UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS720 Research Doc

MSERC20: Mutually secure ERC20

Cryptography in Blockchain: Implementation in Cryptocurrency systems

Name: Adir Miller

Student number: u20692286

https://github.com/MildogMiller/COS720Project

Contents

1 Overview

In recent years we have seen a rise in the use of cryptocurrencies from investments to stores actually accepting cryptocurrencies as payment [1][2][3]. While in recent years the popularity and market cap have decreased, it is still well known in the markets and is far from dead [4] [5]. Not only have we seen this increase in market cap we also see more types of cryptocurrencies emerge such as the NFT boom some years back [6][7]. These tokens were introduced into Ethereum [8], a Blockchain network that allows for decentralized contracts and code execution allowing for more versatile cryptocurrencies. The NFT in question was part of the ERC(Ethereum Request for Comments) standard ERC721 [9]. However the most popular token is the ERC20 [10]. ERC20 sits a base and allows one to define their own token while following a few rules listed. This is what is known as "Token Standardization". This allows external system to easily integrate since they all follow the same base [11]. Allowing for Dapps and Web3 systems to work with any valid token based off ERC20.

The basic principle of blockchain is digitally signed transactions that are grouped together to form blocks. Each one of these blocks are cryptographically linked to its predecessor by including the previous hash in its own hash, this creates the *chain* in blockchain. To ensure the integrity, the chain is distributed across a distributed network and before new blocks/transactions are added it must go through some *consensus protocol* where all nodes agree it should be added to the network. Lastly non repudiation is ensured by signing each transaction with a digital signature.

2 Research And Solutions

2.1 Issue at hand

Issuing and payment of invoices are generally based on trust. An invoice is issued then the client pays the amount on the invoice, however its common for issues to occur with incorrect payment amounts [12],[13]. We propose a crypto based solution to add an extra layer of trust and integrity. This involves the first party to *Request* payment from another before a transfer can succeed, further more that party must pay in a fixed time frame. The requesting party will need to specify an amount to be paid, the paying party will specify an amount and the payment will only go through if both parties specify the same amount. That way neither party can dispute the amount paid, this added layer helps create another layer of integrity

2.2 Solving the issue

To solve this issue we will create a new token that has this request feature built in and will only allow payment if there is a request for it and the right amount is sent through. This token also allows one to inherit it and add on extra features if required like a time delay.

2.3 ERC20

Since Smart contracts allow for so much more versatility you can use them to create your own currencies based off Ethereum. However There are some best practices that you can use to make sure that your proposed solution works with the current DAPPs and integrates with Web3. We have 2 categories of this, ERC(Ethereum Request for Comments) [14] and EIP(Ethereum Improvement Proposals).[15] This allows for more standardization of tokens. The most common token standard is known as ERC20[11][16]. This standard is by far the most popular when it comes to tokens [10]. The proposed solution will follow all listed structures and is a valid ERC20 Token.

2.4 Security of MSERC20

2.4.1 Request for payment

The token does not allow you at all to make a payment unless it has been requested from you. If there is any issues the code reverts and no transactions will go through. Party A will request an amount and list a time frame from Party B, when Party B pays uses the usual *transfer* of ERC20 it will only go through if there is a request. This makes payment compatible with the ERC20 Token standard.[11][16] This function helps solve the issue mentioned in the first section of incorrect payment amounts.

2.4.2 *Approve* Vulnerability

The ERC20 standard has a method that allows one to make payment on someone elses behalf, this is implemented however the request still needs to be made. There is however a vulnerability (race condition) in the base standard where one can actually spend more then there allowance. [17][18][19]. This has not been removed from the base standard for worry's of backwards compatibility. To fix this on my side i have used recommended practices dictated by [19] by using a "compare and set" of sorts to fix the race condition. This means the account holder first needs to set the value to 0 before setting it to any other value.

2.4.3 Logging

Logging is a fundamental part of computer security [20] and their role in *security incident prevention* [21]. It if for this reason I have added more events for the request logic to ensure the ease of logging by external systems that can identify for issues before they happen. While the blockchain itself is essentially a log, events improve lookup times drastically and can be used in observable systems to passively wait for events instead of scanning every new transaction. [22][23]. This allows for more automatic checks to be done improving its overall security.

2.4.4 Documentation

Documentation is extremely important [24][25] in this context of software engineering. It allows other developers to better understand the code and how to work with the code meaning they are less likely to make mistakes that would compromise security. This is why MSERC20 has been extensively documented to ensure that there is no confusion around the contract. MSERC20 follows the standard NATSpec [26] guidelines for solidity code documentation

2.4.5 Testing

Testing software is extremely important in the SDLC [27][28], as it helps ensure there are no errors or bugs in the program that could possibly be exploited. When it comes to smart contracts once they are deployed they cannot be taken down meaning testing is of the utmost importance. This is why MSERC20 has been extensively tested through unit and integration tests with 100% line coverage.

2.5 Security of network and token

The choice of network is Ethereum for its smart contracts which was the first of its kind [29]. The Ethereum network is updated constantly to meet security standards. Ethereum is also a network that prides it self in education of its users to good security practices [30]. It also has been updated over years and time between vulnerabilities found are increasing, the quantity decreasing and the avg severity is decreasing along side showing that over time this is becoming an extremely safe option. This can be proven by looking at the amount of CVEs over the years and their severity.

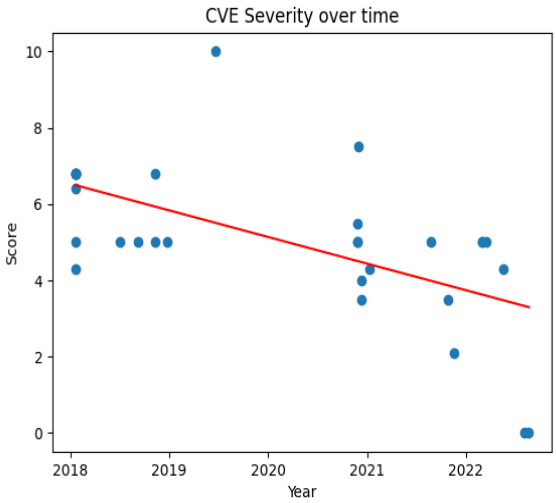

Figure 1: Data derived from [31]

Aggregates per year

Year	Publish Date	Update Date	Score
2018	14	3	5.950000
2019	1	3	10.000000
2020	6	6	5.083333
2021	4	4	3.725000
2022	6	14	3.216667

Figure 2: Data Derived from [31]

Above are 2 figures derived from CVE data [31]. As we can see from Figure: 1 that over time the severity of the CVEs are decreasing, this can be seen by the negative gradient of the line of best fit. This shows that over time the system is becoming more secure even though it became more popular. We can see from Figure Table: 2 that the update dates of the CVEs are mostly recent, this means that they are addressing or at least looking into the issues at hand. It is this level of security and attention to detail as seen in [30] leading me to choose Etherum as my network. ERC20 was chosen as the token base as its the most common token meaning it works with the most dapps and web3 apps making it easier to integrate with existing systems.

3 Documentation

3.1 IMSERC20

3.1.1 name

`function name() external view returns (string)`

Simple return for name

Return Values

Name	Type	Description
[0]	string	name of the currency

3.1.2 symbol

`function symbol() external view returns (string)`

Simple return for symbol

Return Values

Name	Type	Description
[0]	string	symbol of the currency

3.1.3 decimals

`function decimals() external view returns (uint8)`

Simple return for decimals

Return Values

Name	Type	Description
[0]	uint8	decimals the amount of decimals

3.1.4 totalSupply

`function totalSupply() external view returns (uint256)`

Simple return for totalSupply

Return Values

Name	Type	Description
[0]	uint256	totalSupply of the currency

3.1.5 balanceOf

`function balanceOf(address _owner) external view returns (uint256 balance)`

Returns the balance of an account

Parameters

Name	Type	Description
_owner	address	The owner of the account that of which balance details are being requested

3.1.6 transfer

`function transfer(address _to, uint256 _value) external returns (bool success)`

Main transfer request, this will simply call the internal function _maketransfer

Parameters

Name	Type	Description
_to	address	the person being paid
_value	uint256	the amount

3.1.7 transferFrom

`function transferFrom(address _from, address _to, uint256 _value) external returns (bool success)`

Function used to transfer on someone elses behalf. This emits the AllowanceTransfer event

Parameters

Name	Type	Description
_from	address	the paying account
_to	address	the account getting paid
_value	uint256	the amount

3.1.8 approve

`function approve(address _spender, uint256 _value) external returns (bool success)`

Function that dictates how much someone else can spend on your behalf. Value must first be set to 0 to prevent a race condition

Parameters

Name	Type	Description
_spender	address	the person who is spending on anothers behalf
_value	uint256	the approved amount to be spent

3.1.9 allowance

`function allowance(address _owner, address _spender) external view returns (uint256 remaining)`

Simple function that returns the current allowance allocated to a person

Parameters

Name	Type	Description
_owner	address	the person who is going to pay
_spender	address	the person who has permission to use funds from the others account

3.1.10 Approval

`event Approval(address owner, address spender, uint256 value)`

Event that is fired when allowance is changed

Parameters

Name	Type	Description
owner	address	the person who is the account holder
spender	address	the person spending on the others behalf
value	uint256	the amount they can spend

3.1.11 Transfer

`event Transfer(address from, address to, uint256 value)`

Event fired when a transfer is made

Parameters

Name	Type	Description
from	address	the person paying
to	address	the person being paid
value	uint256	the amount being paid

3.2 MSERC20

This contract requires to to request payment before its made

Hooks may be introduced in a later stage

3.2.1 request

```
struct request {
  uint256 amount;
  uint256 endtime;
}
```

3.2.2 _balances

```
mapping(address => uint256) _balances
```

3.2.3 _totalSupply

```
uint256 _totalSupply
```

3.2.4 constructor

```
constructor(string name_, string symbol_, uint16 requestLimit_, uint256 initial) public
```

Constructor of the class, will set the total supply,innitial supply and set this balance to the owner of the contract

Parameters

Name	Type	Description
name_	string	The name of the token
symbol_	string	The symbol of the token
requestLimit_	uint16	The max requests one account can request at a time
initial	uint256	The innitial cap of coins, this will be allocated to the owner

3.2.5 name

```
function name() public view returns (string)
```

Simple return for name

Return Values

Name	Type	Description
[0]	string	name of the currency

3.2.6 symbol

```
function symbol() public view returns (string)
```

Simple return for symbol

Return Values

Name	Type	Description
[0]	string	symbol of the currency

3.2.7 totalSupply

```
function totalSupply() public view returns (uint256)
```

Simple return for totalSupply

Return Values

Name	Type	Description
[0]	uint256	totalSupply of the currency

3.2.8 decimals

`function decimals() public pure returns (uint8)`

Simple return for decimals

Return Values

Name	Type	Description
[0]	uint8	decimals which is by default 18, however method can be overridden

3.2.9 balanceOf

`function balanceOf(address account) public view returns (uint256)`

Returns the balance of an account

Parameters

Name	Type	Description
account	address	that is being requested balance details of

Return Values

Name	Type	Description
[0]	uint256	balance of account

3.2.10 myBalance

`function myBalance() public view returns (uint256)`

Simple function that a requester can use to get their own balance

Return Values

Name	Type	Description
[0]	uint256	balance of requester

3.2.11 requestLimit

`function requestLimit() public view returns (uint16)`

Simple function that can be used to get the global request limit

Return Values

Name	Type	Description
[0]	uint16	balance of requester

3.2.12 isNotEmpty

`function isNotEmpty(struct MSERC20.request inrequest) internal pure returns (bool)`

Helper function that checks if a given request is null

Parameters

Name	Type	Description
inrequest	struct MSERC20.request	the request being checked

Return Values

Name	Type	Description
[0]	bool	requestnotempty which is boolean

3.2.13 getRequestExpiry

`function getRequestExpiry(address requester, address recipient) public view returns (uint256)`

Function to get the request expiry of an existing transaction

Parameters

Name	Type	Description
requester	address	of the request
recipient	address	of the request

Return Values

Name	Type	Description
[0]	uint256	requestExpiry of request

3.2.14 getRequestCount

`function getRequestCount() public view returns (uint32)`

Function that returns how many requests that you have made

3.2.15 getRequestAmount

`function getRequestAmount(address requester, address recipient) public view returns (uint256)`

Function to get the request amount of an existing transaction

Parameters

Name	Type	Description
requester	address	the person paying
recipient	address	the person being paid

Return Values

Name	Type	Description
[0]	uint256	requestAmount of request

3.2.16 transfer

`function transfer(address recipient, uint256 amount) public returns (bool)`

Main transfer request, this will simply call the internal function _maketransfer

Parameters

Name	Type	Description
recipient	address	the person being paid
amount	uint256	the amount

3.2.17 transferFrom

`function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)`

Function used to transfer on someone elses behalf. This emits the AllowanceTransfer event

Parameters

Name	Type	Description
_from	address	the paying account
_to	address	the account getting paid
_value	uint256	the amount

3.2.18 _maketransfer

`function _maketransfer(address sender, address recipient, uint256 amount) internal returns (bool)`

Internal function that actually does the transfer logic, this function is not overridable by design. This emits the Transfer event

Parameters

Name	Type	Description
sender	address	the person paying
recipient	address	the person being paid
amount	uint256	the amount

3.2.19 addNewRequest

`function addNewRequest(address requestee, uint256 amount, uint256 time) public returns (bool)`

Makes a new request from the caller of the contract to the requetee. This emits the NewRequest event

Parameters

Name	Type	Description
requestee	address	the person requesting contract caller is requesting payment from
amount	uint256	amount of the payment
time	uint256	the expiry of the request

3.2.20 _removeRequest

`function _removeRequest(address requester, address recipient) internal returns (bool)`

Helper function that is used to remove a request from the requests array. This emits the RemoveRequest event

Parameters

Name	Type	Description
requester	address	the person who is requesting payment
recipient	address	the person having payment requested from

3.2.21 removeRequest

`function removeRequest(address requester, address recipient) public returns (bool)`

Function that is called to remove a request from the pool, this can be either the requester or the recipient. This will simply return the internal _removeRequest function

Parameters

Name	Type	Description
requester	address	the person who is requesting payment
recipient	address	the person having payment requested from

3.2.22 approve

`function approve(address _spender, uint256 _value) public returns (bool success)`

Function that dictates how much someone else can spend on your behalf. Value must first be set to 0 to prevent a race condition

Parameters

Name	Type	Description
_spender	address	the person who is spending on anothers behalf
_value	uint256	the approved amount to be spent

3.2.23 allowance

`function allowance(address _owner, address _spender) public view returns (uint256 remaining)`

Simple function that returns the current allowance allocated to a person

Parameters

Name	Type	Description
_owner	address	the person who is going to pay
_spender	address	the person who has permission to use funds from the others account

3.2.24 NewRequest

`event NewRequest(address sender, address recipient, uint256 amount)`

Event that is fired when a new request is made

Parameters

Name	Type	Description
sender	address	the person who is requesting payment
recipient	address	the person having payment requested from
amount	uint256	the amount being requested

3.2.25 RemoveRequest

`event RemoveRequest(address actualSender, address sender, address recipient)`

Event that is fired when a request is removed

Parameters

Name	Type	Description
actualSender	address	the person calling this contract
sender	address	the person who is requesting payment
recipient	address	the person having payment requested from

3.2.26 AllowanceTransfer

`event AllowanceTransfer(address accholder, address sender, address recipient, uint256 amount)`

Event that is fired when someone makes a transfer on someone elses behalf

Parameters

Name	Type	Description
accholder	address	the account holder
sender	address	the person calling this contract
recipient	address	the person who is receiving money
amount	uint256	the amount being transfered

4 UML Diagrams

4.1 Sequence diagrams

4.1.1 Transfer

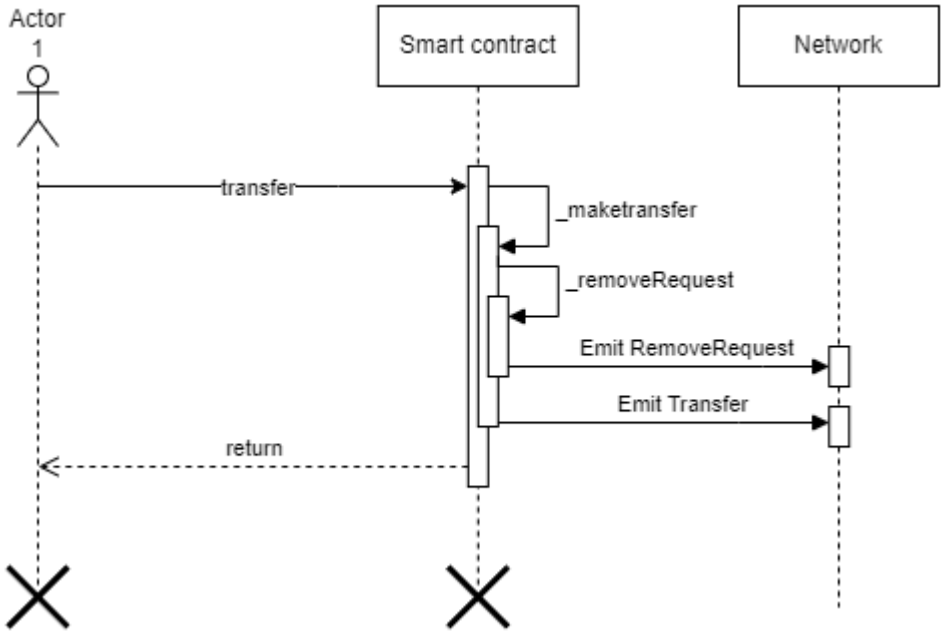

Figure 3: Transfer

4.1.2 Transfer From

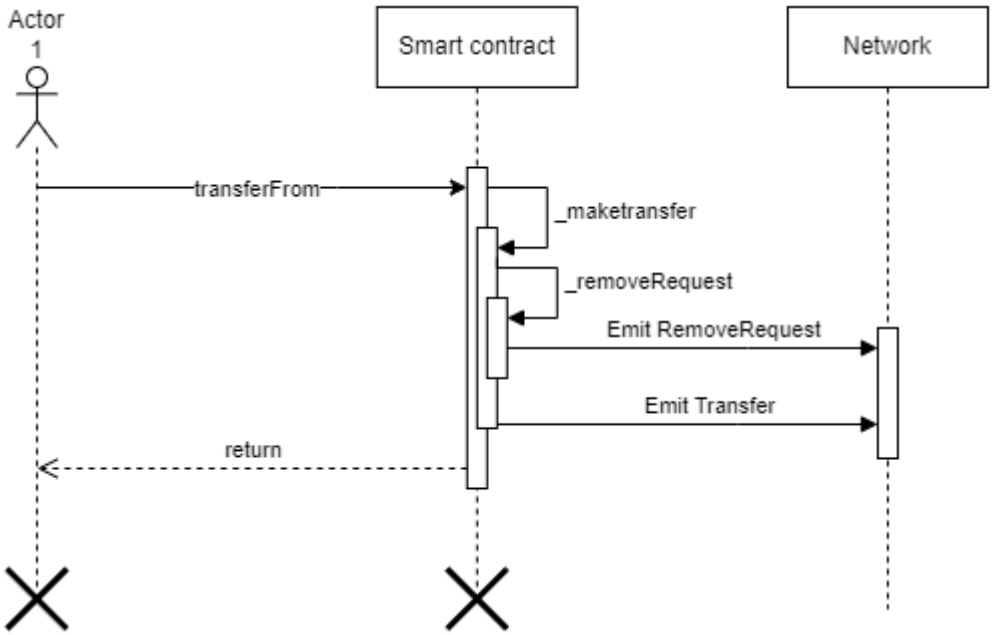

Figure 4: Transfer From

4.1.3 Add new request

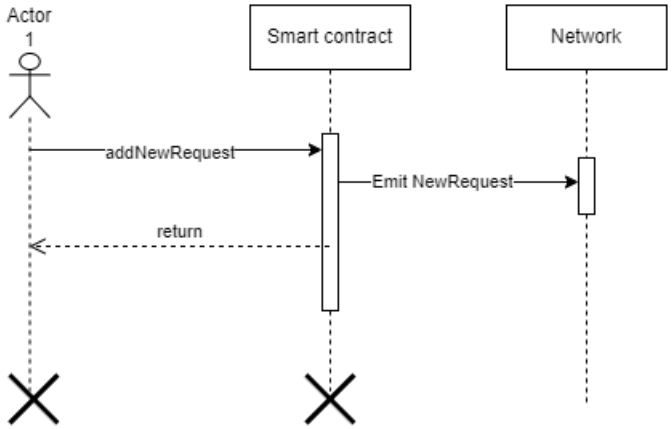

Figure 5: AddNewRequest

4.1.4 Remove request

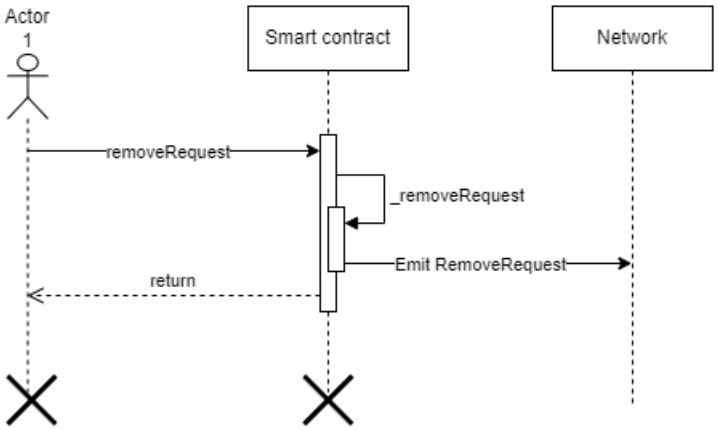

Figure 6: Remove request

4.1.5 Approve

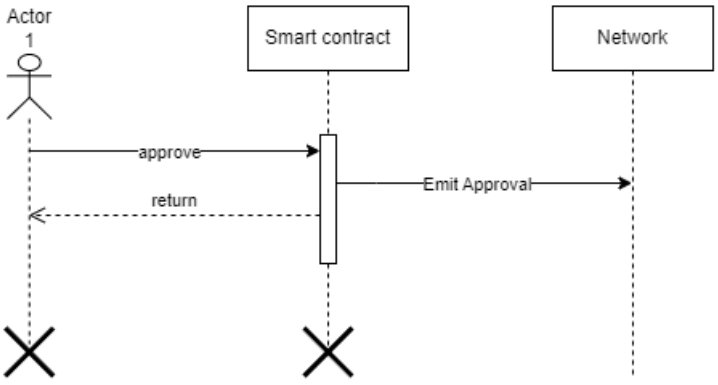

Figure 7: Approve

4.1.6 Allowance

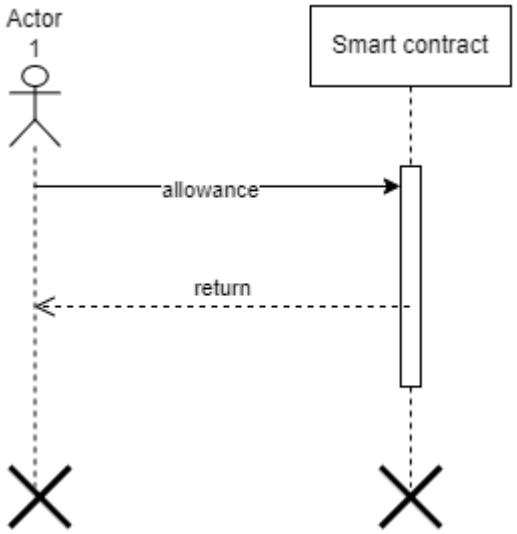

Figure 8: Allowance

4.1.7 BalanceOf

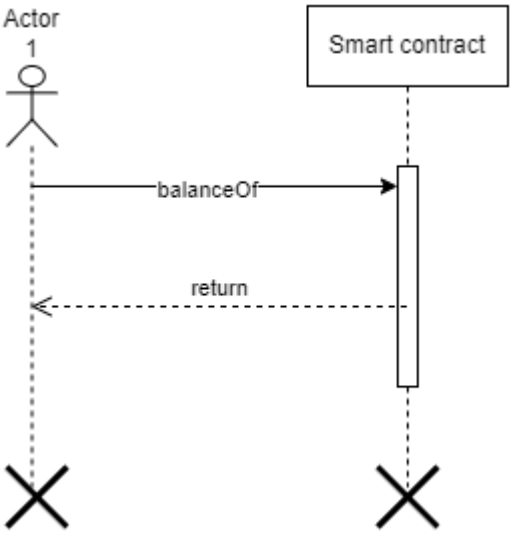

Figure 9: BalanceOf

4.1.8 Decimals

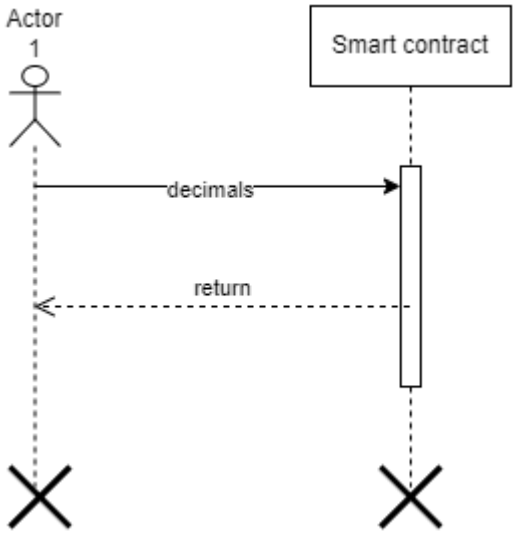

Figure 10: Decimals

4.1.9 getRequestAmount

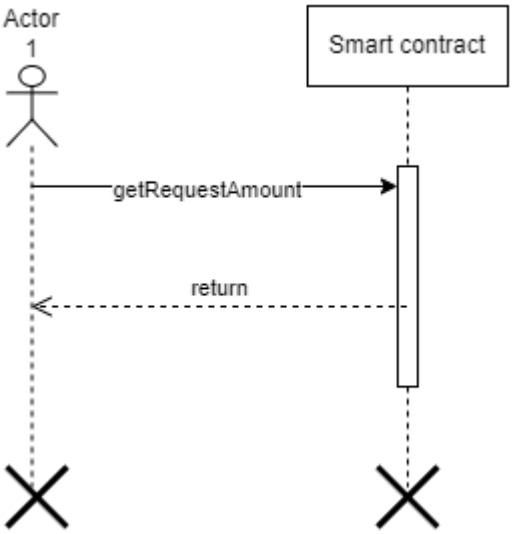

Figure 11: getRequestAmount

4.1.10 getRequestCount

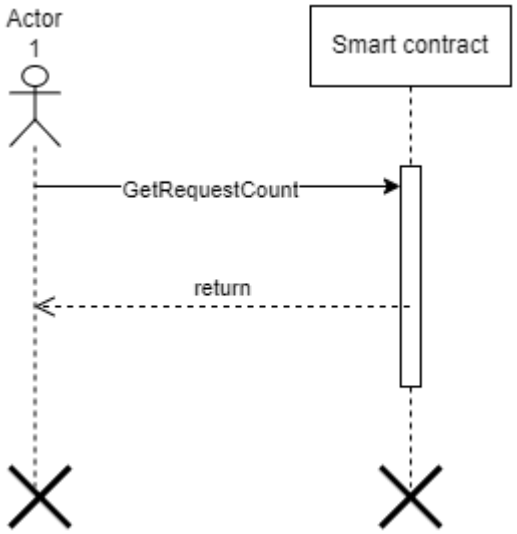

Figure 12: getRequestCount

4.1.11 getRequestExpiry

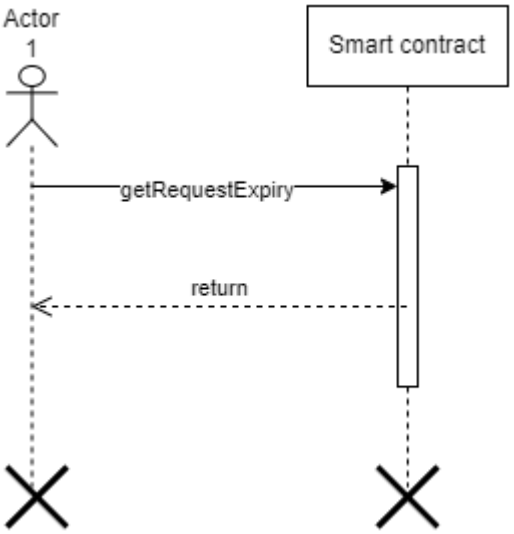

Figure 13: getRequestExpiry

4.1.12 name

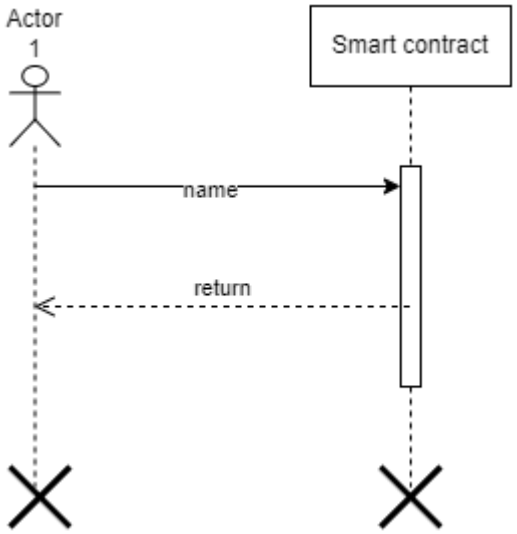

Figure 14: name

4.1.13 requestLimit

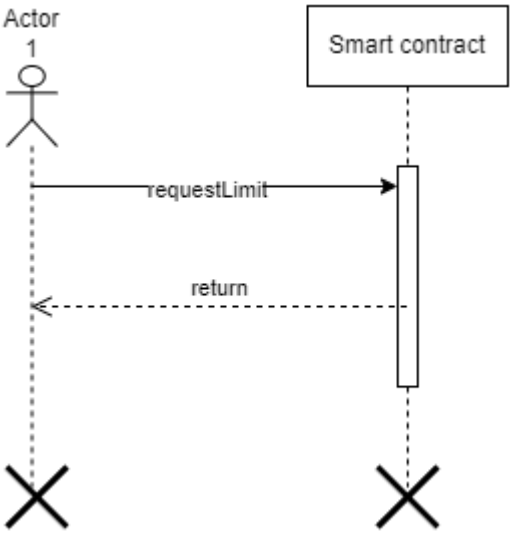

Figure 15: requestLimit

4.1.14 symbol

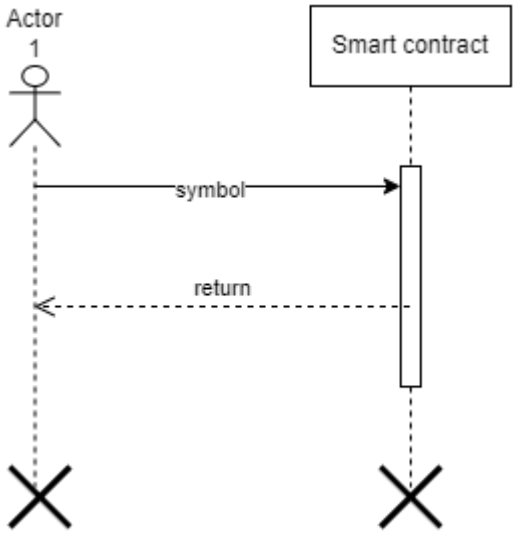

Figure 16: symbol

4.1.15 totalsupply

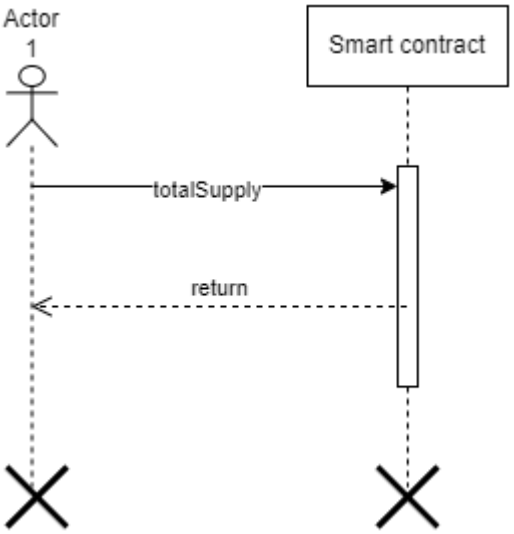

Figure 17: totalsupply

4.1.16 myBalance

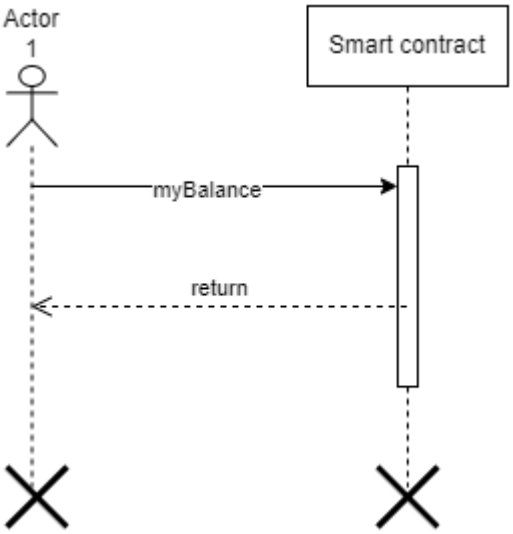

Figure 18: myBalance

4.2 Logic Flows

4.2.1 _makeTransfer

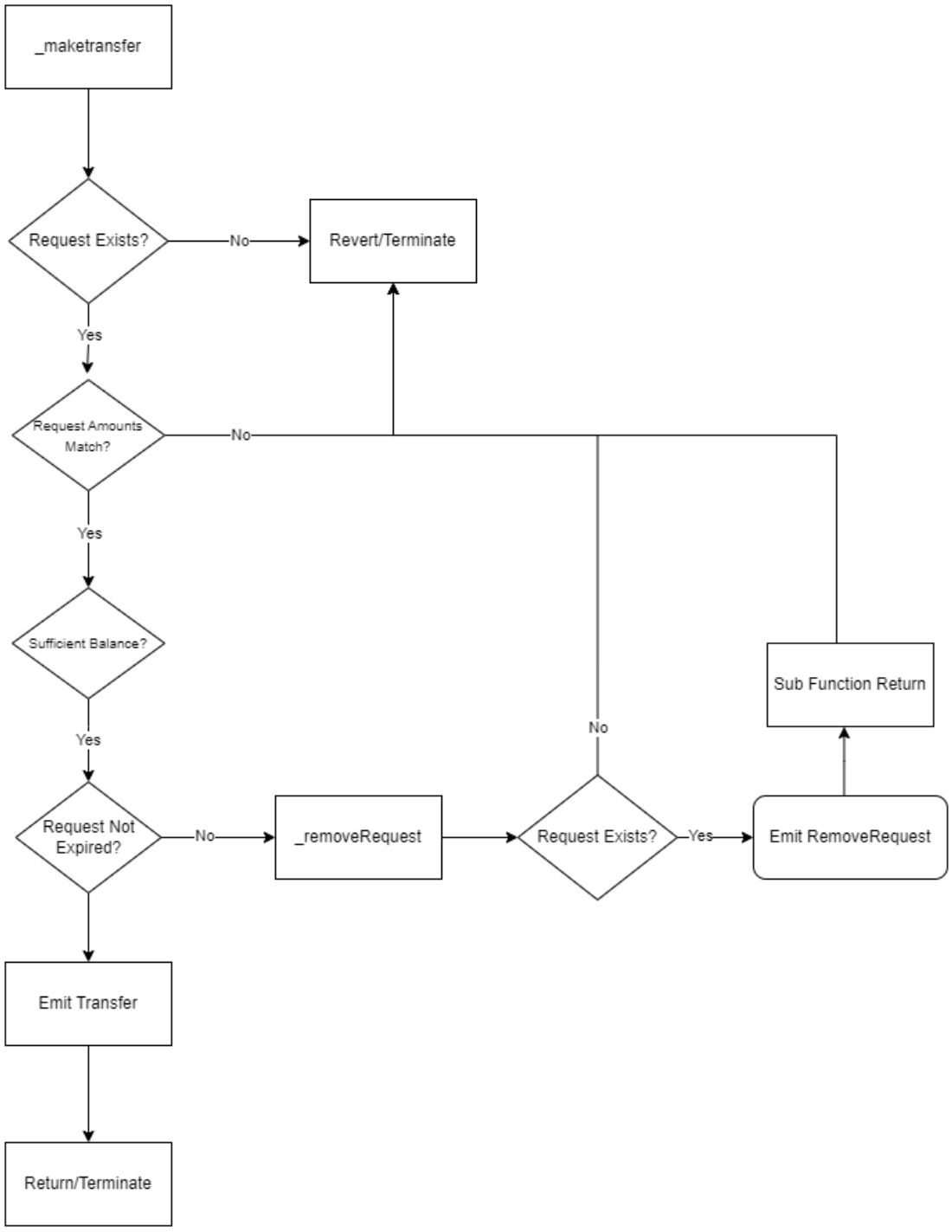

Figure 19: _makeTransfer

4.2.2 Transfer

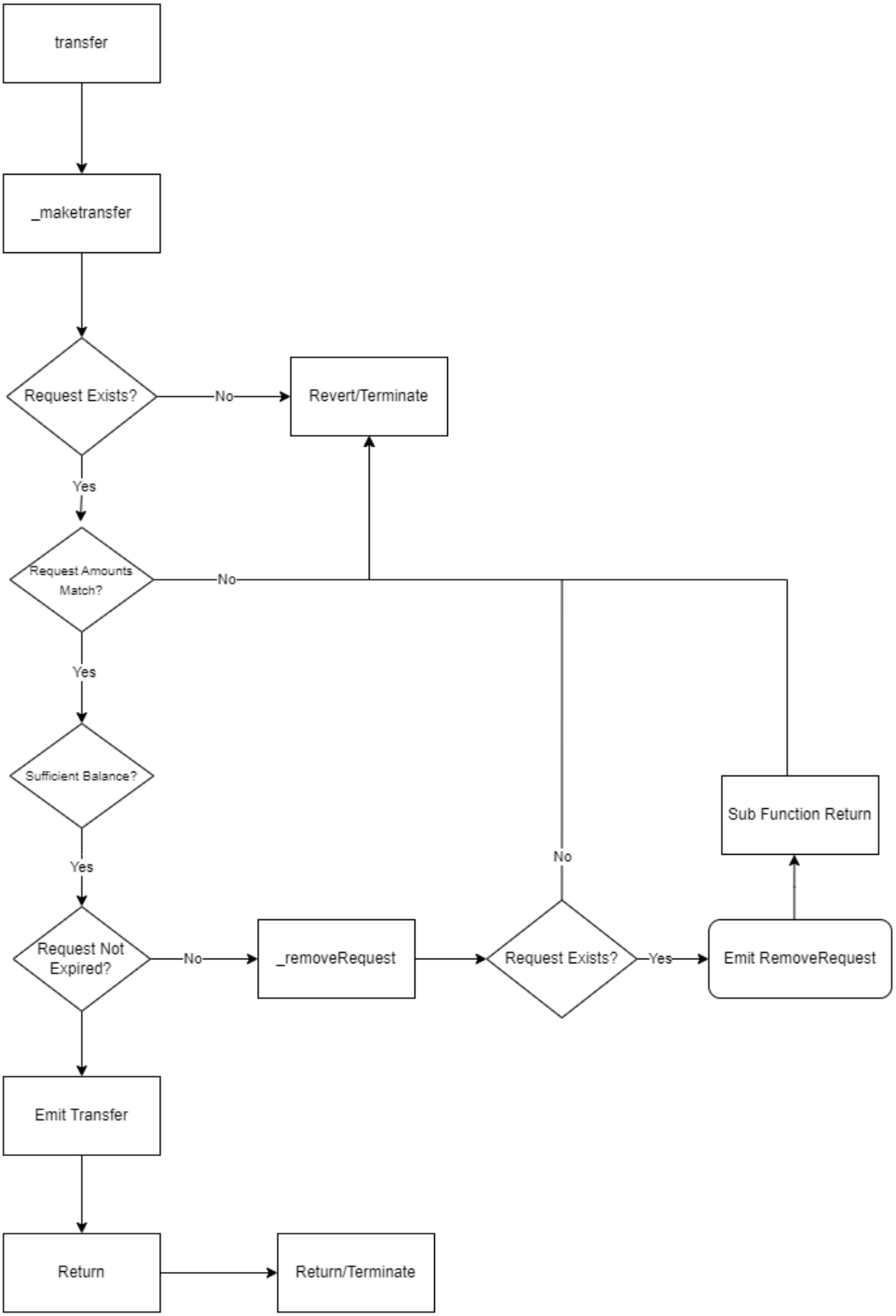

Figure 20: Transfer

4.2.3 Transfer From

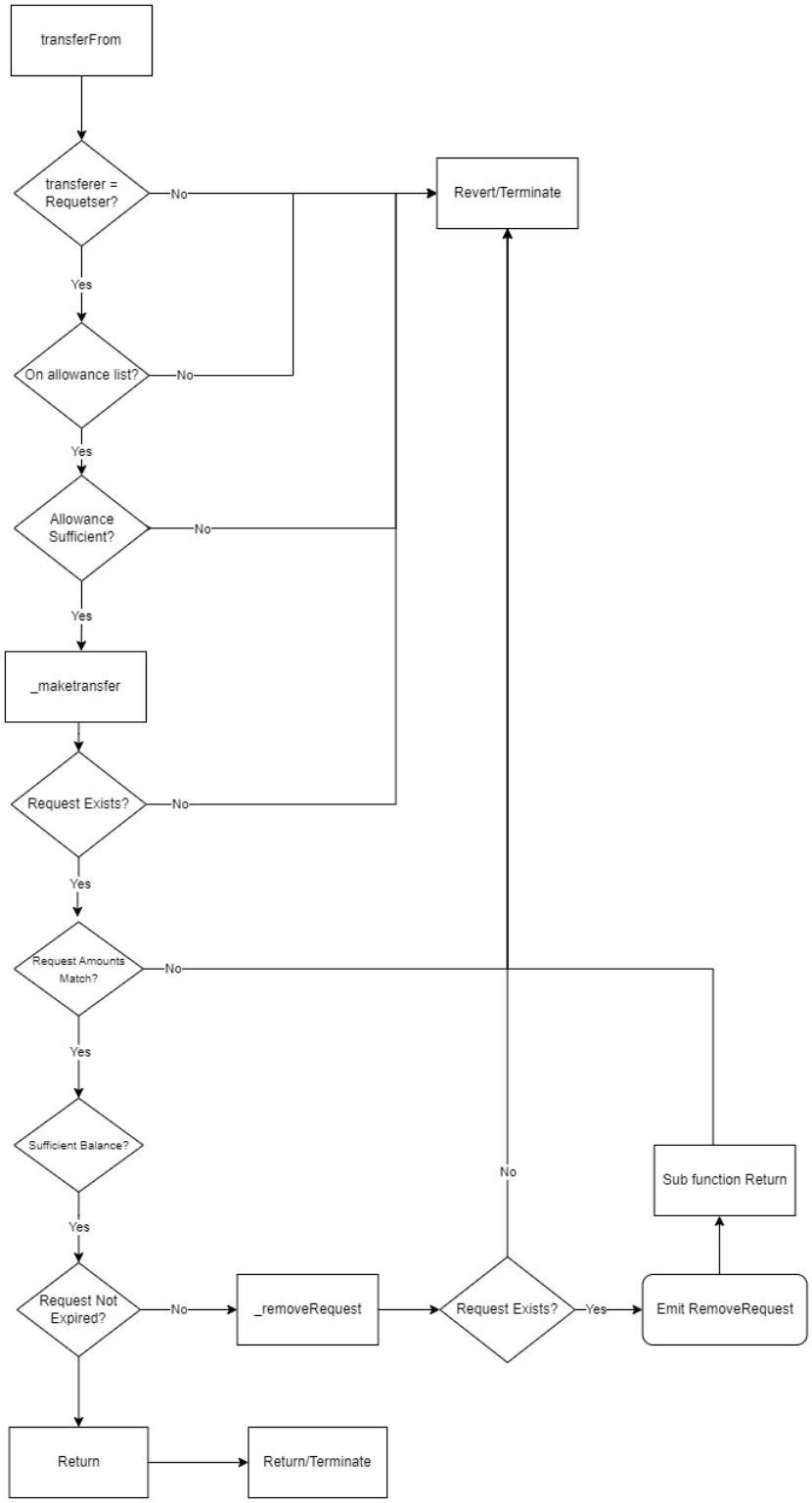

Figure 21: Transfer From

4.2.4 Add New request

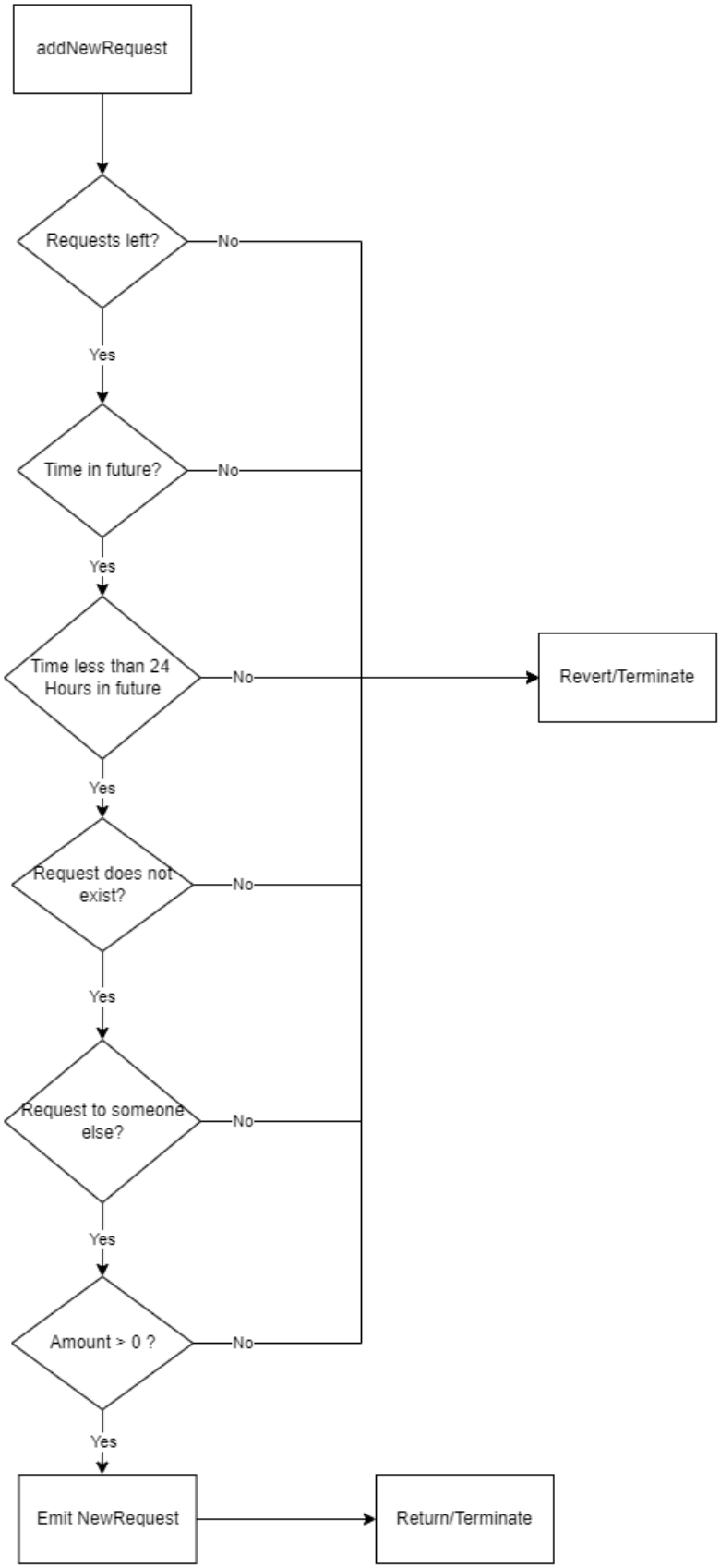

Figure 22: addNewRequest

4.2.5 Remove Request

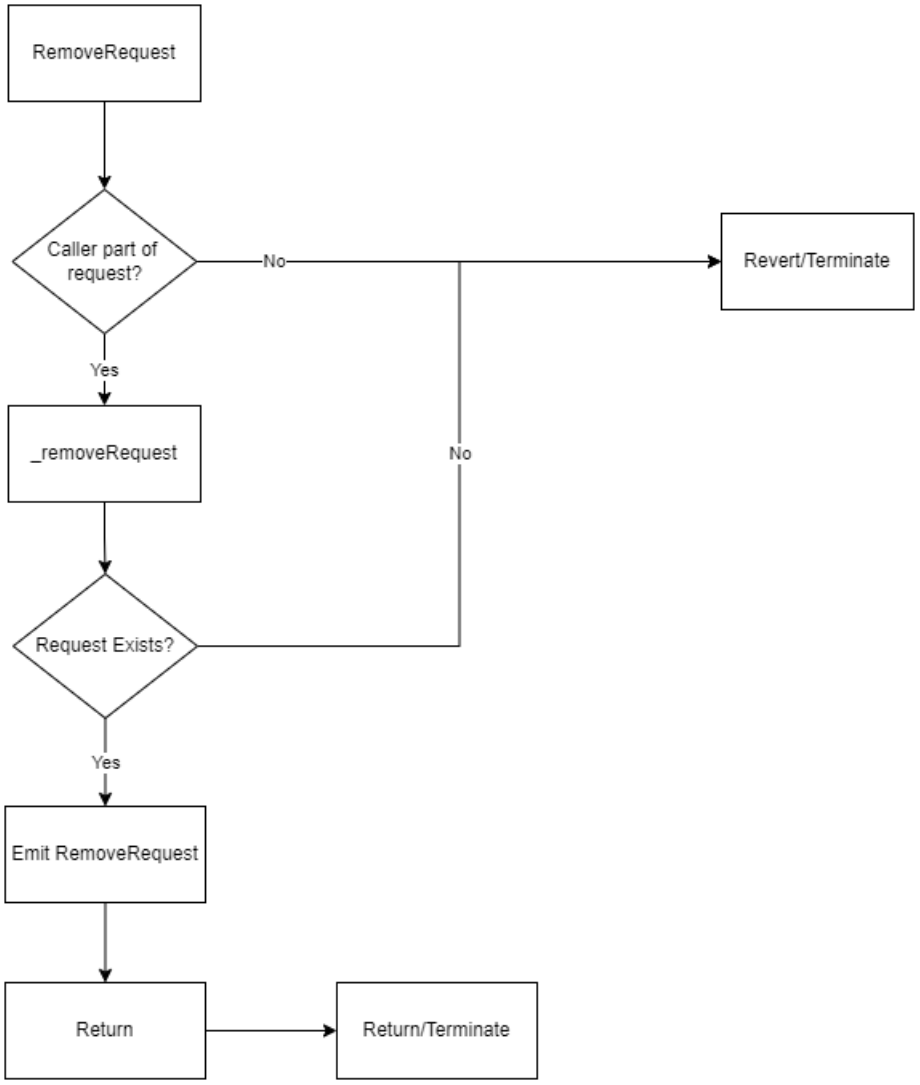

Figure 23: Remove request

4.2.6 approve

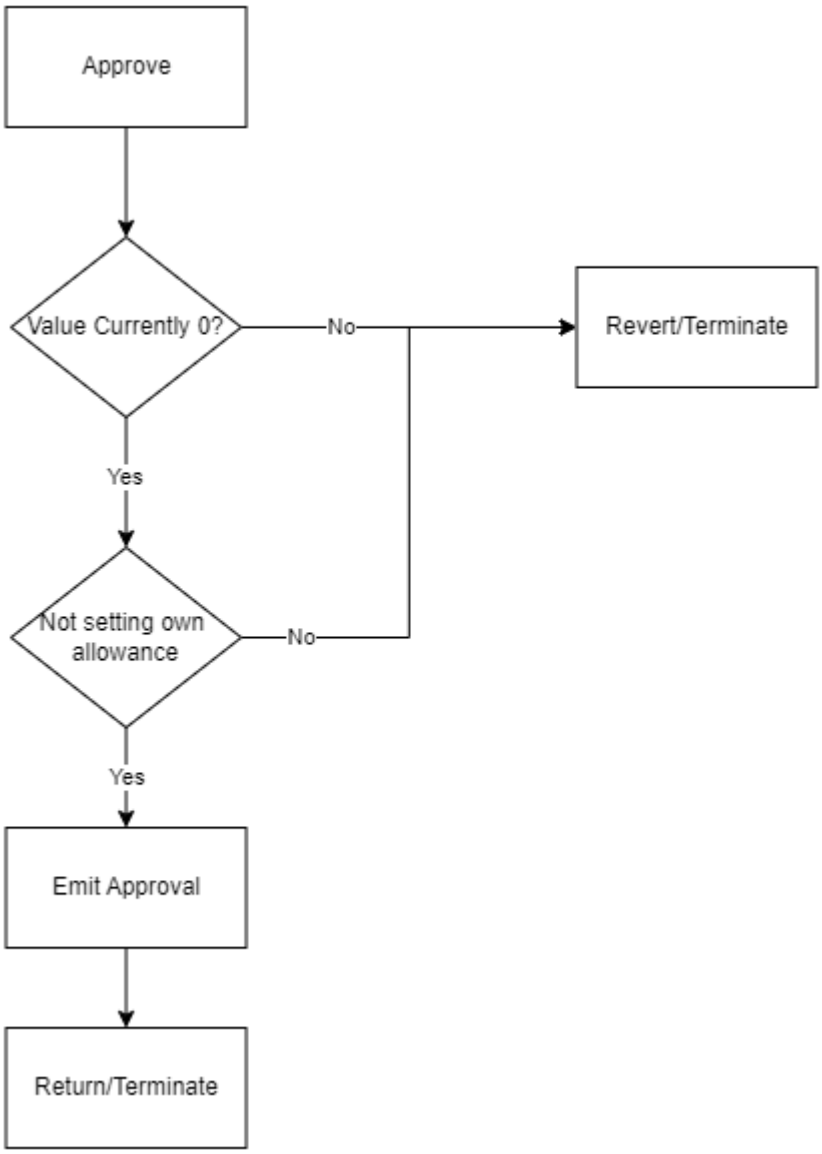

Figure 24: approve

4.2.7 getRequestAmount

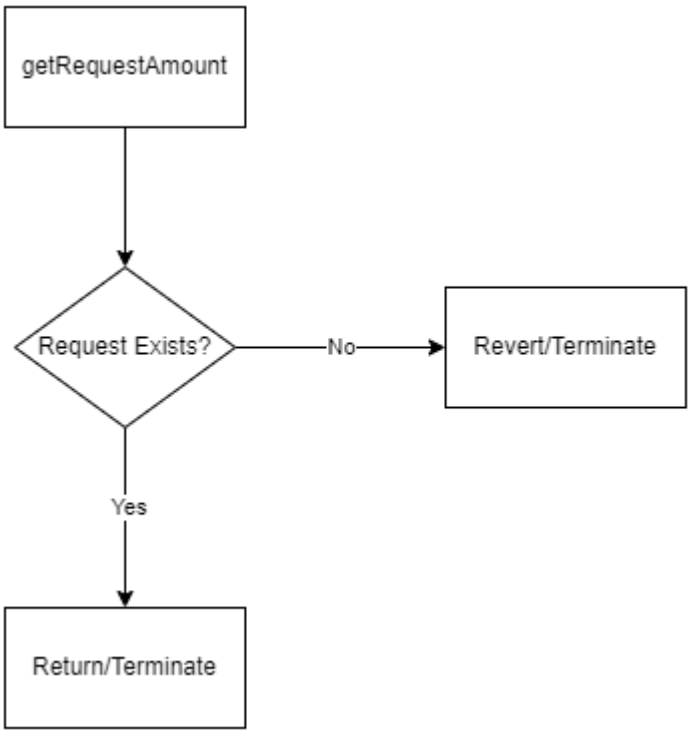

Figure 25: getrequestamount

4.2.8 _removeRequest

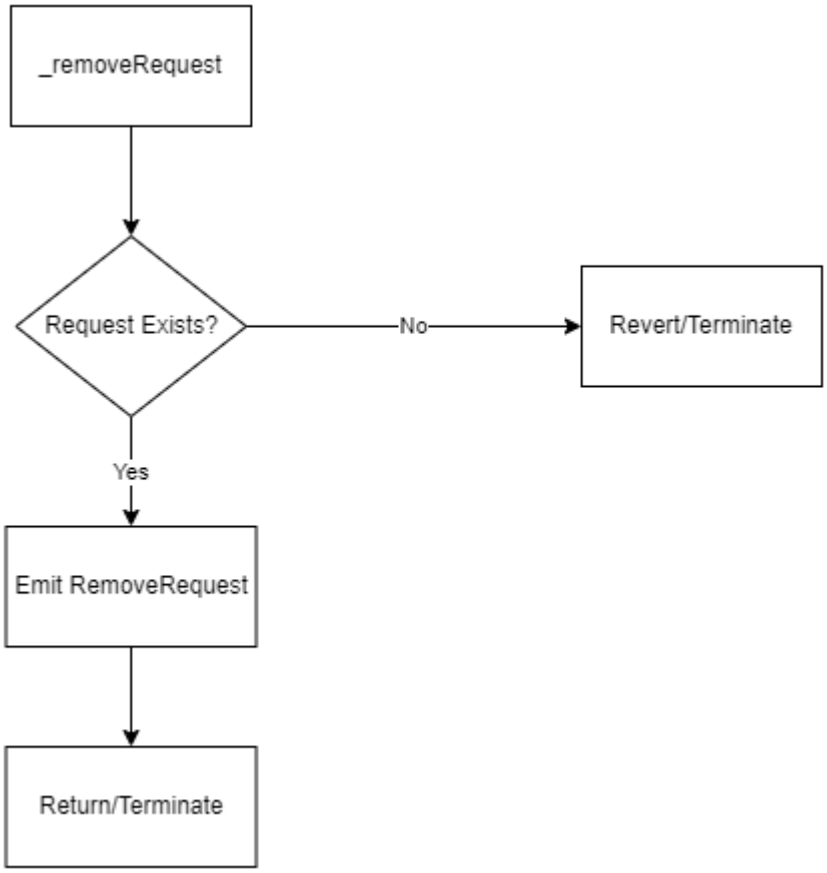

Figure 26: _removeRequest

4.2.9 getRequestExpiry

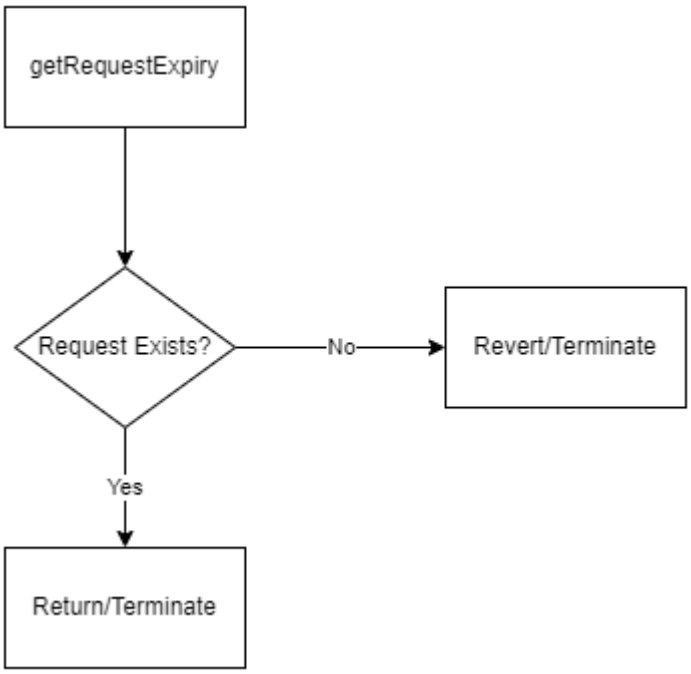

Figure 27: getRequestExpiry

References

[1] K. Shibano and G. Mogi, "An analysis of the acquisition of a monetary function by cryptocurrency using a multi-agent simulation model," *Financial Innovation*, vol. 8, no. 1, p. 87, 2022, ISSN: 2199-4730. DOI: 10.1186/s40854-022-00389-8",. [Online]. Available: https://doi.org/10.1186/s40854-022-00389-8.

[2] *14 companies that accept bitcoin & crypto 2023*, (Accessed on 05/14/2023). [Online]. Available: https://milkroad.com/accept-crypto.

[3] *Who accepts crypto? 250+ stores where you can spend bitcoin + crypto | bitpay*, (Accessed on 05/14/2023). [Online]. Available: https://bitpay.com/directory/.

[4] N. Gandal, T. Hamrick J. T.and Moore, and M. Vasek, "The rise and fall of cryptocurrency coins and tokens," *Decisions in Economics and Finance*, vol. 44, 981–1014, 2021.

[5] A. Feder, N. Gandal, J. Hamrick, T. Moore, and M. Vasek, "The rise and fall of cryptocurrencies," *Workshop on the Economics of Information Security*, [Online]. Available: https://par.nsf.gov/biblio/10066236.

[6] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): overview, evaluation, opportunities and challenges," *CoRR*, vol. abs/2105.07447, 2021. arXiv: 2105.07447. [Online]. Available: https://arxiv.org/abs/2105.07447.

[7] M. Nadini, L. Alessandretti, F. Di Giacinto, M. Martino, L. M. Aiello, and A. Baronchelli, "Mapping the nft revolution: Market trends, trade networks, and visual features," *Scientific Reports*, vol. 11, no. 1, p. 20 902, 2021, ISSN: 2045-2322. DOI: 10.1038/s41598-021-00053-8. [Online]. Available: https://doi.org/10.1038/s41598-021-00053-8.

[8] *What is ethereum? | ethereum.org*, (Accessed on 05/14/2023). [Online]. Available: https://ethereum.org/en/what-is-ethereum/.

[9] *Erc-721 non-fungible token standard | ethereum.org*, (Accessed on 05/14/2023). [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-721/#:~:text=What%20is%20a%20Non%2DFungible,concerts%20and%20sports%20matches%2C%20etc..

[10] *Erc-20 token standard | ethereum.org*, (Accessed on 05/14/2023). [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/.

[11] *Erc-20 token standard | ethereum.org*, (Accessed on 05/14/2023). [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/.

[12] *How to handle incorrect payment from a client [email templates]*, (Accessed on 05/14/2023). [Online]. Available: https://www.invoiceowl.com/blog/how-to-handle-incorrect-payment/.

[13] *What to do when a client pays the wrong amount: Free email template*, (Accessed on 05/14/2023). [Online]. Available: https://getjobber.com/academy/incorrect-payment-received-template/.

[14] *Erc | ethereum improvement proposals*, (Accessed on 05/24/2023). [Online]. Available: https://eips.ethereum.org/erc.

[15] *Home | ethereum improvement proposals*, (Accessed on 05/24/2023). [Online]. Available: https://eips.ethereum.org/.

[16] F. Vogelsteller and V. Buterin, *Erc-20: Token standard," ethereum improvement proposals, no. 20*, (Accessed on 05/24/2023), Nov. 2015. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20.

[17] M. Vladimirov and D. Khovratovich, *Erc20 api: An attack vector on approve/transferfrom methods*, (Accessed on 05/24/2023). [Online]. Available: https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit.

[18] *Erc: Token standard · issue #20 · ethereum/eips*, (Accessed on 05/24/2023). [Online]. Available: https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729.

[19] *Erc 20 - openzeppelin docs*, (Accessed on 05/24/2023). [Online]. Available: https://docs.openzeppelin.com/contracts/4.x/api/token/erc20#IERC20-approve-address-uint256-.

[20] A. Slagell and W. Yurcik, "Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization," in *Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, 2005.*, 2005, pp. 80–89. DOI: 10.1109/SECCMW.2005.1588299.

[21] D. Forte, "The importance of log files in security incident prevention," *Network Security*, vol. 2009, no. 7, pp. 18–20, 2009, ISSN: 1353-4858. DOI: https://doi.org/10.1016/S1353-4858(09)70090-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S135348580970090X.

[22] *Events | solidity by example | 0.8.17*, (Accessed on 05/24/2023). [Online]. Available: https://solidity-by-example.org/events/.

[23] *Learn solidity: What are events?* (Accessed on 05/24/2023). [Online]. Available: https://www.alchemy.com/overviews/solidity-events.

[24] N. J. Kipyegen and W. P. Korir, "Importance of software documentation," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 5, p. 223, 2013.

[25] V. S. Chomal and J. R. Saini, "Significance of software documentation in software development process," *International Journal of Engineering Innovations and Research*, vol. 3, no. 4, p. 410, 2014.

[26] *Natspec format — solidity 0.8.20 documentation*, (Accessed on 05/24/2023). [Online]. Available: https://docs.soliditylang.org/en/v0.8.20/natspec-format.html.

[27] T. R. Devi, "Importance of testing in software development life cycle," *International Journal of Scientific & Engineering Research*, vol. 3, no. 5, pp. 1–5, 2012.

[28] M. E. Khan and F. Khan, "Importance of software testing in software development life cycle," *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 2, p. 120, 2014.

[29] V. Buterin, *Ethereum: A next-generation smart contract and decentralized application platform.* 2014. [Online]. Available: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.

[30] *Ethereum security and scam prevention | ethereum.org*, (Accessed on 05/14/2023). [Online]. Available: https://ethereum.org/en/security/.

[31] *Ethereum : Security vulnerabilities*, (Accessed on 05/14/2023). [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-17524/Ethereum.html.