First Smart Contract Allowing Cryptoasset Recovery

Beomjoong Kim[1], Hyoung Joong Kim[1*], and Junghee Lee[1]
[1] School of Cybersecurity, Korea University
Seoul 02841, South Korea
[e-mail: {physika73, khj-, j_lee}@korea.ac.kr]
[*]Corresponding author: Hyoung Joong Kim

Received October 23, 2021; revised January 11, 2022; accepted February 21, 2022; published March 31, 2022

Abstract

Cryptoassets such as Bitcoin and Ethereum are widely traded around the world. Cryptocurrencies are also transferred between investors. Cryptocurrency has become a new and attractive means of remittance. Thus, blockchain-based smart contracts also attract attention when central banks design digital currencies. However, it has been discovered that a significant amount of cryptoassets on blockchain are lost or stranded for a variety of reasons, including the loss of the private key or the owner's death. To address this issue, we propose a method for recoverable transactions that would replace the traditional transaction by allowing cryptoassets to be sent to a backup account address after a deadline has passed. We provide the computational workload required for our method by analyzing the prototype. The method proposed in this paper can be considered as a good model for digital currency design, including central bank digital currency (CBDC).

Keywords: Blockchain, central bank digital currency (CBDC), cryptoasset, distributed ledger technology (DLT), smart contract.

This research was supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number : HI19C0785).

http://doi.org/10.3837/tiis.2022.03.006 **ISSN : 1976-7277**

1. Introduction

Assume that Alice sends *X* bitcoins to an address that does not exist. Alice may regret this action and want to recover the bitcoin, but there is no way to get it back until she finds the private key of the non-existent address, which is known to be computationally infeasible. Assume that Bob receives *Y* bitcoins, but he cannot remember his private key. Similarly, there is no way to get the bitcoin back unless he finds his private key, which is also computationally infeasible. There are other scenarios in which people would want to get their cryptocurrency back. Surprisingly, there has been no way to undo such transactions until now. There is a method that allows transactions to be suspended [1] in order to prevent such accidents, but it only halts the movement of cryptocurrency and does not guarantee the returns after the incident. Generally, people would not even try to get their coins back because they thought it would harm the integrity or immutability of the blockchain. In this paper, a method for recovering cryptoassets without compromising immutability is proposed, to our knowledge, for the first time.

Of course, returning all cryptoassets in any case would damage the immutability of the blockchain. Cryptoassets must be returned only in special cases, and their return must not compromise immutability. Special cases should be limited to cases where return is absolutely necessary in real life. This cryptoasset recovery functionality differentiates digital currency from analog currency. Moreover, this feature will be an advantage for digital currency.

With analog currency, it is impossible to transfer money to an account that does not exist. Thus, there is no need to be able to get the money back. On the other hand, it is possible that Alice tries to send money to Bob, but she accidentally sends money to Charles. At this point, Alice must either beg Charles for her money back or sue Charles. Laws are needed to recover erroneous remittances [2]. In online payment services, refund options are provided for such erroneous transactions, for example, by PayPal [3] and Apple Pay [4]. In the event of a cryptoasset holder's death, there are ways [5] for the bereaved family to claim the cryptoasset. In any case, recovery is not easy and takes a long time. With digital currency, the smart contract solves the problem automatically for certain cases using the method proposed in this paper, so there is no need for Alice to beg or sue Charles. This feature will be another advantage of digital currency.

The core value of the blockchain is its transparency and immutability. All transactions and smart contracts are recorded on the blockchain and can be viewed by anyone. Thus, the transparency of the blockchain is maintained. Because it is transparent, it is easy to check whether the blockchain ledger is immutable; to guarantee immutability, the blockchain ledger must be transparent. Immutability is referred to as the ability of a blockchain ledger to remain unaltered and unchanged.

However, the immutability of the blockchain has sometimes been compromised. The case of The DAO (Decentralized Autonomous Organization) in 2016 was an example. A serious bug in The DAO's smart contract called the recurrence attack was exploited, and an enormous amount of cryptocurrency was abnormally withdrawn. The DAO was run on the Ethereum blockchain platform. After the exploitation of the recurrence attack, participants in this organization were divided into two groups. One group refused to edit the records on the Ethereum blockchain, subscribing to the idea “Code is Law.” Another group argued that for the protection of investors, the blockchain should be returned to the state as it was in just before the cryptocurrency was abnormally withdrawn. The conflict of opinion between the two groups was not resolved, and in the end, the Ethereum blockchain was split into the Ethereum Classic blockchain, which retained its immutability, and the original Ethereum blockchain that gave

up its immutability [6]. This rarely happens because it is considered a violation of the fundamental principle of blockchain technology [7].

The spirit of the blockchain, according to which all policies are maintained based on the majority consent of the participants, was undermined in Ethereum at this time. A blockchain consensus algorithm was needed to allow participants to reach consensus [8-12]. However, immutability can be a drawback, and it is still a controversial issue [13-14]. An approach is needed to resolve the controversy while maintaining integrity, which is the core value of the blockchain. This paper presents a meaningful approach from that point of view.

Even after the DAO's attack, many of the smart contracts deployed on blockchain platforms were found to have vulnerabilities [15]. Despite the fact that the actual exploitations of vulnerabilities were limited [16], the need for modifications to smart contract code arose. Some solutions have been developed to meet these needs, such as Open Zeppelin's upgradeable smart contract [17]. Still, such solutions are inapplicable in the case of erroneous transactions between individuals, such as sending cryptoassets to the incorrect recipient.

There are no parties in the blockchain world who can force a cryptoasset to move from one account to another [18]. As a result, intervention by a third party in cryptoasset losses, as in the real world, is impossible. When it comes to cryptoasset losses on blockchain due to the loss of passwords for unlocking a wallet, there exist third-party recovery services to regain them, such as Crypto Asset Recovery [19]. This kind of service allows people to access their own wallet with a password guessed by luck based on a brute force search using the client's personal information and preferences. The method cannot be used for a case where the password is truly randomly chosen and sufficiently long, and where the client's information is not fully available. Furthermore, they can recover the lost password of the wallet only; they cannot recover the password of the cryptoasset itself. Moreover, because password recovery with the participation of a third party may be considered a privacy violation, these services should only be used as a last resort.

In this paper, inspired by our previous work [20], we propose a method for recoverable transactions that is applicable to cryptoassets that require recovery. The blockchain should not forego immutability as a trade-off to prevent users' unintended cryptoasset loss. With the use of our method, participants are given options for preventing such unintentional losses while not violating any of the blockchain's core principles, such as the immutability.

As many central banks around the world are considering the implementation of central bank digital currency (CBDC) for the modernization of the monetary landscape [21], interest in distributed ledger technology (DLT) is growing [22-23]. The method presented in this paper can contribute to correcting remittance errors and protecting against unintended losses in CBDC as well as in other cryptocurrency designs. This contribution highlights the difference between digital currency and analog currency. Our method is easy to implement and illustrates common sense. Based on this proposal, it is hoped that interest in more convenient digital currency design will increase.

2. Scope of Recovery

In this section, we specify the scope of the cryptoassets that need to be recovered according to the focus of our research: unintended loss of cryptoassets on blockchain. The cases in each of the following three categories may result in significant losses to the sender, receiver, or society involved in the cryptoassets.

2.1 Unintentionally Burnt Cryptoassets

There exist transactions that bring about the unintentional burning of cryptoassets. The term "burnt cryptoassets" in the blockchain community refers to cryptoassets that have been transferred to an address that cannot be unlocked. In Ethereum, for example, people send transactions to the address "0x0" in order to burn their cryptoassets, which is known as the "Black Hole Address": "0x0000000000000000000000000000000000000000." Because unintentional burning can be considered a risk for users and the issuing party of the cryptoasset, most of the recent ERC-20 [24] tokens adopt the "*transfer*" or "*transferFrom*" function with a burn-safe method, as described in the algorithm in **Table 1**. The function "*REVERT*" in the algorithm is used to stop execution and revert state changes, like the "*OP_REVERT*" operation code on the Ethereum Virtual Machine (EVM), to avoid the unnecessary transaction fee.

However, the burn-safe method for ERC-20 tokens cannot prevent users from burning their tokens by sending them to addresses other than "0x0." Furthermore, as the method is not mandatory in token issuance, not all ERC-20 tokens are built with this technique.

Table 1. Algorithm for a burn-safe transfer method on ERC-20 tokens

```
    Input: To, balance_From, balance_To, Amount
1   if To = 0x0 then
2   |   REVERT
3   else
4   |   balance_From ← balance_From - Amount
5   |   balance_To ← balance_To + Amount
6   end
```

2.2 Erroneous Transactions

Erroneous transactions with incorrect inputs (e.g., faulty receiver address or wrong amount of cryptoasset sent) may result in regret. In such cases, refunds are difficult to obtain [25] on the blockchain because there are no mediators to handle erroneous transactions. To realize a refund, the recipient must confirm the windfall transaction in good faith and initiate a refund. As a result, if an erroneous transaction occurs with incorrect inputs, the sender has no way to get the money back except by the receiver's goodwill.

Erroneous transactions caused by incorrect inputs are not only made by accident. Some may carry out such transactions to launch attacks. The "Short Address Attack," [26] as shown in **Fig. 1**, is possible in this case, which shortens the receiver's address intentionally in order to manipulate the amount of cryptoassets sent.

2.3 Forgotten and Lost Private Keys

According to Chainalysis, as cited by the New York Times [27], approximately 20 percent of existing Bitcoin is lost or otherwise stranded. If a recipient of a transaction loses a private key, cryptoassets on the blockchain are considered lost. Because a private key is too long and complex to memorize (e.g., 32 random bytes for Bitcoin and Ethereum), people record it as a text file or write on paper, then store it in a safe place. People using a wallet application can set a relatively short password instead of long private keys to make it easier to access.

People who use a private key to move cryptoassets have no way to recover the cryptoassets if the key is lost [28]. Wallet application users can use as a backup seed keys that are initially provided by wallet applications. However, as such unlocking methods are dependent on the users' management skills, they are not always an appropriate way to prevent cryptoasset loss.

Furthermore, if the key owner dies, the bereaved family may have few options for recovering the cryptoassets. If this is the case, the cryptoassets may be left stranded, because third parties cannot access such cryptoassets in the blockchain world.

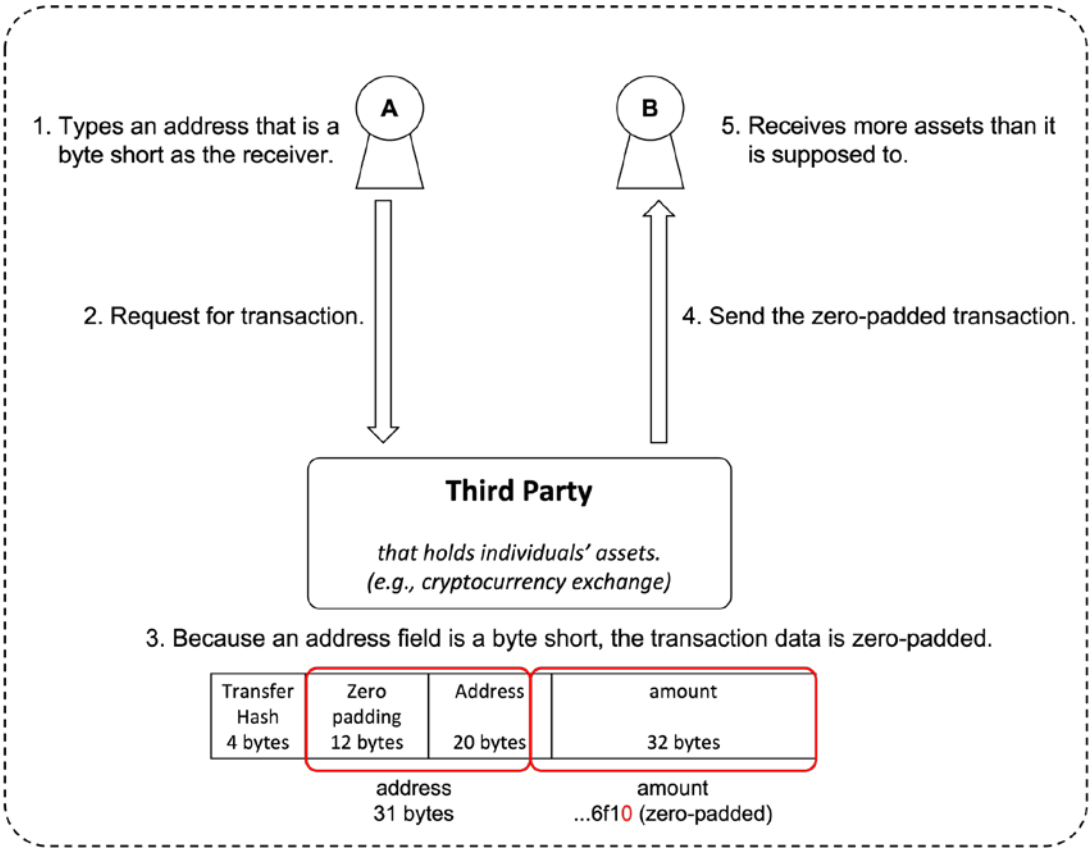

Fig. 1. A short address attack scenario by typing a byte short address on Ethereum.

3. Proposed Method

We propose a recoverable transaction method by introducing an intermediary smart contract called Kim smart contract (KSC) in order to avoid the cryptoasset losses described in Section 2. The transaction should be reversible to avoid the accidental burning of cryptoassets, because, for example, a method that examines whether the receiver's address is the "Black Hole Address" is not flawless. The method enables senders to retrieve the funds after some time has passed, while not violating the blockchain's immutability.

It is observed that there may be at least two reasons why a cryptoasset does not move [29] after a certain period of time. The first reason is that there is no immediate need to utilize the cryptoasset. For example, holders who expect the price of the cryptoasset to rise may be waiting to use it. This is the reason in most cases. The second reason may be that the private key is forgotten or lost. Thus, if the cryptoasset does not move even after a certain period of time, a smart contract such as KSC that allows the cryptoasset to be returned to a specific account can be used.

There are two key points presented in this paper. One is setting a specific time, that is, a deadline, and the other is the KSC, which allows cryptoassets to be withdrawn or moved without having the original receiver's private key. Originally, cryptoassets could not be moved from an account without the private key of the account. However, after the deadline has passed, the cryptoasset can be moved without the private key because it is in the KSC. Here, time plays the role of a tripwire. This is one of the key points of the proposed method. In short, time matters.

After a cryptoasset is placed in an account, the status of the cryptoasset is finalized. The cryptoasset in the account can be withdrawn or moved only when its private key becomes available. However, if the cryptoasset is in a smart contract, the cryptoasset can be withdrawn or moved according to the contract even when its private key is not available. That is why the KSC is used in this proposal. The cryptoasset can be stored temporarily in the KSC to enhance the security of the cryptoasset. Similarly, the smart contract matters. This is another key point of the proposed method.

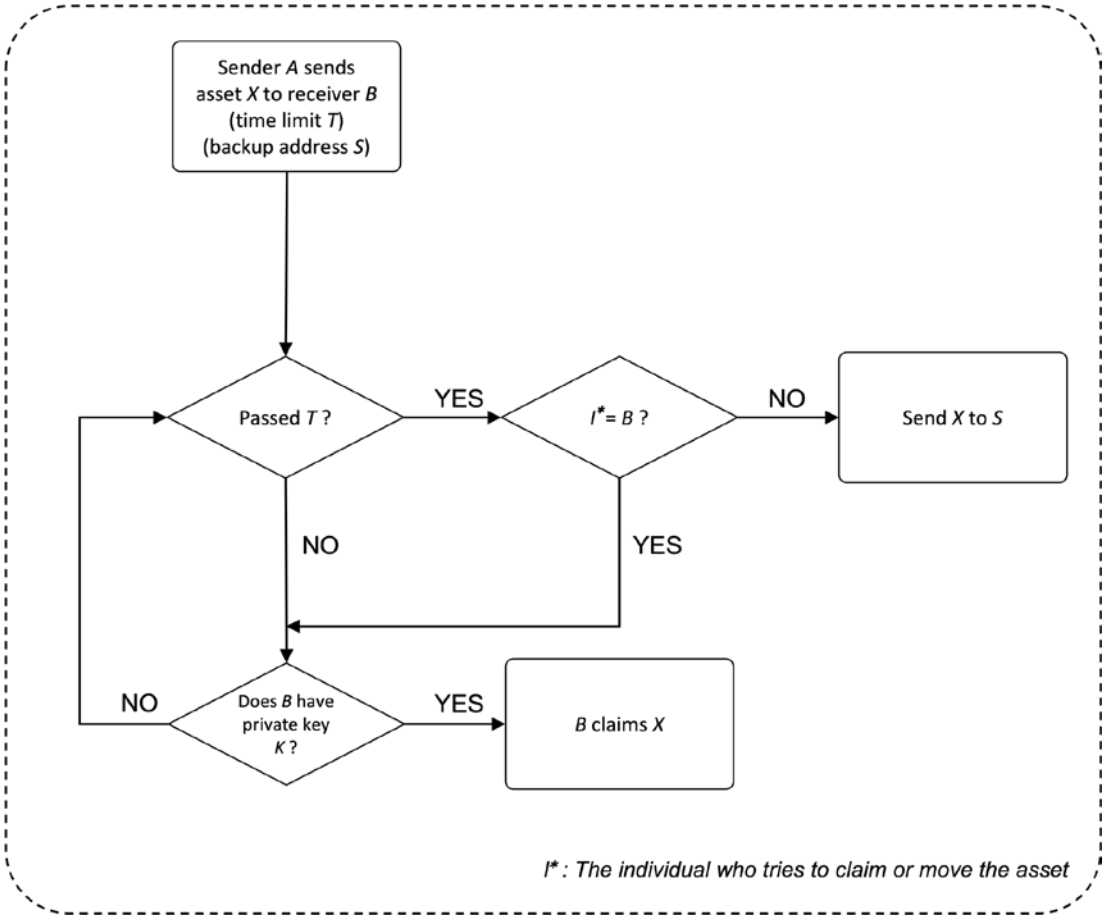

Fig. 2. Flowchart for workflow logic of the proposed method.

The transaction should make the cryptoasset reversible, while the transaction itself remains irreversible. Furthermore, because cryptoassets become refundable after a certain period of time, erroneous transaction problems can be resolved. Even if a cryptoasset is accidentally sent to an address that does not exist, or if a private key is forgotten or lost, there is now a way,

using the proposed method, to return the cryptoasset to the sender so that the rightful owners can recover it. Even after the deadline has passed, the owner of the private key can withdraw the cryptoasset unless someone has run the KSC. That someone includes the sender, the receiver, or any third party, but most often it is the sender or receiver.

The use of a backup account address can also enhance the safety of transactions. The backup account address, agreed upon in advance between the sender and receiver, is the address of an account that **holds** the cryptoasset (i.e., receives the cryptoasset) after a certain period of time. In this paper, the suggested backup account address is the sender's account address.

The reason that the sender's account address is suggested as the backup account address is simple. If a sender accidentally sends the cryptoasset to a non-existent address, the cryptoasset should be returned to the sender. If a receiver forgets or loses the private key, the cryptoasset should be returned to the sender because the sender and the receiver know the history of the transaction between them. If a receiver dies without passing on a private key to family or relatives, the cryptoasset similarly should be returned to the sender because the participants in a transaction know the history of the transaction between them. They are good partners to negotiate on the terms and conditions of the return of the cryptoasset in question.

To sum up, our method prevents unintended cryptoasset losses by employing the aforementioned fail-safes. Through this method, Alice, the sender (having address *A*), sets a time limit *T* and specifies a backup account's address *S* (actually *A*, as suggested in this paper) before sending cryptoasset *X* to Bob, the receiver (having address *B*). The KSC, which allows users to set such parameters and use the proposed method, is assumed in this proposal to be a contract account with a fixed, well-known address for the sake of convenience. Both parties must agree in advance on deadline *T* and the backup account's address *S* when using the KSC. The deadline *T* serves as a grace period for Bob, the rightful receiver, associated with the address *B* to claim the cryptoasset *X*. Even after the deadline has passed, Bob can claim the cryptoasset if the cryptoasset has not been moved to the backup account, as long as he keeps the private key.

Table 2. Notations used for the method

Symbol	Meaning
K	Address of Kim smart contract
A	Address of a sender's account
B	Address of a receiver's account
S	Address of a backup account
I(*f*)	Address of the individual who calls the function *f*
X	Amount of cryptoasset
T	Value of a deadline
C	Value of the current time
R	Value of the receipt

Thus, we need three kinds of functions in the KSC. The first function sets the deadline and specifies the backup account's address. This function sends *X* from Alice's account to the KSC itself, from which the receiver can claim *X*. The KSC exists at the address *K*. The second function claims the unclaimed cryptoasset *X* from the KSC to the receiver's account. The third function moves the unclaimed cryptoasset *X* from the KSC to the backup account.

Upon agreement between Alice and Bob, Alice sends *X* to Bob. However, *X* is not transferred directly to Bob's account, but to the KSC. The KSC allows us a degree of freedom to

revert, because Alice and Bob may agree to set *S* to Alice's address. But without KSC (i.e., in the case of traditional transactions), once *X* is placed in the receiver's account, it cannot be returned without having the private key.

Bob, the recipient, should execute the function *Claim* to send the cryptoasset *X* from the smart contract to his account in order to put the digital currency into his own possession. This execution can be proceeded regardless of whether the deadline has passed.

After the deadline, anyone can execute the function *Move* to move the cryptoasset in the KSC to the backup account (with the address S) as long as the cryptoasset remains unclaimed. If the cryptoasset has been withdrawn by the receiver from the KSC or moved to the backup account, the function *Move* fails. The cryptoasset *X* moved from the KSC cannot be claimed by Bob.

Alice can make the cryptoasset refundable to her by setting *A* as *S*. A flowchart for the workflow logic of the method is provided in **Fig. 2**. This method is an on-chain solution, and it is applicable to all blockchain platforms that support smart contracts. It can also be implemented on any blockchain platform that does not support smart contracts, if the platform has time-limiting functionalities (e.g., Bitcoin has the *nLockTime* parameter). If it cannot be implemented on-chain, the use of side channels should be considered, as Decker and Wattenhofer [30] did for their work. The KSC is described in the following paragraphs.

3.1 Initialization

To initialize the KSC, a trusted third party (TTP) needs to deploy the KSC, which is based on the pseudo-code shown in **Table 3**. The contract helps individuals to make recoverable transactions between themselves. The ready-made KSC can be reused multiple times. The TTP needs smart contract auditing to identify the vulnerabilities of the contract to mitigate damage. To manage vulnerabilities, following studies that identify the well-known attacks and protections for them, such as the study [31], would be appropriate. Nonetheless, there are some open challenges [32] in blockchain technology, thus the party who deploys the contract should be prepared for possible attacks in the future. The TTP also needs to make the smart contract public, as the blockchain community does, for transparency.

3.2 Sending the Cryptoasset

To send the cryptoasset, a sender has to make a transaction that sends cryptoasset *X* to the KSC's address by calling the *Send* function. The sender should also set values for *B*, *S*, and *T*. Upon receiving the call, the function sets the receiver's address, the backup account address, the deadline, and the amount of the cryptoasset. The function also sets to *false* the Boolean value that corresponds to the transaction. This Boolean value will be used to determine whether the cryptoasset has already been claimed or moved.

After the function call is completed (i.e., the transaction for the function call written on the block), the sender receives a receipt *R*, which is a hash value based on *A*, *B*, *S*, *C*, *T*, and *X*. This receipt will be used to identify the particular set of transaction data. Here, *C* is the value of the current time, which is usually a blockchain-native calculable value (*block.timestamp* in Solidity).

If multiple calls of the *Send* function with the same parameters are made in a short period of time, their receipts are rarely identical. However, their receipts may be identical theoretically when *A*, *B*, *S*, *C*, *T*, and *X* are all identical. This is due to the fact that the current time value *C* is based on the block's time stamp. Because blocks are created in a non-continuous manner according to the block creation rate, the time stamps of calls within a block are the same, resulting in the same value of receipt *R*. If this is the case, the KSC interprets such calls

as an attempt to send multiple *X*s to *B*. Thus, if *n* identical calls (including the current time parameter *C*) are made, *n* times *X* amount of cryptoassets will be moved when the *Claim* or *Move* function is executed with the identical receipts.

Table 3. Pseudocode of the KSC

Global Variables: $B(R)$, $S(R)$, $T(R)$, $X(R)$, and $Bool(R)$ map a receipt to the receiver's address, backup address, deadline, amount of cryptoasset, and Boolean value, respectively. R is a value of the receipt.

```
1  Function Send (b, s, t, x) :
2      R ← hash( A || b || s || C || t || x )   // The address of the sender's account A and the
                                                   current time value C are blockchain-native calculable values.
3      B(R) ← b
4      S(R) ← s
5      T(R) ← t
6      X(R) ← X(R) + x
7      Bool(R) ← false
8  return R
9
10 Function Claim (R) :
11     if I (Claim) ≠ B(R) then
12         REVERT
13     else
14         if Bool(R) ≠ false then
15             REVERT
16         else
17             Bool(R) ← true
18             Send X(R) to B(R)
19         end
20     end
21 return
22
23 Function Move (R) :
24     if C ≤ T(R) then
25         REVERT
26     else
27         if Bool(R) ≠ false then
28             REVERT
29         else
30             Bool(R) ← true
31             Send X(R) to S(R)
32         end
33     end
34 return
```

3.3 Communication

An individual (having address *I*(*f*)) who wants to claim or move cryptoasset *X* needs to call the KSC's *Claim* or *Move* function using *R*. Here, *I*(*f*) is the address of the individual who wants to call function *f*. To find out *R*, the individual can: (1) ask the sender for *R* using any communication channel, or (2) look into the KSC account. The address of the account (*K*) is a well-known address.

3.4 Claiming the Cryptoasset

The owner of *B* is the only one who can execute the *Claim* function (*B* is equal in value to the address *I*(*Claim*)), and the Boolean value that corresponds to the receipt must be *false*. Otherwise, the contract executes *REVERT*. If all the conditions are met, the contract converts the Boolean value to *true* and sends *X* to *B*. Because *I*(*Claim*) is usually a blockchain-native calculable value (*msg.sender* in Solidity [33]), the use of this parameter is not a considerable burden when it is implemented.

3.5 Moving the Cryptoasset

Only after the deadline (when *C* is greater than *T*) is it possible to execute the *Move* function. In addition, the Boolean value that corresponds to the receipt must be *false*. Otherwise, the contract executes *REVERT*. If all the conditions are met, any *I*(*Move*) can make the contract convert the Boolean value to *true* and send *X* to *S*.

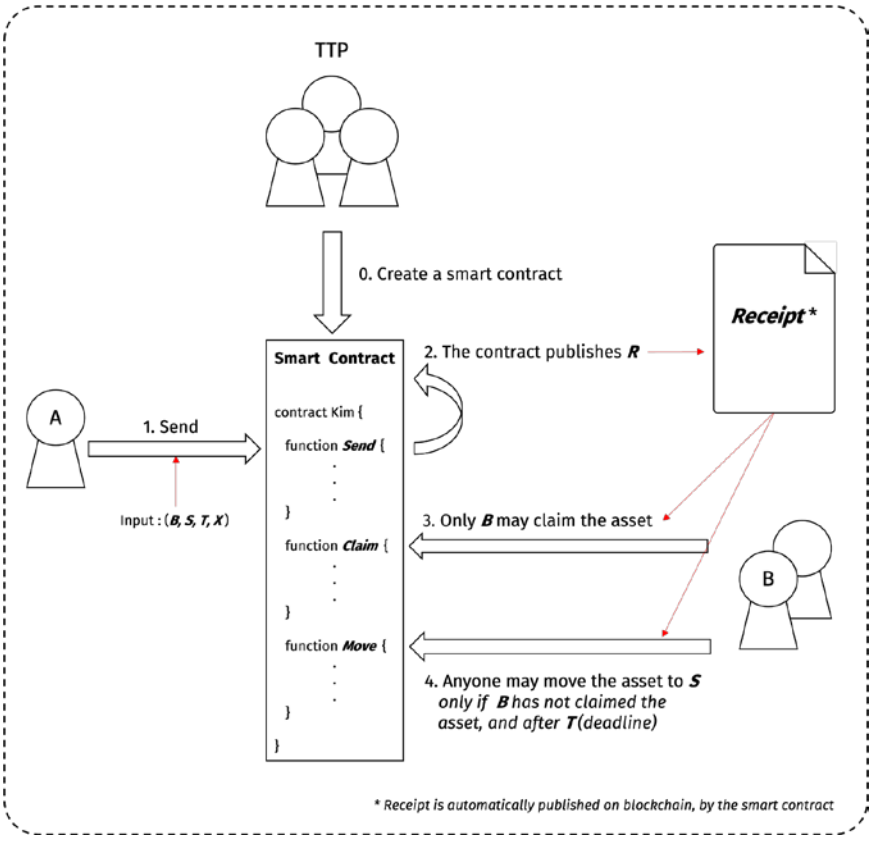

Fig. 3. Workflow for the KSC.

4. Discussion

In this section, the analysis and limitations of the proposed method are discussed. The analysis provides insight into the computational workload the proposed method requires. The proposed method is summarized in **Fig. 4**.

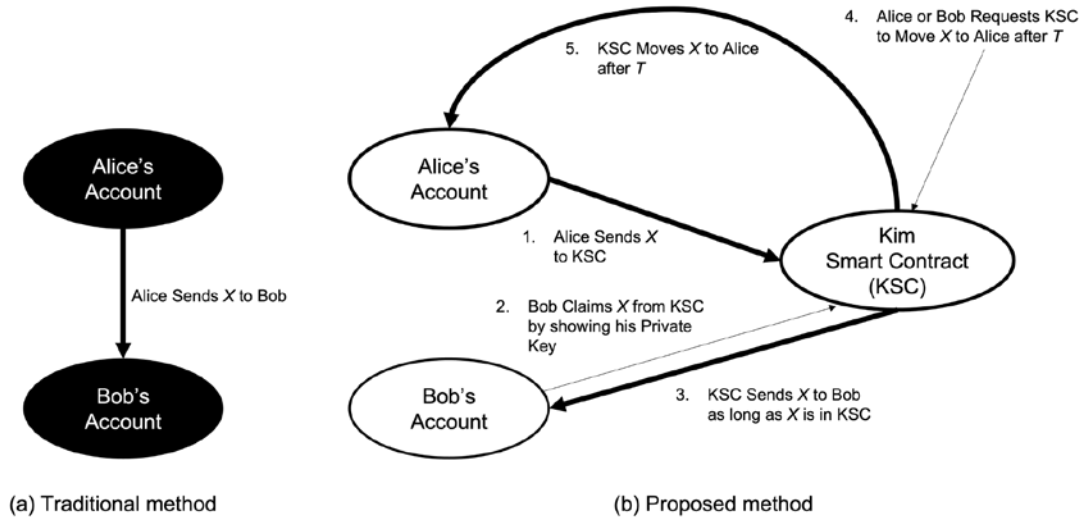

Fig. 4. Comparison between the traditional transaction method and the proposed method (backup account is set as Alice's account, as suggested).

4.1 Analysis

To analyze the proposed method, we created an Ethereum-based prototype using the Solidity smart contract language. The prototype transferred Ethereum-native cryptoassets (i.e., Ether) as proposed.

$$(Trasaction\ fee) = (Gas\ Used) \times (Gas\ Price) \tag{1}$$

On the EVM, transaction fees are calculable using Equation (1). The gas price is dependent on the network status, whereas the gas used is dependent on the computing power the code execution needs, which is pre-defined on the EVM [34]. Thus, evaluation of gas consumption for the proposed prototype provides insight into the computational workload of the proposed method, which can be widely applied on EVM-compatible blockchain platforms (e.g., Binance Smart Chain, Matic Network, Tron, Klaytn, and so on).

Table 4. Amount of gas consumed for each execution

Execution	Gas Consumption
Initialization by TTP	$\sim 4.67 \times 10^5$
Send	$\sim 1.14 \times 10^5$
Claim	$\sim 5.58 \times 10^4$
Move	$\sim 5.79 \times 10^4$

The gas consumption of the prototype shown in **Table 4** is calculated based on "London EVM" [35]. The *Send* function consumed about 1.14×10^5 gas, which is approximately four times greater than the 2.10×10^4 of the traditional transaction. For the *Claim* and *Move* functions, their consumption in gas was about 5.58×10^4 and 5.79×10^4, respectively.

It appears that the method necessitates greater gas consumption than does the traditional transaction, which is not recoverable. However, the users who need the proposed method would want to use it for security reasons while managing large amounts of cryptoassets. In other words, we assume that they would not hesitate to use the method due to the increased gas fee when transferring large amounts of cryptoassets.

In the blockchain environment, how long a function call of a smart contract takes depends on two factors: (1) pre-defined block creation rate, which varies depending on the platform a smart contract is implemented on, and (2) the platform's network congestion level, which is based on the demands for transactions and the capacity of each block. Therefore, the analysis of gas consumptions for functions on KSC may appear sufficient to evaluate the proposed method. However, to enlighten those who are unfamiliar with the technical background of the blockchain, time consumption for each function call is also analyzed in **Fig. 5**, in order to evaluate the performance of the KSC when it is run on a single node. This experiment is based on a local EVM, with a virtual machine configured as shown in **Table 5**.

Table 5. A virtual machine configuration, which runs the KSC on local EVM

Operating System	Ubuntu 20.04
Processors	4 processor cores, for i7-10700 host CPU
Memory	8 GB RAM

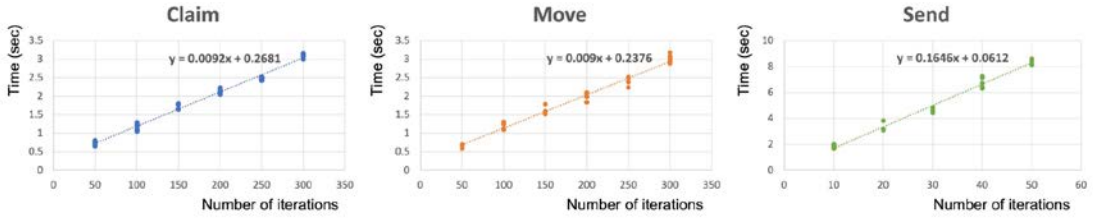

Fig. 5. Time consumed on multiple iterations of the KSC's function call, for *Claim*, *Move* and *Send* functions from the left.

The experiment is carried out for multiple calls to the KSC's *Claim*, *Move*, and *Send* functions, with the maximum number of iterations determined by the actual Ethereum node's performance. To make the experiment meaningful, the function calls were made without the Boolean value that checks for redundancy. Also, to reduce errors in the results, each execution for the number of iterations was repeated five times. Other parameters, such as the amount of the cryptoasset, were fixed throughout the experiment to control the experiment environment.

The gradient and y-intercept of the results' linear trend line show us the approximate value of an execution time for each function, as well as the overhead for a function call. As a result, it can be examined that each call to the *Claim* and *Move* functions takes approximately 0.0092 seconds and 0.009 seconds to complete, respectively, while the *Send* function takes 0.1646 seconds. The overhead for the call to *Claim*, *Move* and *Send* functions are 0.2681, 0.2376, and 0.0612 seconds, respectively. The difference in approximate execution time between the *Claim* and *Move* functions is due to the conditional statement declared on each function, which is the only logical difference between them.

4.2 Limitations

People may find it rather difficult to use the proposed method because such transactions require calling the KSC. To enhance users' accessibility, those interested in implementations can create a decentralized application (DApp) [36], as shown in **Fig. 6**.

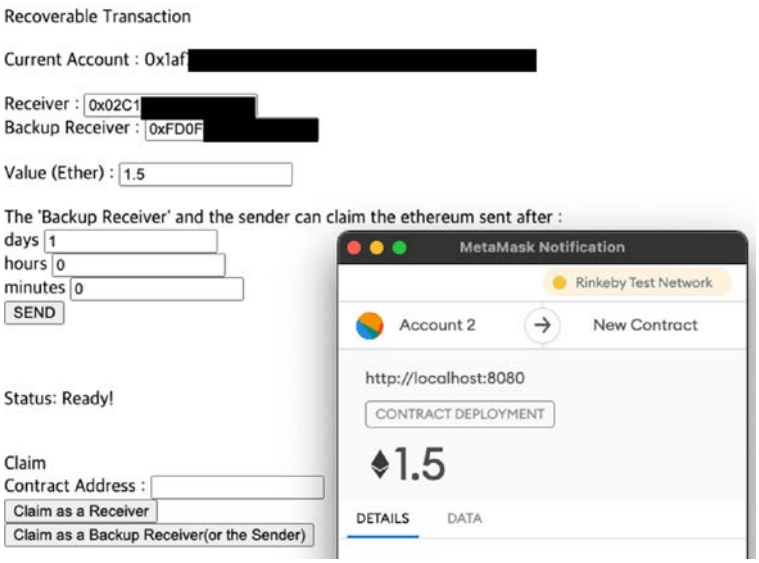

Fig. 6. Simple web-based DApp for the prototype.

To collect time information on blockchain, the timestamps of blocks must be used. However, timestamps are not a true real-time value because they are quantized by the block creation rate. As a result, participants should be aware that the time limit they set may behave in a manner other than intended.

4.3 Security

The security of the proposed method is heavily reliant on the security of smart contract itself. Thus, as mentioned in Section 3.1, potential vulnerabilities need to be addressed at the initialization stage. Aside from such vulnerabilities inherent in smart contract, security aspects of the method that rely on the security of blockchain platform that runs the KSC should be analyzed.

Function calls between individuals and the KSC, which take the form of transactions, are used to realize the proposed method. Because of security techniques such as transaction nonce, attacks on such transactions are difficult to realize in major blockchain platforms. As a result, exemplary blockchain attack techniques such as double-spending and replay attacks [37] are not available against the proposed method unless and only if the blockchain platform that runs the method is resistant to the attacks. Resistance to other possible attacks, such as denial of service (DoS) and block withholding, can also be deemed sufficient if the blockchain platform is resistant to such attacks. In conclusion, security of the proposed method is dependent on the security of the blockchain platform itself.

5. Conclusion

Blockchain's immutable nature has protected its users from malicious manipulation of the ledger. However, as there are no centralized authorities that can change the state of the ledgers, there have been few solutions for unintended cryptoasset losses. In this paper, a recoverable transaction method is proposed that can restore unintended cryptoasset losses while not infringing on the blockchain's immutability. By analyzing the proposed prototypes, the computing power requirements were studied. The limitations of the proposed work that users should be aware of were discussed. This recoverable transaction method will reduce the rate of unintended cryptoasset losses on blockchain and provide insights to those who are implementing DLTs in their cryptoasset management. It would be desirable for CBDC designers to consider the proposed method to enhance security and user satisfaction.

References

[1] K. Iyer and C. Dannen, *Building Games with Ethereum Smart Contracts*, Berkeley, CA, USA: Appress, 2018.

[2] C. R. Moulton, "Article 4A: Erroneous funds transfers and the discharge for value rule," *Annu. Rev. Bank. Law*, 1996. Article (CrossRef Link)

[3] S. John, "How to cancel a PayPal payment if the receiver has not yet claimed it," 2019, [Online]. Available: https://www.businessinsider.com/how-to-cancel-paypal-payment, Accessed on: Sep. 22, 2021.

[4] L. Hays, "How to cancel an apple pay payment on iphone or ipad," 2021, [Online]. Available at: https://www.iphonelife.com/content/how-to-cancel-payment-apple-pay-your-iphone, Accessed on: 2021-09-22.

[5] M. Wilkens, "Privacy and security during life, access after death: Are they mutually exclusive?," *Hastings Law J.*, vol. 62, no. 4, pp. 1037–1064, 2011. Article (CrossRef Link)

[6] M. I. Mehar, C. L. Shier, A. Giambattista, E. Gong, G. Fletcher, R. Sanayhie, H. M. Kim, and M. Laskowski, "Understanding a revolutionary and flawed grand experiment in blockchain: The DAO attack," *Journal of Cases on Information Technology (JCIT)*, vol. 21, no. 1, p. 14, 2019. Article (CrossRef Link)

[7] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. of BigData Congr.*, pp. 557–564, 2017. Article (CrossRef Link)

[8] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *Proc. of MIPRO*, pp. 1545-1550, 2018. Article (CrossRef Link)

[9] M. S. Ferdous, M. J. M. Chowdhury, M. A. Hoque, and A. Colman, "Blockchain consensus algorithms: A survey," *arXiv preprint arXiv:2001.07091*, 2020. Article (CrossRef Link)

[10] V. Gramoli, "From blockchain consensus back to Byzantine consensus," *Future Generation Computer Systems*, vol. 107, pp. 760-769, 2020. Article (CrossRef Link)

[11] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems* (*TOPLAS*), vol. 4, no. 3, pp. 382-401, 1982. Article (CrossRef Link)

[12] F. Yang, W. Zhou, QQ. Wu, R. Long, N. N. Xiong, and M. Zhou, "Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism," *IEEE Access*, vol. 7, pp. 118541-118555, 2019. Article (CrossRef Link)

[13] U. Tatar, Y. Gokce, and B. Nussbaum, "Law versus technology: Blockchain, GDPR, and tough tradeoffs," *Computer Law and Security Review*, vol. 38, 2020. Article (CrossRef Link)

[14] E. Politou, F. Casino, E. Alepis, and C. Patsakis, "Blockchain mutability: Challenges and proposed solutions," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 1972-1986, 2021. Article (CrossRef Link)

[15] H. Poston, "Mapping the OWASP top ten to blockchain," *Procedia Comput. Sci.*, vol. 177, pp. 613-617, 2020. Article (CrossRef Link)
[16] D. Perez and B. Livshits, "Smart contract vulnerabilities: Vulnerable does not imply exploited," *arXiv preprint arXiv:1902.06710v5*, 2020. Article (CrossRef Link)
[17] O. Zeppelin, "Proxy patterns," 2018. [Online]. Available: https://blog.openzeppelin.com/proxy-patterns, Accessed on: 2021-09-22.
[18] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2014.
[19] TokenPost, "$4.2b of lost bitcoin due to forgotten crypto wallet passwords can still be recovered," 2021. [Online]. Available: http://tokenpost.com/42B-of-lost-Bitcoin-due-to-forgotten-crypto-wallet-passwords-can-still-be-recovered-9238, Accessed on: 2021-09-22.
[20] H. J. Kim, "A method for recovering coin assets," Korea Patent No. 10-2021-0107177, 2021.
[21] PwC, "PwC global CBDC index 2021," PwC Tech. Rep., April 2021. Article (CrossRef Link)
[22] M. Kumhof and C. Noone, "Central bank digital currencies - design principles and balance sheet implications," *Bank of England working papers*, vol. 725, 2018. Article (CrossRef Link)
[23] D. Chaum, C. Grothoff, and T. Moser, "How to issue a central bank digital currency," *arXiv preprint arXiv:2103.00254*, 2021. Article (CrossRef Link)
[24] F. Vogelsteller and V. Buterin, "EIP20 (ERC20) token standard," 2015. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20, Accessed on: 2021-09-22.
[25] B. Notheisen, J. B. Cholewa, and A. P. Shanmugam, "Trading real-world assets on blockchain: An application of trust-free transaction systems in the market for lemons," *Bus. Inf. Syst. Eng.*, vol. 59, no. 6, pp. 425–440, 2017. Article (CrossRef Link)
[26] C. F. Torres, A. K. Iannillo, A. Gervais, and R. State, "The eye of Horus: Spotting and analyzing attacks on Ethereum smart contracts," *arXiv preprint arXiv:2101.06204*, 2021. Article (CrossRef Link)
[27] N. Popper, "Lost passwords lock millionaires out of their bitcoin fortunes," 2021. [Online]. Available: https://www.nytimes.com/2021/01/12/technology/bitcoin-passwords-wallets-fortunes.html, Accessed on: 2021-09-22.
[28] J. Voas and N. Kshetri, "Lost and never found," *Computer*, vol. 54, no. 07, pp. 12-13, 2021. Article (CrossRef Link)
[29] V. Cermak, "Can bitcoin become a viable alternative to fiat currencies? An empirical analysis of bitcoin's volatility based on a GARCH model," *SSRN Electron. J.*, 2017. Article (CrossRef Link)
[30] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Proc. of SSS 2015: Stabilization, Safety, and Security of Distributed Systems*, pp. 3-18, 2015. Article (CrossRef Link)
[31] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, pp. 24416-24427, 2020. Article (CrossRef Link)
[32] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. H. Nyang, and D. Mohaisen, "Exploring the attack surface of blockchain: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 1977-2008, 2020. Article (CrossRef Link)
[33] Ethereum, "Block and transaction properties," [Online]. Available: https://docs.soliditylang.org/en/v0.8.7/units-and-global-variables.html#block-and-transaction-properties, Accessed on: 2021-09-22.
[34] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, pp. 1-41, 2014. Article (CrossRef Link)
[35] V. Buterin, E. Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta, "EIP-1559: Fee market change for eth 1.0 chain," 2019. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1559, Accessed on: 2021-09-22.
[36] K. Wu, "An empirical study of blockchain-based decentralized applications," *arXiv preprint arXiv:1902.04969*, 2019. Article (CrossRef Link)
[37] N. Anita and M. Vijayalakshmi, "Blockchain security attack: A brief survey," in *Proc. of Int. Conf. Comput. Commun. Netw. Tech.*, pp. 1-6, 2019. Article (CrossRef Link)

Beomjoong Kim received his bachelor's degree in physics from Korea University in 2021. He is currently pursuing the Integrated Ph.D. degree in information security with Korea University. His research interests include blockchain and personal data.

Hyoung Joong Kim received his B.S., M.S., and Ph.D. degrees from Seoul National University in 1978, 1986, and 1989, respectively. He was a Professor with Kangwon National University from 1989 to 2006. He is currently a Professor with the School of Cybersecurity, Korea University. His research interests include distributed computing, machine learning, multimedia security, and cryptocurrency.

Junghee Lee received his B.S. and M.S. degrees in computer engineering from Seoul National University in 2000 and 2003, respectively, and his Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology in 2013. From 2003 to 2008, he worked at Samsung Electronics on electronic system level design of the mobile system-on-chip. From 2014 to 2019, he was with the Department of Electrical and Computer Engineering at The University of Texas at San Antonio as an Assistant Professor. He has been with the School of Cybersecurity, Korea University, since 2019. His research interests include secure design or hardware-assisted security of processors, non-volatile memory, storage, and dedicated hardware.