. 2, pp. 51–59, Mar. 2002.
[7] (2018). *Quorum*. [Online]. Available: https://github.com/jpmorganchase/quorum
[8] (2017). *Coupon Statistics: The Ultimate Collection*. [Online]. Available: https://blog.accessdevelopment.com/ultimate-collection-coupon-statistic%s
[9] (2017). emphDigital Coupon Marketing—Statistics and Trends. [Online]. Available: https://www.invespcro.com/blog/digital-coupon-marketing
[10] (2019). *Digital Coupons Continue to be the Fastest Growing Method of Redemption due to Shoppers' Increased Demand for Convenience*. [Online]. Available: https://www.globenewswire.com/news-release/2019/02/13/1724510/0/en/Digi%tal-Coupons-Continue-to-be-the-Fastest-Growing-Method-of-Redemption-Due-to-Sho%ppers-Increased-Demand-for-Convenience.html
[11] (2017). *The Coupon Insider: Digital vs. Paper Coupons*. [Online]. Available: https://livingonthecheap.com/coupon-insider-digital-paper-coupons//
[12] R. G.-P. M.-V. Agarwal and N. Modani, "An architecture for secure generation and verification of electronic coupons," in *Proc. USENIX Annu. Tech. Conf.*, Boston, MA, USA, Jun. 2001, p. 51.
[13] S.-C. Hsueh and J.-M. Chen, "Sharing secure m-coupons for peer-generated targeting via eWOM communications," *Electron. Commerce Res. Appl.*, vol. 9, no. 4, pp. 283–293, Jul. 2010.
[14] R. Rivest, "The MD5 message-digest algorithm," Tech. Rep., 1992.
[15] C.-C. Chang, C.-C. Wu, and I.-C. Lin, "A secure e-coupon system for mobile users," *Int. J. Comput. Sci. Netw. Secur.*, vol. 6, no. 1, p. 273, 2006.
[16] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Appl. Innov.*, vol. 2, nos. 6–10, p. 71, 2016.
[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008.
[18] M. Szydlo, "Merkle tree traversal in log space and time," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Springer, 2004, pp. 541–554.
[19] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.
[20] N. Szabo, "Smart contracts: Building blocks for digital markets," Tech. Rep., 2018.
[21] V. Buterin, "A next-generation smart contract and decentralized application platform," Tech. Rep., 2014.
[22] V. Buterin, "A next-generation smart contract and decentralized application platform," *White Paper*, vol. 3, p. 37, Jan. 2014.
[23] U. Maurer, "Modelling a public-key infrastructure," in *Proc. Eur. Symp. Res. Comput. Secur.* Springer, 1996, pp. 325–350.
[24] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Springer, 2006.
[25] (2019). *Apache JMeter—Apache JMeterT*. [Online]. Available: https://jmeter.apache.org/
[26] K. Wolter and P. Reinecke, "Performance and security tradeoff," in *Proc. Int. School Formal Methods Design Comput., Commun. Softw. Syst.* Springer, 2010, pp. 135–167.
[27] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data*, Jun. 2019, pp. 123–140.
[28] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 95–112.
[29] A. S. Podda and L. Pompianu, "An overview of blockchain-based systems and smart contracts for digital coupons," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops*, Jun. 2020, pp. 770–778.
[30] C.-S. Hsu, S.-F. Tu, and Z.-J. Huang, "Design of an E-voucher system for supporting social welfare using blockchain technology," *Sustainability*, vol. 12, no. 8, p. 3362, Apr. 2020.

JONGBEEN HAN received the B.S. degree in computer engineering from Hansung University, in 2015, and the M.S. degree from the Department of Computer Science and Engineering, Seoul National University, in 2019, where he is currently pursuing the Ph.D. degree in computer science and engineering. His research interests include blockchain, operating, and distributed systems.

YONGSEOK SON received the B.S. degree in information and computer engineering from Ajou University, in 2010, and the M.S. and Ph.D. degrees from the Department of Intelligent Convergence Systems and Electronic Engineering and Computer Science, Seoul National University, in 2012 and 2018, respectively. He was a Postdoctoral Research Associate in electrical and computer engineering at the University of Illinois at Urbana-Champaign. Currently, he is an Assistant Professor with the School of Computer Science and Engineering, Chung-Ang University. His research interests include operating, distributed, and database systems.

HYEONSANG EOM received the B.S. degree in computer science and statistics from Seoul National University (SNU), Seoul, South Korea, in 1992, and the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, MD, USA, in 1996 and 2003, respectively. He was an Intern with the Data Engineering Group, Sun Microsystems, CA, USA, in 1997, and a Senior Engineer with the Telecommunication Research and Development Center, Samsung Electronics, South Korea, from 2003 to 2004. He is currently a Professor with the Department of Computer Science and Engineering, SNU, where he has been a Faculty Member, since 2005. His research interests include high performance storage systems, operating systems, distributed systems, cloud computing, energy efficient systems, fault-tolerant systems, security, and information dynamics.