Ultrade CODEX

Ultrade EVM & Algorand Cross Chain Testing

Smart Contract Audit Report

DATE: 10th December 2024 | **PREPARED BY:** Entersoft Pty Ltd

Contents

Revision History & Version Control **3**
1. Executive Summary **3**
2. Overview **4**
2.1 Project Overview 4
2.2. Scope 4
2.3 Summary of Findings 4
2.4 Security Level References 4
3. Comprehensive Analysis findings:Manual Testing **5**
4. Tested For Scenarios **6**
4.1 Deposit EVM to Codex crosschain duplication 6
1. Deposit from EVM to Codex 6
2. Duplicate Deposit Attempt 6
4.2 Withdraw Codex to EVM crosschain duplication 9
1. Withdraw from Codex to EVM 9
2. Duplicate Withdrawal Attempt 10
4.3 Manipulating Chain Id in VAA during Deposit 12
4.4 Manipulating Amount in VAA during Deposit 18
4.5 Manipulating Address in VAA during Deposit 23
4.6 Manipulating Normalized Wallet Address in VAA during Withdraw 30
4.7 Manipulating String in VAA during Withdraw 41
4.8 Manipulating Amount in VAA during Withdraw 50
4.9 Manipulating ChainId in VAA during Withdraw 58
7. Auditing Approach and Methodologies Applied **67**
8. Limitations on Disclosure and Use of this Report **68**

Revision History & Version Control

Version	Date	Author(s)	Description
3.0	10-December-2024	Gurkirat singh, Tharun Betina	Final Report

1. Executive Summary

Entersoft has conducted a cross chain smart contract testing of the Ultrade EVM Algorand project through a comprehensive smart contract audit. The primary objective was to identify potential vulnerabilities and risks within the codebase, ensuring adherence to industry-leading standards while prioritizing security, reliability, and performance. Our focus was on prompt and efficient identification and resolution of vulnerabilities to enhance the overall robustness of the solidity smart contract.

Testing Methodology:

Our testing methodology in Solidity adhered to industry standards and best practices, integrating partially implemented OWASP and NIST SP 800 standards for encryption and signatures.

We have performed a detailed manual analysis, adherence to industry standards, and the use of a comprehensive toolset. Our approach ensured a thorough evaluation within the designated Solidity code files.

Findings and Security Posture:

Our audit process emphasized a comprehensive and meticulous approach to security analysis, prioritizing depth and precision throughout the evaluation. The audit was conducted from December 2, 2024, to December 10, 2024. During this timeframe, Entersoft thoroughly reassessed and validated the security posture of the Cross-Chain functionality for EVM and Algorand. Following this detailed evaluation, the system was designated as 'Secure,' signifying the absence of identified vulnerabilities and the robustness of the codebase in mitigating potential security threats.

2. Overview

2.1 Project Overview

Entersoft has meticulously tested cross chain transactions which involved EVM, Algorand and Wormhole from December 2, 2024, and December 10, 2024 with a primary focus on Solidity code files integral to blockchain functionality, emphasizing vulnerabilities in associated gas claiming. The working of basic functionalities was also tested during the review.

2.2. Scope

The audit scope covers the CODEX-dev smart contract available in the GitHub private repository:

Github	https://github.com/ultrade-org/CODEX
Branch	Dev Branch
Commit ID	e623676a0a6e8d69337019137d0f6dc1b8e4d1a2

OUT-OF-SCOPE: External contracts, other imported smart contracts.

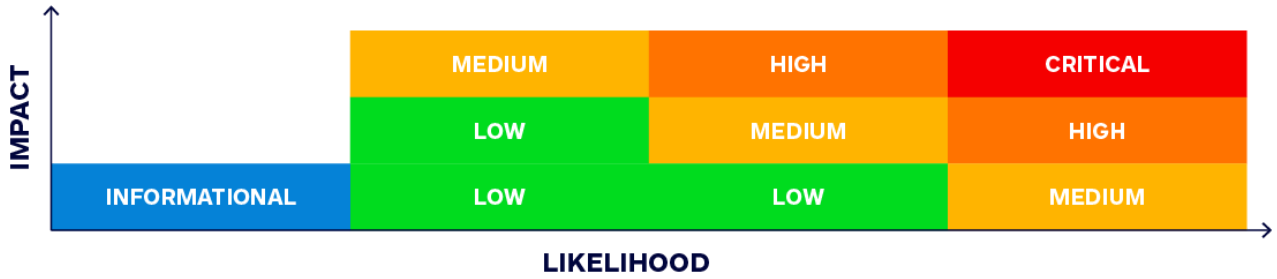
2.3 Summary of Findings

The following table is the summary of findings, which summarizes the overall risks identified during the audit.

Critical	High	Medium	Low	Informational
0	0	0	0	0

2.4 Security Level References

Every vulnerability in this report was assigned a severity level from the following classification table:



3. Comprehensive Analysis findings:Manual Testing

Phase 1: Dynamic Analysis

Dynamic analysis was conducted, which involved a more hands-on approach to scrutinizing the intended functionality and security of the cross chain transactions. Further, we devised a comprehensive suite of manual tests and manipulated raw VAA to validate the expected behavior of the smart contracts under various scenarios.

Processes and Test Cases:

Our dynamic analysis systematically explored the cross-chain functionalities of the protocol, concentrating on critical areas susceptible to vulnerabilities. During testing, when a deposit transaction occurred, the transaction hash was retrieved from Testnet Etherscan, and the raw VAA was obtained from Wormhole Scan. The VAA was decrypted to reveal its payload, which was then altered and processed on Algorand to observe whether the transaction succeeded or reverted. Similarly, for withdrawal transactions from Algorand to the EVM chain, the process was reversed to ensure comprehensive validation.

Throughout the dynamic analysis phases, our team dedicated substantial efforts to meticulously review the codebase, identify vulnerabilities, and develop robust manual test cases to assess the security posture of the protocol comprehensively.

4. Tested For Scenarios

4.1 Deposit EVM to Codex crosschain duplication

Result	Pass
Description	Testing Cross-Chain Deposit Using Wormhole VAA In this scenario, we are validating the behavior of cross-chain deposit transactions facilitated by the Wormhole protocol. The focus is on the manual verification of the VAA to ensure the prevention of duplicate transactions.
Function	Deposit

Steps Performed	1. Deposit from EVM to Codex
	
	● A transaction is initiated on the EVM network to deposit tokens into the Codex contract on Algorand.
	● Wormhole generates a VAA for this transaction.
	● The VAA is manually verified on Algorand by submitting it to the Codex contract.
	● The deposit is successful, as evidenced by:
	  ○ A change in the user's balance in Codex.
	  ○ Confirmation of the transaction ID: Q4JNULJITJWMTLLCDIBOUDODNIT34JKDDU3TMR5NDMQEYPTYA77A.
	  ○ VAA details include RAW VAA and PAYLOAD.
	
	2. Duplicate Deposit Attempt
	
	● The same RAW VAA is re-submitted to the Codex contract.
	● The transaction is rejected, as expected, with an error indicating duplication or a replay attack.
	● This demonstrates the system's ability to prevent double-spending or redundant transaction processing by verifying the uniqueness of the VAA.

Observation

Initial balance of user in codex 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d

```
● (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
  login chain id 10002
  login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
  login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
  assest Chain id 65537
  assest address 434354505553444300000000000000000000000000000000000000000000000
  b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x04\xe2
  \x00' 32
  userBalance 320000
```

```
● gurkiratsingh@Gurkirats-MacBook-Air evm % npm run depositToCodex

  > TokenManager@1.0.0 depositToCodex
  > ts-node ./ts-scripts/main.ts --depositToCodex

  chain {
    description: 'Ethereum Sepolia',
    chainId: 10002,
    rpc: 'https://ethereum-sepolia-rpc.publicnode.com',
    tokenBridge: '0xDB5492265f6038831E89f495670FF909aDe94bd9',
    wormholeRelayer: '0x7B1bD7a6b4E61c2a123AC6BC2cbfC614437D0470',
    wormhole: '0x4a8bc80Ed5a4067f1CCf107057b8270E0cC11A78',
    cctpTokenMessenger: '0x9f3B8679c73C2Fef8b59B4f3444d4e156fb70AA5',
    cctpMessageTransmitter: '0x7865fAfC2db2093669d92c0F33AeEF291086BEFD',
    USDC: '0x1c7D4B196Cb0C7B01d743Fbc6116a902379C7238',
    WETH: '0xfff9976782d46cc05630d1f6ebab18b2324d6b14',
    testToken: '0x3DdB199aFB250ee004aC0747b2954291fB66E56A'
  }
  token 0x1c7D4B196Cb0C7B01d743Fbc6116a902379C7238
  amount 100000n
  _proxy {
    '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
    '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
    '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
    '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
    '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
  }
  currentGasPrice 19506484071
```

[This is a Sepolia **Testnet** transaction only]

Transaction Hash:	0x57d4e039fd30145a0111e32210e909dce148134f13beef2fb0bc11b7412d7860
Status:	Success
Block:	7194780 1 Block Confirmation
Timestamp:	40 secs ago (Dec-02-2024 07:53:24 AM UTC)
Transaction Action:	Call Deposit To Codex Function by 0xf0bA633D...51784B44d on 0x98Ed3E65...2f1ed97b3
From:	0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d
Interacted With (To):	0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3
ERC-20 Tokens Transferred:	All Transfers Net Transfers From 0xf0bA633D...51784B44d To 0x98Ed3E65...2f1ed97b3 For 0.1 ERC-20: USDC (USDC)
Value:	0 ETH
Transaction Fee:	0.0022261471957696 ETH
Gas Price:	21.18364795 Gwei (0.00000002118364795 ETH)

WORMHOLE SCAN TXN:
https://wormholescan.io/#/tx/0x57d4e039fd30145a0111e32210e909dce148134f13beef2fb0bc11b7412d7860?network=Testnet

RAW Vaa:
AQAAAAABABtqgHsmfmTrBiAMFTeIwXZqMyjrHm/GZQYEqOz8yfayLpO2lDFGW97vYaPpc6xnghZBpSu1VE6KUvLCwkzF4SkBZ01ndAAAABInEgAAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAAABLIZGVwb3NpdAAAAAAAAAAAAAAAPC6Yz2Miy7+qLml/jtM45UXhLRNQ0NUUFVTREMAAAAAAAAAAAAAAA

Run Receive Message in Codex:

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBB
hJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VA
CVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4F
BCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c6380000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000677561726469616e
submitted_txns, submitted_accts [<algosdk.transaction.ApplicationCallTxn object at 0x10707b8c0>, <algosdk.transaction.Applicat
ionCallTxn object at 0x104e4f110>] ['JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34', 'Z7AISPXE4YFESMU35JXBEM6JZSK
TVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']
txns [<algosdk.transaction.ApplicationCallTxn object at 0x10707b8c0>, <algosdk.transaction.ApplicationCallTxn object at 0x104e
4f110>, <algosdk.transaction.ApplicationCallTxn object at 0x1032d3800>]
result {'confirmed-round': 46343296, 'pool-error': '', 'txn': {'lsig': {'l': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoC
iAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCD
UDNAQlCDUEQv+0Iok='}, 'txn': {'apaa': ['dmVyaWZ5U2lncw==', 'ABtqgHsmfmTrBiAMFTelwXZqMyjrHm/GZQYEqOz8yfayLpO2lDFGW97vYaPpc6xngh
ZBpSu1VE6KUvLCwkzF4SkB', 'E5R71IsY5T/a7ud/NHM5Gscnxjg=', '8d76l0FAJmW3tZ86UARPT0dC7oV0rrNardAgtAYa1fg=']', 'apat': ['JM7ZHEEG74
NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34', 'Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']', 'apid': 8652
5623, 'fv': 46343293, 'gen': 'testnet-v1.0', 'gh': 'SGO1GKSzyE7IEPItTxCByw9x8FmnrCDexi9/cOUJOiI=', 'grp': 'iQshxEqgF18+KMR322M
Vy1NHrECGI8egCOJpB7yL3bc=', 'lv': 46344293, 'snd': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'type': 'appl
'}}}
box 420000
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06h\xa
0' 32
userBalance 420000
```

<u>TXN id on codex part -</u>
Q4JNULJITJWMTLLCDIBOUDODNIT34JKDDU3TMR5NDMQEYPTYA77A

Q4JNULJIT...	46343296	AFIA...64JU	→	727096228	Application Call	7 minutes
		CH6M...N2CE	→	727096157	Application Call	7 minutes

Deposit successful

<u>Deposit with same raw vaa to see if the txn takes place getting below error:</u>

```
if __name__ == "__main__":
    # vaaString = "AQAAAAABADLN5o1cBeYIXb+vVY/C+rjUy5VBiGCca8pjYX+6rVoFe1S4RDzv4q4zIT4smkURmmju7ZEdJRQy6fHDdQCOQ/4BZz39hAAAABAnEgAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx
    # vaaString = "AQAAAAABADLN5o1cBeYIXb+vVY/C+rjUy5VBiGCca8pjYX+6rVoFe1S4RDzv4q4zIT4smkURmmju7ZEdJRQy6fHDdQCOQ/4BZz39hAAAABAnEgAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx

    # Below is the raw vaa a deposit from EVM
    verfiedVAA = "AQAAAAABABtqgHsmfmTrBiAMFTelwXZqMyjrHm/GZQYEqOz8yfayLpO2lDFGW97vYaPpc6xnghZBpSu1VE6KUvLCwkzF4SkBZ01ndAAAABInEgAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx7
    dbEntryID = 657679

    # print(b64decode(vaaString).hex(),"This is vaa raw data")
    # "6465706f73697400000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d434354505553444300000000000000000000000000000000000000000000000000000000000000000000

    # falseData = "0100000000010032cde68d5c05e6085dbfaf558fc2fab8d4cb954188609c6bca63617fbaad5a057b54b8443cefe2ae33213e2c9a45119a68eeed911d251432e9f1c375008e43fe01673d

    # print(b64encode(bytes.fromhex(falseData)),"This is false data encoded")
    receiveMessageByEnv(SupportedChains.Etherem, SupportedChains.Etherem, TokenType.USDC, verfiedVAA, False, dbEntryID)

    # AQAAAAABADLN5o1cBeYIXb+vVY/C+rjUy5VBiGCca8pjYX+6rVoFe1S4RDzv4q4zIT4smkURmmju7ZEdJRQy6fHDdQCOQ/4BZz39hAAAABAnEgAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAAA
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATRQXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKD
XxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXw
NAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c638000000000
```

- Transaction ID: BWIOW6HSUN5SP3WJ73U664YFHFJTULQ3AFVR7KCU4ZOQIE6Y5G4Q.

2. Duplicate Withdrawal Attempt

- The same withdrawal transaction ID and VAA are reused to attempt a duplicate withdrawal on the EVM network.
- This results in an error, confirming that the system prevents replay attacks or duplicate transaction executions.

Observation

<u>Withdraw Codex to EVM</u>

Run withdraw from codex to evm

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
10002 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
dest_private_key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
434354505553444300000000000000000000000000000000000000000000000000 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL
KVNGXKQ4OCKMU5Y
login 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 10002
236
10002
b'\x19Ethereum Signed Message:\n549{"Login_Address": "0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d", "Login_ChainId": "10002", "
Token_Address": "434354505553444300000000000000000000000000000000000000000000000000", "Receiver_Address": "0xf0ba633d8c8b2efea8b
988fe3b4ce3951784b44d", "Receiver_ChainId": "10002", "Amount": "10000", "Withdraw_Fee": "False"}AAAAAAAAAAAAAAAA8LpjPYyLLv6ouY
j+O0zjlReEtE0AAAAAAAAnEkNDVFBVU0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0AAAAAAAAnEgAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
userBalance 410000
after algo amount of codex 2000000
after algo amount of msg processor 2000000
```

Balance after withdraw

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06A\x9
0' 32
userBalance 410000
```

Run withdraw from codex txn hash:
BWIOW6HSUN5SP3WJ73U664YFHFJTULQ3AFVR7KCU4ZOQIE6Y5G4Q

BWIOW6HSU... inner	46346920	AMSO...TEKM →	727096228	Application Call	3 minutes

Raw Vaa from codex withdraw:
AQAAAAABAPLmCmlFcrAjpzGO4DpTfHgp8cm0rNd0Lv/P7UjBLFxFYFRK5zc3RKr8MgdUHrwGot2NyD9rJ4yH9G7iXi66OK0BZ02QsAAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAACIAd2l0aGRyYXcAAAAAAAAAAAAAADwumM9jlsu/qi5iP47TOOVF4S0TQAAAAAAACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcQAA2Q63jyo3sn7sn+6e9zBTlTOi4bAWsfqFTmXQQT2Om5AAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

Txn Id as input to receiveEncodedMsg:
BWIOW6HSUN5SP3WJ73U664YFHFJTULQ3AFVR7KCU4ZOQIE6Y5G4Q

```
gurkiratsingh@Gurkirats-MacBook-Air evm % npm run receiveEncodedMsg

> TokenManager@1.0.0 receiveEncodedMsg
> ts-node ./ts-scripts/main.ts --receiveEncodedMsg

_proxy {
  '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
  '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10002': '0x98Ed3E6543F5779D210135C319ff8dc2f1ed97b3',
  '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
  '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
  '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
}
Uint8Array(309) [
    1,   0,   0,   0,   0,   1,   0, 242, 230,  10, 105,  69,
  114, 176,  35, 167,  49, 142, 224,  58,  83, 124, 120,  41,
  241, 201, 180, 172, 215, 116,  46, 255, 207, 237,  72, 193,
   44,  92,  69,  96,  84,  74, 231,  55,  55,  68, 170, 252,
   50,   7,  84,  30, 188,   6, 162, 221, 141, 200,  63, 107,
   39, 140, 135, 244, 110, 226,  94,  46, 186,  56, 173,   1,
  103,  77, 144, 176,   0,   0,   0,   0,   0,   8,  17, 252,
  194, 187, 219, 179, 156, 225, 213,   0, 172,   6, 139,  35,
   78, 204, 242,  27,
  ... 209 more items
] This is raw encoded message
01000000000100f2e60a694572b023a7318ee03a537c7829f1c9b4acd7742effcfed48c12c5c4560544ae7373744aafc3207541ebc06a2dd8dc83f6b278c87
f46ee25e2eba38ad01674d90b000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000022007769746
8647261770000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000002712434354505553444300000000000000000000
0000000000000000000000000000000000000000000000000000271200000000000000000000000000000000000000000000000000000
```

```
      to: '0x98Ed3E6543F5779D210135C319ff8dc2f1ed97b3',
      data: '0x496d76ff0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000135010000000000100f2e60a694572b023a7318ee03a537c7829f1c9b4acd7742effcfed48c12c5c4560544ae7373744aafc32075
41ebc06a2dd8dc83f6b278c87f46ee25e2eba38ad01674d90b000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08
b000000000000000220077697468647261770000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000000271243435450555
3444300000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0002710000d90eb78f2a37b27eec9fee9ef730539533a2e1b016b1fa854e65d0413d8e9b900000000000000000000000000098ed3e6543f5779d210135c319ff8
dc2f1ed97b3000000000000000000000000',
      type: 2,
      accessList: null
    },
    error: Error: processing response error (body="{\"jsonrpc\":\"2.0\",\"id\":49,\"error\":{\"code\":3,\"message\":\"executio
n reverted: Message already processed\",\"data\":\"0x08c379a0000000000000000000000000000000000000000000000000000000000000000200
0000000000000000000000000000000000000000000000000000000000000194d65737361676520616c72656164792070726f636573736564000000000000000
0\"}}\n", error={"code":3,"data":"0x08c379a0000000000000000000000000000000000000000000000000000000000000000200000000000000000000
0000000000000000000000000000000000000000000000000000000000000194d65737361676520616c72656164792070726f636573736564000000000000000"}, requestBody=
"{\"method\":\"eth_estimateGas\",\"params\":[{\"type\":\"0x2\",\"maxFeePerGas\":\"0x4b0d23612\",\"maxPriorityFeePerGas\":\"0x5
9682f00\",\"from\":\"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d\",\"to\":\"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3\",\"data
\":\"0x496d76ff0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000
0000000000000135010000000000100f2e60a694572b023a7318ee03a537c7829f1c9b4acd7742effcfed48c12c5c4560544ae7373744aafc3207541ebc06a
2dd8dc83f6b278c87f46ee25e2eba38ad01674d90b000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b00000000
0000000220077697468647261770000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000000271243435450555344430000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000027100
00d90eb78f2a37b27eec9fee9ef730539533a2e1b016b1fa854e65d0413d8e9b900000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed9
7b3000000000000000000000000\"}],\"id\":49,\"jsonrpc\":\"2.0\"}", requestMethod="POST", url="https://ethereum-sepolia-rpc.publi
cnode.com", code=SERVER_ERROR, version=web/5.7.1)
        at Logger.makeError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/c
ontracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:269:28)
        at Logger.throwError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/
contracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:281:20)
        at /Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/chains/ev
m/node_modules/@ethersproject/web/src.ts/index.ts:341:28
        at step (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/cha
ins/evm/node_modules/@ethersproject/web/lib/index.js:33:23)
        at Object.next (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contra
cts/chains/evm/node_modules/@ethersproject/web/lib/index.js:14:53)
        at fulfilled (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contract
s/chains/evm/node_modules/@ethersproject/web/lib/index.js:5:58)
        at processTicksAndRejections (node:internal/process/task_queues:95:5) {
      reason: 'processing response error',
      code: 'SERVER_ERROR',
      body: '{"jsonrpc":"2.0","id":49,"error":{"code":3,"message":"execution reverted: Message already processed","data":"0x08
c379a0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000
000000194d65737361676520616c72656164792070726f636573736564000000000000000"}}\n',
      error: [Error],
      requestBody: '{"method":"eth_estimateGas","params":[{"type":"0x2","maxFeePerGas":"0x4b0d23612","maxPriorityFeePerGas":"0
x59682f00","from":"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d","to":"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3","data":"0x496
d76ff0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000
0000135010000000000100f2e60a694572b023a7318ee03a537c7829f1c9b4acd7742effcfed48c12c5c4560544ae7373744aafc3207541ebc06a2dd8dc83f6
b278c87f46ee25e2eba38ad01674d90b000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b00000000000000220
077697468647261770000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000000271243435450555344430000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000027
12000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
2a37b27eec9fee9ef730539533a2e1b016b1fa854e65d0413d8e9b900000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b30000000
0000000000000000000"}],"id":49,"jsonrpc":"2.0"}',
      requestMethod: 'POST',
      url: 'https://ethereum-sepolia-rpc.publicnode.com'
    }
```

4.3 Manipulating Chain Id in VAA during Deposit

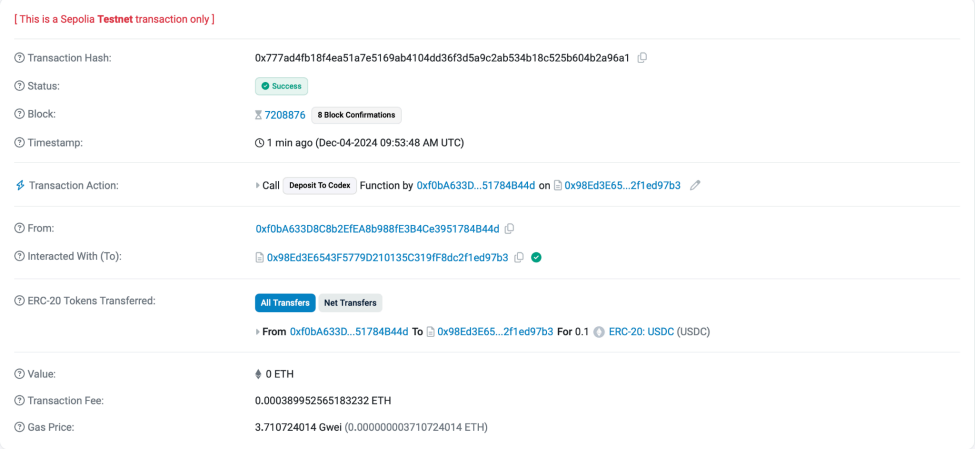
Result	Pass
Description	This process tests the robustness of a cross-chain token deposit mechanism, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect chainId) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Deposit
Steps Performed	1. **Check Balance Check:** • Verify initial token balances of the user and the contract. 2. **Deposit Function Execution:** • Execute the deposit function on the EVM chain. • Capture the transaction hash for further analysis. 3. **Correct VAA Collection:** • Retrieve the original VAA from Wormhole scan for the deposit transaction. 4. **Decrypt the Correct VAA:** • Decode the raw VAA into a readable format for inspection. • Decoded Data: • **Amount:** 0x000000000000000000000000000186a0 • **Chain ID:** 0x2712 • **Recipient Address:** 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d

5. **Manipulate VAA:**
   - Modify the chainId in the payload from the correct value (e.g., 10002) to an invalid value (e.g., 10004).
   - Manipulating the amount 0x2712 to 0x2714 which is 10002 to 10004.
   - Reconstruct the VAA with the modified payload.
6. **Encrypt and Test Altered VAA:**
   - Encrypt the manipulated VAA and submit it to the Algorand network manually for verification.
   - Confirm that the transaction is rejected, ensuring no balance update occurs.
7. **Re-test with Correct VAA:**
   - Re-submit the original, unaltered VAA to verify the deposit transaction proceeds as intended.
   - Check if the user's balance is updated correctly.
8. **Final Validation:**
   - Ensure the manipulated VAA fails and does not alter balances.
   - Confirm the correct VAA succeeds, incrementing the user's balance.

Observation

Step -1 Check for the user balance

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000000
b'\x00\x00\x00
```



Step -3 Collecting the correct vaa from the wormhole scan

Wormhole scan of deposit txn
https://wormholescan.io/#/tx/0x777ad4fb18f4ea51a7e5169ab4104dd36f3d5a9c2ab534b18c525b604b2a96a1?network=Testnet

Raw Vaa
AQAAAAABACjIYkW2c0v2UnBjY2mCcsMzmqBHmAzmbdCSRnU+ri1ITLhU+OORFSAkaU6NzaiKOCQ/6HLpi6K4JsmXbIOGSzAAZ1AmrAAAABUnEgAAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAAABXIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ0NUUFVTREMAAAAAAAAAAAAAAAA

Token address==434354505553444300000000000000000000000000000000000000
0000000000000== CCTPUSDC

amount==0000000000000000000000000000000000000000000000000000000000000186a0==
100000

loginchainid==0000000000000000000000000000000000000000000000000000000000000271
2 == 10002

Loginaddress==000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce
3951784b44d

Step -5 Changing the decrypted raw vaa into wrong vaa

Changing the chain id into wrong chain id in payload

Changing the login chain id to == 10004 == 2714
from == 2712 == 10002

<u>Correct vaa</u>
0100000000010028c86245b6734bf652706363698272c3339aa047980ce66dd09246753eae2d4
84cb854f8e391152024694e8dcda88a38243fe872e98ba2b826c9976e53864b3000675026ac000
00015271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b3000000
0000000015c86465706f736974000000000000000000000000f0ba633d8c8b2efea8b988fe3b4
ce3951784b44d434354505553444300000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000186a00000000
000000000000000000000000000000000000000000000000000000000027120000000000000000
0000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d000000000000000000000000000000
000000000000000000000000000000000186a000000000000000000000000000000000000000000
000000000000000000000027120000000000000000000000000f0ba633d8c8b2efea8b988fe3b4
ce3951784b44d

<u>Result after decoding</u>

Argument Types
uint256,uint256,bytes32
Add the value types, each seperated by a comma
Encoded data
000000000000000000000000000000000000000000000000000000000000186a000000000000000000000000000000000000000000000000000000000

0100000000010028c86245b6734bf652706363698272c3339aa047980ce66dd09246753eae2d4
84cb854f8e391152024694e8dcda88a38243fe872e98ba2b826c9976e53864b3000675026ac000
00015271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b3000000
0000000015c86465706f7369740000000000000000000000000f0ba633d8c8b2efea8b988fe3b4
ce3951784b44d434354505553444300000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000186a00000000000
000000000000000000000000000000000000000000000000000000000027140000000000000000000
0000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d

Step -6 Encrypting the wrong vaa with changed chain id

Encrypting wrong vaa or false data

```
falseData =
"0100000000010028c86245b6734bf652706363698272c3339aa047980ce66dd09246753eae2d484cb854f8e391152024694e8dcda88a38243fe
872e98ba2b826c9976e53864b3000675026ac000000015271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b3000
0000000000015c86465706f7369740000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d434354505553444300000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000186a000000000000
000000000000000000000000000000000000000000000000000000000027140000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44
d"

print(b64encode(bytes.fromhex(falseData)),"This is false data encoded")
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
b'AQAAAAABACjIYkW2c0v2UnBjY2mCcsMzmqBHmAzmbdCSRnU+ri1ITLhU+OORFSAkaU6NzaiKOCQ/6HLpi6K4JsmXblOGSzAAZ1AmrAAAABUnEgAAAAAAAAAAAAAA
AJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAAABXIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGGoAAAAAAAAAAAAAAAAAAA
```

Step -8 Final balance of the user must remain the same as before

Final Balance of user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is not updated as expected

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000

b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\tN\xd0'
 32
userBalance 610000
```

Step -9 Entering the correct raw vaa to see if the balance gets updated

Correct Vaa

AQAAAAABACjIYkW2c0v2UnBjY2mCcsMzmqBHmAzmbdCSRnU+ri1ITLhU+OORFSAkaU6NzaiKOCQ/6HLpi6K4JsmXblOGSzAAZ1AmrAAAABUnEgAAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAAABXIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ0NUUFVTREMAAAAAAAAAAAAAAAA

4.4 Manipulating Amount in VAA during Deposit

Result	Pass
Description	This process tests the robustness of a cross-chain token deposit mechanism, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect amount) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Deposit
Steps Performed	1. **Check Initial Balances:** ● Verify the balances of the involved user accounts before initiating the deposit process. 2. **Deposit Function Execution:** ● Execute the deposit transaction on the EVM side. ● Collect the resulting VAA for further testing. 3. **Correct VAA Collection:** ● Retrieve the generated VAA from Wormhole scan. 4. **Decrypt the Correct VAA:** ● Decode the raw VAA into a readable format for inspection. ● Decoded Data: ● **Amount:** 0x00000000000000000000000000186a0 ● **Chain ID:** 0x2712 ● **Recipient Address:** 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 5. **Manipulate VAA:** ● Create a malicious VAA by altering the decrypted data. ● Manipulating the amount 00000000000000000000000000000000000000000000000000000000000186a0 to 0000000000000000000000000000000000000000000000000000000000030d40 which is from 100000 to 200000. 6. **Encrypt and Test Altered VAA:** ● Encode the manipulated data back into a raw VAA. ● Feed the manipulated VAA into the deposit processing mechanism. ● Observe and confirm that the system correctly rejects the altered VAA, ensuring the user's balance remains unaffected. 7. **Re-test with Correct VAA:** ● Re-submit the original, unaltered VAA to verify the deposit transaction proceeds as intended. ● Confirm that the transaction succeeds and the user's balance updates as expected. 8. **Final Validation:** ● Ensure the manipulated VAA fails and does not alter balances. ● Confirm the correct VAA succeeds, incrementing the user's balance.

Observation

Step -1 Check for both the balances

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000
b'\x00\x00\x00
```

Correct Vaa

AQAAAAABAMAGZxAjdyG0WEYUO9ufDMpy9aG/9IgNN5WSuefnvw6lapKMOeV06xZY+7LzOo5SH
ulN1Mm50krfSRvli7PcQScBZ1A/VAAAABYnEgAAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx7Z
ezAAAAAAAAABbIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ0NUUF
VTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAGGoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcSAAAAAAAAAAAA
AAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0=

Decrypted raw vaa

01000000000100c0066710237721b45846143bdb9f0cca72f5a1bff4880d379592b9e7e7bf0ea56a
928c39e574eb1658fbb2f33a8e521ee94dd4c9b9d24adf491be58bb3dc41270167503f5400000016
2712000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b30000000000
000016c86465706f7369740000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce39
51784b44d4343545055534443000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000186a000000000000
000000000000000000000000000000000000000000000000000271200000000000000000000000
000f0ba633d8c8b2efeA8b988fe3b4ce3951784b44d

Payload-

6465706f7369740000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44
d4343545055534443000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000186a000000000000000000000
000000000000000000000000000000000000000000000000000271200000000000000000000000000f0ba63
3D8c8b2efea8b988fe3b4ce3951784b44d

Step -5 Changing the decrypted raw vaa into wrong vaa

1. Scenario — (Changing the chain id into wrong chain id in payload)

Changing the amount to == 200000

Correct vaa

01000000000100c0066710237721b45846143bdb9f0cca72f5a1bff4880d379592b9e7e7bf0ea56a
928c39e574eb1658fbb2f33a8e521ee94dd4c9b9d24adf491be58bb3dc41270167503f5400000016
2712000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b30000000000
000016c86465706f7369740000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce39
51784b44d4343545055534443000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000186a000000000000
000000000000000000000000000000000000000000000000000271200000000000000000000000
0f0ba633d8c8b2efea8b988fe3b4ce3951784b44d000000000000000000000000000000000000
000000000000000000000000000186a000000000000000000000000000000000000000000000000
000000000000000000271200000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce39517
84b44d

Decoded result

Argument Types
uint256,uint256,bytes32
Add the value types, each seperated by a comma
Encoded data
000000000000000000000000000000000000000000000000000000000000186a000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000000000000000002712

Login address==000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce
3951784b44d

Changing the amount to == 200000

amount==0000000000000000000000000000000000000000000000000000000000030d40

Wrong Vaa with changed chain id
01000000000100c0066710237721b45846143bdb9f0cca72f5a1bff4880d379592b9e7e7bf0ea56a
928c39e574eb1658fbb2f33a8e521ee94dd4c9b9d24adf491be58bb3dc41270167503f5400000016
2712000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b30000000000
000016c86465706f736974000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce39
51784b44d434354505553444300000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000030d4000000000000
000000000000000000000000000000000000000000000000000027120000000000000000000000
000f0ba633d8c8b2efEa8b988fe3b4ce3951784b44d

Step -6 Encrypting the wrong vaa with changed chain id

Encrypting wrong vaa or false data

```
falseData =
"01000000000100c0066710237721b45846143bdb9f0cca72f5a1bff4880d379592b9e7e7bf0ea56a928c39e574eb1658fbb2f33a8e521ee94dd
4c9b9d24adf491be58bb3dc41270167503f5400000016271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b3000
0000000000016c86465706f736974000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d434354505553444300000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000030d4000000000000
000000000000000000000000000000000000000000000000000027120000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44
d"

print(b64encode(bytes.fromhex(falseData)),"This is false data encoded")
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
b'AQAAAAABAMAGZxAjdyG0WEYUO9ufDMpy9aG/9IgNN5WSuefnvw6lapKMOeV06xZY+7LzOo5SHulN1Mm50krfSRvli7PcQScBZ1A/VAAAABYnEgAAAAAAAAAAAAAA
AJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAAABbIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcSAAAAAAAAAAAAAAAA8LpjPYyLLv6ouYj+
O0zjlReEtE0=' This is false data encoded
```

Wrong raw vaa
AQAAAAABAMAGZxAjdyG0WEYUO9ufDMpy9aG/9IgNN5WSuefnvw6lapKMOeV06xZY+7LzOo5SH
ulN1Mm50krfSRvli7PcQScBZ1A/VAAAABYnEgAAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/jcLx7Z
ezAAAAAAAAABbIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ0NUUF
VTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAMNQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcSAAAAAAAAAAAAAAAA
AAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0=

Step -7 Feeding the wrong raw vaa to the runReceiveMessage.py

Error Raised as expected

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7PSIQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENho8NhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQ8NAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA1BTXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQ1CDUEQv+0Iok=
MiCCRYNAA0A4EhCCRYBwA1BTXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQ1CDUEQv+0Iok=}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENho8NhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQ8NAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA1BTXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQ1CDUEQv+0Iok=
sig_addr 72706228 JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRD7KAKO2U6HZVZ25IQOG3WMHN34
sig_addr 86525623 Z7AISPXE4YFE5MU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c638000000000
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBB
hJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VA
CVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4F
BCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c63800000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000006775617264696 16e
submitted_txns, submitted_accts [<algosdk.transaction.ApplicationCallTxn object at 0x109f53c20>, <algosdk.transaction.Applicat
ionCallTxn object at 0x10724dac0>] ['JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34', 'Z7AISPXE4YFESMU35JXBEM6JZSK
TVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']
txns [<algosdk.transaction.ApplicationCallTxn object at 0x109f53c20>, <algosdk.transaction.ApplicationCallTxn object at 0x1072
4dac0>, <algosdk.transaction.ApplicationCallTxn object at 0x109c99880>]
result {'confirmed-round': 46411583, 'pool-error': '', 'txn': {'lsig': {'l': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoC
iAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCD
UDNAQlCDUEQv+0Iok='}, 'txn': {'apaa': ['dmVyaWZ5U2lncw==', 'AMAGZxAjdyG0WEYUO9ufDMpy9aG/9IgNN5WSuefnvw6lapKMOeV06xZY+7LzOo5SHu
lN1Mm50krfSRvli7PcQScB', 'E5R71IsY5T/a7ud/NHM5Gscnxjg=', 'JA2eRY49zpwy3YlOxFraeFoTtmk9Q9Pswx9aQJ1EZMA=']', 'apat': ['JM7ZHEEG74
NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34', 'Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']', 'apid': 8652
5623, 'fv': 46411581, 'gen': 'testnet-v1.0', 'gh': 'SGO1GKSzyE7IEPItTxCByw9x8FmnrCDexi9/cOUJOiI=', 'grp': 'yPxxCakVyZM/aMqTMoP
6FxkOoRtrxYEDI0nUdVdnlno=', 'lv': 46412581, 'snd': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'type': 'appl
'}}}
box 810000
```

Final balance of the user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is updated and incremented

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0c\\\x
10' 32
userBalance 810000
```

txn id on algorand—
EWYFJWPDCZCJ5GTFGIILW3OORTXFXELJGEEB342B5ACJSZK75MTA

EWYFJWPDC...	46411583	AFIA...64JU →	727096228	Application Call	1 minute
		CH6M...N2CE →	727096157	Application Call	1 minute

4.5 Manipulating Address in VAA during Deposit

Result	Pass
Description	This process tests the robustness of a cross-chain token deposit mechanism, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect address) can bypass verification and affect user balances, ensuring system resilience against such exploits. The tests focus on two scenarios involving VAA payload manipulation: 1. **Scenario 1:** Modifying the login address in the VAA payload. 2. **Scenario 2:** Modifying both the msg.sender and the login address in the VAA payload. In both scenarios, the aim is to determine if the changes allow the system to process a transaction that updates the balance of an unintended user or rejects it as expected.
Function	Deposit
Steps Performed	1. **Check Initial Balances:** • **User 1**: 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d (Correct login address). • **User 2**: 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d (Incorrect address).

2. **Deposit Function Execution:**
   - Execute the deposit transaction on the EVM side.
   - Collect the resulting VAA for further testing.
3. **Correct VAA Collection:**
   - Retrieve the generated VAA from Wormhole scan.
4. **Decrypt the Correct VAA:**
   - Decode the raw VAA into a readable format for inspection.
   - Decoded Data:
     - **Amount:** 0x00000000000000000000000000000186a0
     - **Chain ID:** 0x2712
     - **Recipient Address:** 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
5. **Manipulate VAA:**
   - **Scenario 1**: Replace the login address in the payload with a wrong address:
     - **New Address**: 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d.
   - **Scenario 2**: Replace both the login address and msg.sender with the wrong address.
6. **Encrypt and Test Altered VAA:**
   - Re-encrypt the manipulated VAA using the provided encryption script.
7. **Re-test with Correct VAA:**
   - Re-submit the original, unaltered VAA to verify the deposit transaction proceeds as intended.
   - Confirm that the transaction succeeds and the user's balance updates as expected.
8. **Final Validation:**
   - Ensure the manipulated VAA fails and does not alter balances of both user1 and user2.
   - Confirm the correct VAA succeeds, incrementing the user's 1 balance.

Observation

Step -1 Check for both the balances

Bal of 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d

```
● (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
  login chain id 10002
  login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
  login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
  assest Chain id 65537
  assest address 434354505553444300000000000000000000000000000000000000000000000
  b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06A\x9
  0' 32
  userBalance 410000
```

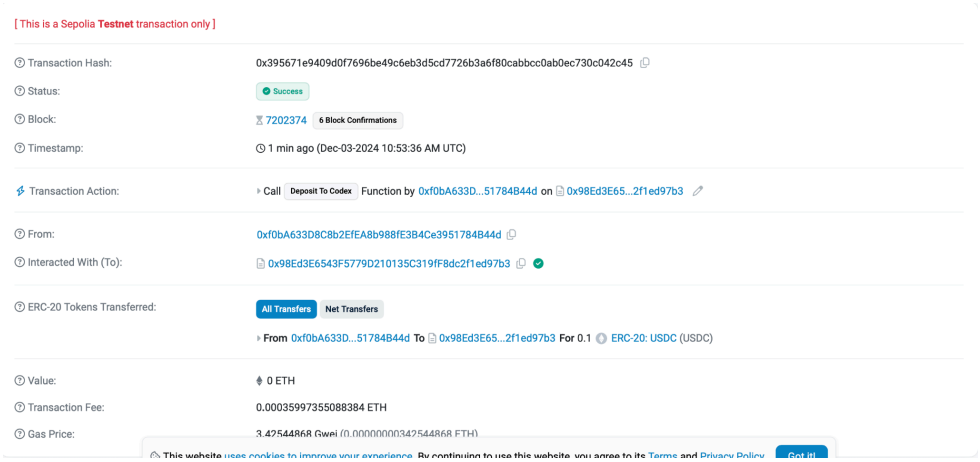
Bal of 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d

```
● (venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
  login chain id 10002
  login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
  login address 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d
  assest Chain id 65537
  assest address 434354505553444300000000000000000000000000000000000000000000000
  b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xa1
  ' 32
  userBalance 500000
```

Step -2 Running the deposit function

```
● gurkiratsingh@Gurkirats-MacBook-Air evm % npm run depositToCodex

  > TokenManager@1.0.0 depositToCodex
  > ts-node ./ts-scripts/main.ts --depositToCodex

  chain {
    description: 'Ethereum Sepolia',
    chainId: 10002,
    rpc: 'https://ethereum-sepolia-rpc.publicnode.com',
    tokenBridge: '0xDB5492265f6038831E89f495670FF909aDe94bd9',
    wormholeRelayer: '0x7B1bD7a6b4E61c2a123AC6BC2cbfC614437D0470',
    wormhole: '0x4a8bc80Ed5a4067f1CCf107057b8270E0cC11A78',
    cctpTokenMessenger: '0x9f3B8679c73C2Fef8b59B4f3444d4e156fb70AA5',
    cctpMessageTransmitter: '0x7865fAfC2db2093669d92c0F33AeEF291086BEFD',
    USDC: '0x1c7D4B196Cb0C7B01d743Fbc6116a902379C7238',
    WETH: '0xfff9976782d46cc05630d1f6ebab18b2324d6b14',
    testToken: '0x3DdB199aFB250ee004aC0747b2954291fB66E56A'
  }
  token 0x1c7D4B196Cb0C7B01d743Fbc6116a902379C7238
  amount 100000n
  _proxy {
    '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
    '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
    '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
    '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
    '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
  }
  currentGasPrice 2086218861
```

Txn hash —
0x395671e9409d0f7696be49c6eb3d5cd7726b3a6f80cabbcc0ab0ec730c042c45



Step -3 Collecting correct vaa from wormhole scan

Wormhole scan of deposit txn
https://wormholescan.io/#/tx/0x395671e9409d0f7696be49c6eb3d5cd7726b3a6f80cabbcc0ab0ec730c042c45?network=Testnet

Raw Vaa

AQAAAAABANiSEVJFcwHY1Zc3qDWGfHSsV73ILYo5WAQL0/CyI4hTTUjFW9SeG0CCN2eeEwN6H
N9FyO4NZYDkPNW6+QdSXRUAZ07jMAAAABMnEgAAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/j
cLx7ZezAAAAAAAAABPIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ
0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAGGoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcSAAAAA
AAAAAAAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0=

Step -4 Decrypting correct raw Vaa

Correct Vaa

AQAAAAABANiSEVJFcwHY1Zc3qDWGfHSsV73ILYo5WAQL0/CyI4hTTUjFW9SeG0CCN2eeEwN6H
N9FyO4NZYDkPNW6+QdSXRUAZ07jMAAAABMnEgAAAAAAAAAAAAAAAJjtPmVD9XedIQE1wxn/j
cLx7ZezAAAAAAAAABPIZGVwb3NpdAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ
0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAGGoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcSAAAAA
AAAAAAAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0=

Decrypted raw vaa

01000000000100d8921152457301d8d59737a835867c74ac57bdc82d8a3958040bd3f0b2238853
4d48c55bd49e1b408237679e13037a1cdf45c8ee0d6580e43cd5baf907525d1500674ee33000000
013271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300000000
00000013c86465706f736974000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce
3951784b44d434354505553444300000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000186a0000000000
000000000000000000000000000000000000000000000000000000271200000000000000000000
00000f0ba633D8c8b2efea8b988fe3b4ce3951784b44d

Payload-

6465706f736974000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44
d434354505553444300000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000186a000000000000000000
000000000000000000000000000000000000000000000000000000271200000000000000000000000f0ba63
3D8c8b2efea8b988fe3b4ce3951784b44d

Deposit=6465706f736974

Msg.sender==000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d
== 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d

Token address==434354505553444300000000000000000000000000000000000000
0000000000000== CCTPUSDC

amount==000000000000000000000000000000000000000000000000000000000000000186a0 ==
100000

loginchainid==0000000000000000000000000000000000000000000000000000000000000000271
2 == 10002

Loginaddress==000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d

1. Scenario — (Changing the address into wrong login address in payload)

Step -5 Changing the decrypted raw vaa into wrong vaa in decrypted vaa

Wrong address —

000000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d

replacing above address instead of below address in payload

000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d

Wrong vaa decrypted

01000000000100d8921152457301d8d59737a835867c74ac57bdc82d8a3958040bd3f0b2238853

4d48c55bd49e1b408237679e13037a1cdf45c8ee0d6580e43cd5baf907525d1500674ee33000000
0132712000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300000000
00000013c86465706f736974000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce
3951784b44d434354505553444300000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000186a0000000000
000000000000000000000000000000000000000000000000000000000000271200000000000000000000
000006f7ccf76681Da80cbd7ad4b28697a2881a70286d

Step -6 Encrypting the wrong vaa with changed login address

Encrypting wrong vaa or false data

```
falseData =
"01000000000100d8921152457301d8d59737a835867c74ac57bdc82d8a3958040bd3f0b22388534d48c55bd49e1b408237679e13037a1cdf45c
8ee0d6580e43cd5baf907525d1500674ee33000000013271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b3000
000000000013c86465706f736974000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d434354505553444300000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000186a000000000000
000000000000000000000000000000000000000000000000000000000000271200000000000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286
d"

print(b64encode(bytes.fromhex(falseData)),"This is false data encoded")
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
b'AQAAAAABANiSEVJFcwHY1Zc3qDWGfHSsV73ILYo5WAQL0/CyI4hTTUjFW9SeG0CCN2eeEwN6HN9FyO4NZYDkPNW6+QdSXRUAZ07jMAAAABMnEgAAAAAAAAAAAAAA
AJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAABPIZGVwb3NpdAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGGoAAAAAAAAAAAAAAAAAAA
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06A\x9
0' 32
userBalance 410000
```

Final Balance of user 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d is not updated as expected

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xa1
' 32
userBalance 500000
```

<u>2. Scenario — (Changing the address into wrong login address in both the places in payload in msg.sender and login address)</u>

Step -5 Changing the decrypted raw vaa into wrong vaa changing the login address in both payload and msg.sender and login address

<u>Wrong vaa with both address msg.sender and login address changed</u>

01000000000100d8921152457301d8d59737a835867c74ac57bdc82d8a3958040bd3f0b2238853
4d48c55bd49e1b408237679e13037a1cdf45c8ee0d6580e43cd5baf907525d1500674ee33000000
013271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300000000
00000013c86465706f7369740000000000000000000000006f7ccf76681da80cbd7ad4b28697a2
881a70286d434354505553444300000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000186a00000000000
000000000000000000000000000000000000000000000000000027120000000000000000000000
00006f7ccf76681Da80cbd7ad4b28697a2881a70286d

Step -6 Encrypting the wrong vaa with changed login address and Msg.sender

```
falseData =
"01000000000100d8921152457301d8d59737a835867c74ac57bdc82d8a3958040bd3f0b22388534d48c55bd49e1b408237679e13037a1cdf45c
8ee0d6580e43cd5baf907525d1500674ee33000000013271200000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b3000
0000000000013c86465706f7369740000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d434354505553444300000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000186a000000000000
000000000000000000000000000000000000000000000000000027120000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286
d"

print(b64encode(bytes.fromhex(falseData)),"This is false data encoded")
```

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
b'AQAAAAABANiSEVJFcwHY1Zc3qDWGfHSsV73ILYo5WAQL0/CyI4hTTUjFW9SeG0CCN2eeEwN6HN9FyO4NZYDkPNW6+QdSXRUAZ07jMAAAABMnEgAAAAAAAAAAAAAA
AJjtPmVD9XedIQE1wxn/jcLx7ZezAAAAAAAAABPIZGVwb3NpdAAAAAAAAAAAAAAAAG98z3ZoHagMvXrUsoaXoogacChtQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
```

Step -8 Final bal of the both the users see if the balance gets uploaded

Balance after both the scenario:

Final Balance of user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is not updated as expected

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06A\x9
0' 32
userBalance 410000
```

Final Balance of user 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d is not updated as expected

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d
assest Chain id 65537
assest address 434354505553444300000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xa1
' 32
userBalance 500000
```

Step -9 Entering the correct raw vaa to see if the balance gets updated

<u>Correct Vaa</u>

AQAAAAABANiSEVJFcwHY1Zc3qDWGfHSsV73ILYo5WAQL0/CyI4hTTUjFW9SeG0CCN2eeEwN6H
N9FyO4NZYDkPNW6+QdSXRUAZ07jMAAAABMnEgAAAAAAAAAAAAAAAJtPmVD9XedIQE1wxn/j
cLx7ZezAAAAAAAAABPIZGVwb3NpdAAAAAAAAAAAAAAAAPC6Yz2Miy7+qLmI/jtM45UXhLRNQ
0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAGGoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runRecieveMessage.py
vaa_verify {'hash': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'result': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBB
hJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VA
CVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok='}
vaa_verify(result) BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoCiAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4F
BCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCDUDNAQlCDUEQv+0Iok=
sig_addr 727096228 JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34
sig_addr 86525623 Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ
keys: 0113947bd48b18e53fdaeee77f3473391ac727c63800000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000677561726469616e
submitted_txns, submitted_accts [<algosdk.transaction.ApplicationCallTxn object at 0x130949df0>, <algosdk.transaction.Applicat
ionCallTxn object at 0x13133eb40>] ['JM7ZHEEG74NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34', 'Z7AISPXE4YFESMU35JXBEM6JZSK
TVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ']
txns [<algosdk.transaction.ApplicationCallTxn object at 0x130949df0>, <algosdk.transaction.ApplicationCallTxn object at 0x1313
3eb40>, <algosdk.transaction.ApplicationCallTxn object at 0x13135fda0>]
result {'confirmed-round': 46380029, 'pool-error': '', 'txn': {'lsig': {'l': 'BiAEAQAgFCYBADEgMgMSRDEBIxJEMRCBBhJENhoBNhoDNhoC
iAADRCJDNQI1ATUAKDXwKDXxNAAVNQUjNQMjNQQ0AzQFDEEARDQBNAA0A4FBCCJYFzQANAMiCCRYNAA0A4EhCCRYBwA18TXwNAI0BCVYNPA08VACVwwUEkQ0A4FCCD
UDNAQlCDUEQv+0Iok='}, 'txn': {'apaa': ['dmVyaWZ5U2lncw==', 'ANiSEVJFcwHY1Zc3qDWGfHSsV73ILYo5WAQL0/CyI4hTTUjFW9SeG0CCN2eeEwN6HN
9FyO4NZYDkPNW6+QdSXRUA', 'E5R71Is5T/a7ud/NHM5Gscnxjg=', '12DTPMZ8Ac1/PI+sgXAaI+cdPTAtp24zJQShQTDFb58=']', 'apat': ['JM7ZHEEG74
NDGMWTZDQAH74QEN7YXRTRO7KAKO2U6HZVZ25IQOG3WMHN34', 'Z7AISPXE4YFESMU35JXBEM6JZSKTVGPASEUQXLM7XTEKVUHOYTRVQHVYVQ'], 'apid': 8652
5623, 'fv': 46380027, 'gen': 'testnet-v1.0', 'gh': 'SGO1GKSzyE7IEPItTxCByw9x8FmnrCDexi9/cOUJOiI=', 'grp': 'TpdF5dJu4rfmHtUypf9
Yepo006bCoJ122RO2ewzJnjk=', 'lv': 46381027, 'snd': 'EZATROXX2HISIRZDRGXW4LRQ46Z6IUJYYIHU3PJGP7P5IQDPKVX42N767A', 'type': 'appl
'}}}
box 510000
```

Final bal of the user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is updated and incremented

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xc8
0' 32
userBalance 510000
```

Final bal of the user 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d is not updated

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x07\xa1
' 32
userBalance 500000
```

txn id on algorand — ESNATZ6PCSOBDEF4I37QAHR2ZA2KXZG7MOKBDNG7BIQ72Q76N4GQ

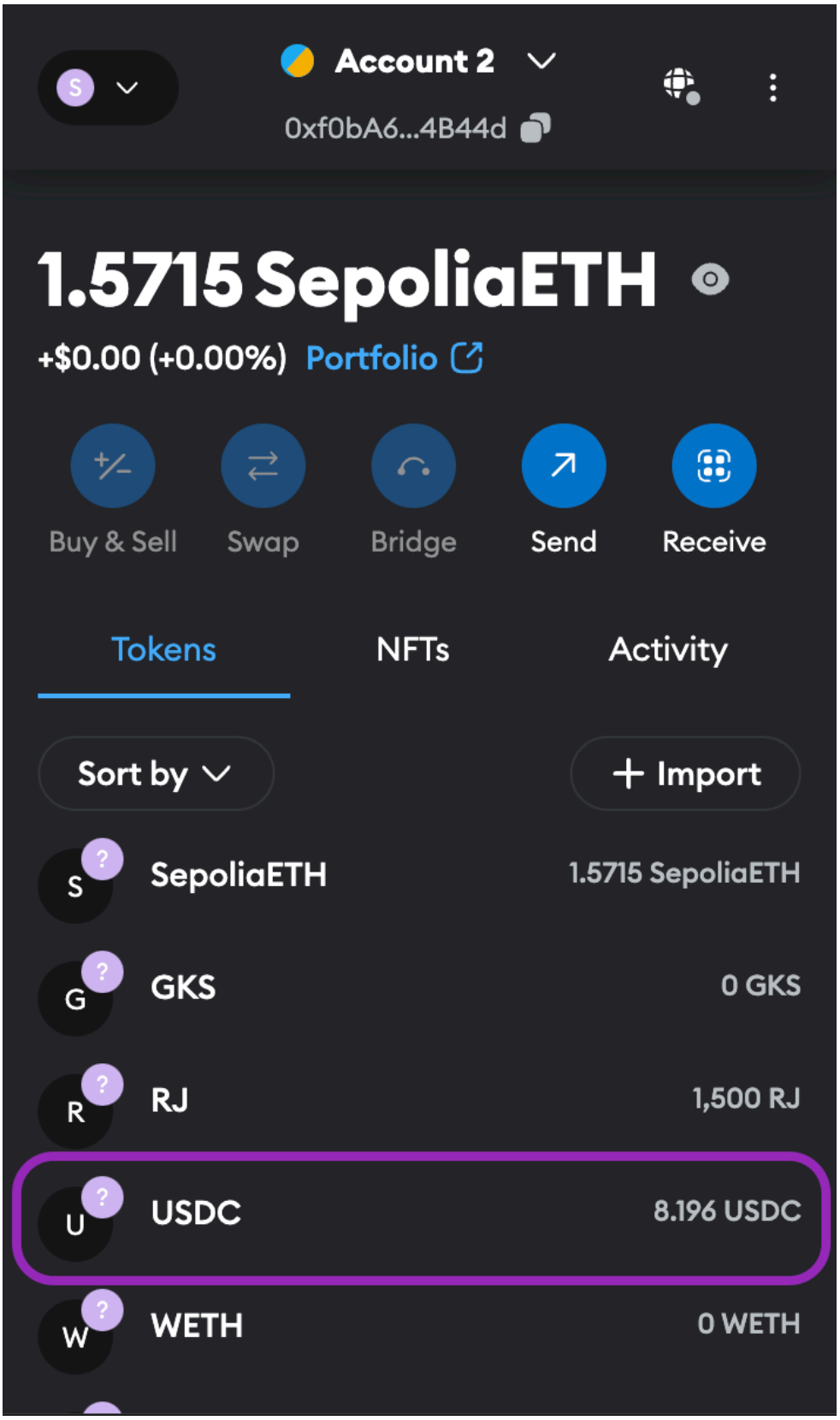
ESNATZ6PC...	46380029	AFIA...64JU	→	727096228	Application Call	1 minute
		CH6M...N2CE	→	727096157	Application Call	1 minute

4.6 Manipulating Normalized Wallet Address in VAA during Withdraw

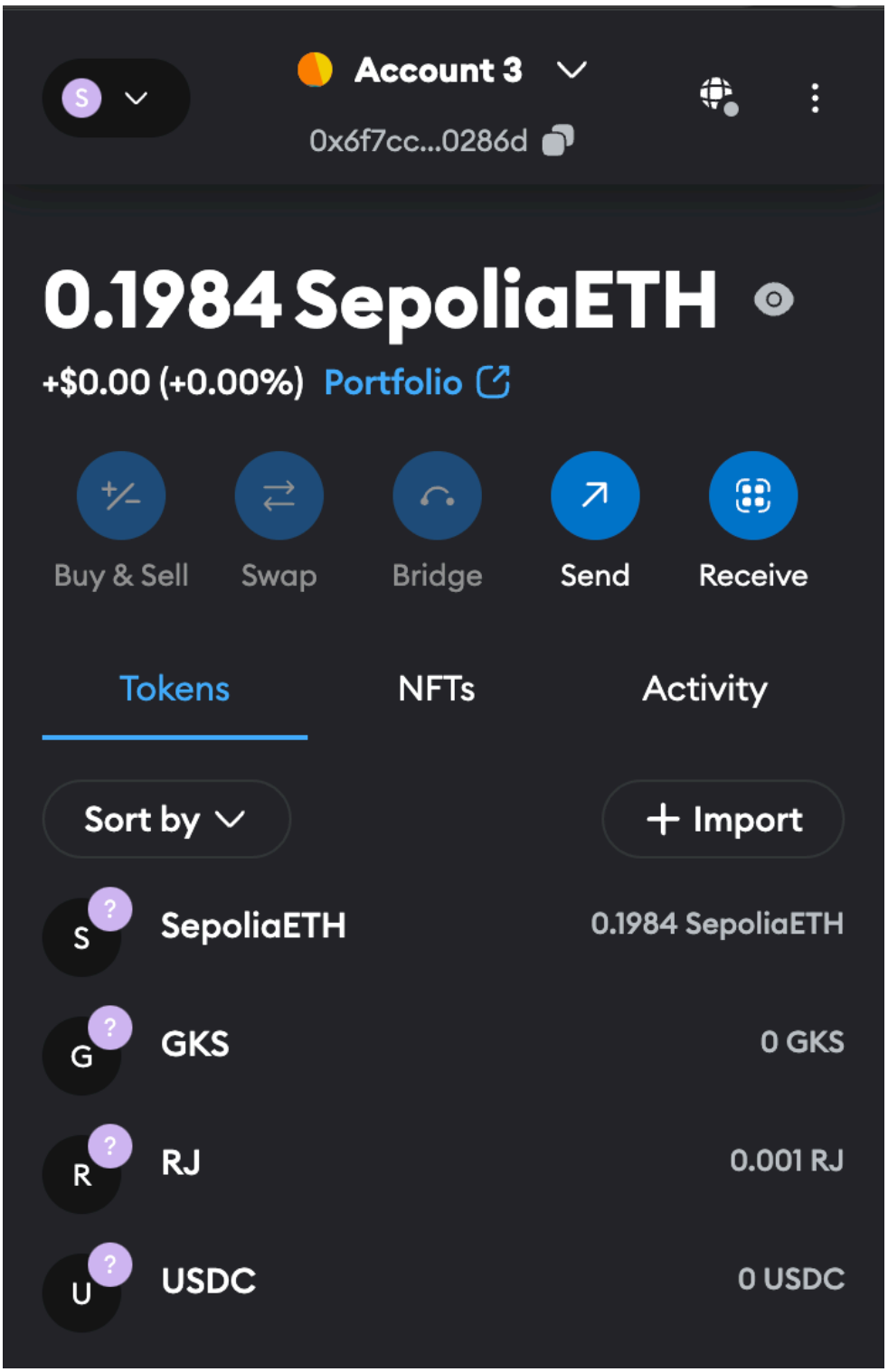
Result	Pass
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to EVM, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect address) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw

Steps Performed	

	1. **Check Initial Balances:**
	• User 1: 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
	• User 2: 0x6f7ccf76681da80cbd7ad4b28697a2881a70286d
	• Record the initial USDC balances of the wallet and Codex before initiating withdrawal.
	2. **Execute Withdrawal**
	• Execute the withdrawal transaction on the Codex side.
	• Collect the resulting VAA for further testing.
	3. **Correct VAA Collection:**
	• Retrieve the generated VAA from Wormhole scan.
	4. **Decrypt the Correct VAA:**
	• Decode the raw VAA into a readable format for inspection.
	• Decoded Data:
	  • Withdraw string: 7769746864726177
	  • Normalizer Wallet Address: 000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d
	  • Destination Chain Id: 0000000000002712
	  • Token Address Unified USDC: 4343545055534443000000000000000000000000000000000000000000000000000000000
	  • Destination Chain Id: 0000000000002712
	  • Amount: 00000000000000000000000000000000000000000000000000000000000000002710
	  • Native Bool: 00
	  • Transaction Id: 1452f659d05e3284ee4630b04bb6c7529edd8aa7b1173fd25f8350b452124e550000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300
	5. **Manipulate VAA:**
	• Change the normalizer wallet address in the VAA payload to simulate malicious behavior.
	• **Original Normalizer Address:** 000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d
	• **Incorrect Address:** 000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d
	6. **Encrypt and Test Altered VAA:**
	• Re-encrypt the manipulated VAA using the provided encryption script.
	• Feed the modified encrypted VAA to the receiveEncodedMsg function and validate balances.
	• No changes to user balances, ensuring verification works correctly.
	7. **Re-test with Correct VAA:**
	• Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended.
	• Confirm that the transaction succeeds and the user's balance updates as expected.
	8. **Final Validation:**
	• Ensure the manipulated VAA fails and does not alter balances of both user1 and user2.
	• User 1's balance (0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d) increases as expected.

Observation	**Step -1 Check for both the balances** EOA wallet usdc balance if the user 1- 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d 

EOA wallet usdc balance if the user 2 -
0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d




```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00
```

Step -2 Run Withdraw Function

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
10002 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
dest_private_key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
434354505553444300000000000000000000000000000000000000000000000000 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL
KVNGXKQ4OCKMU5Y
login 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 10002
236
10002
b'\x19Ethereum Signed Message:\n549{"Login_Address": "0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d", "Login_ChainId": "10002", "
Token_Address": "434354505553444300000000000000000000000000000000000000000000000000", "Receiver_Address": "0xf0ba633d8c8b2efea8b
988fe3b4ce3951784b44d", "Receiver_ChainId": "10002", "Amount": "10000", "Withdraw_Fee": "False"}AAAAAAAAAAAAAAAA8LpjPYyLLv6ouY
j+O0zjlReEtE0AAAAAAAAAnEkNDVFBVU0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0AAAAAAAAAnEgAAAAAA
AAAAAAAAAAAAAAAAAAAA
```

784b44d0000000000002712434354505553444300000000000000000000000000000000000
0000000000000000000000000271200000000000000000000000000000000000000000000
0000000000000000271000883255 48e5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1
b165eb0000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Payload
776974686472617700000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b4
4d0000000000000271243435450555344430000000000000000000000000000000000000000
0000000000000000000000271200000000000000000000000000000000000000000000000000
0000000000271000883255 48e5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb
0000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Step -5 Changing the decrypted raw vaa into wrong vaa

Changing the normalizer wallet address to user
000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d ==
000000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d

Wrong Vaa with changed normalizer address
0100000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf
7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a4f799e16ddabb45e
a89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b23
4eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000026007769746864
7261770000000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881
a70286d0000000000002712434354505553444300000000000000000000000
0000000000000000000000000000000000000000000271200000000000000000
0000000000000000000000000000000000000000000027100088325548e55
77f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb00000000000
000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Step -6 Encrypting wrong vaa with changed chain id

Encrypted wrong vaa
AQAAAAABAEXqdNXPMXm+pP+iBh7NtaK3uBjWmiZuP/v312N4ZyE9MLLJLyHj0fHaK7Kb1+gT+9R
6T3meFt2rtF6onbjV0sYBZ1FeAAAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCL
AAAAAAAAACYAd2I0aGRyYXcAAAAAAAAAAAAAAABvfM92aB2oDL161LKGI6KIGnAobQAAAAAA
ACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcQAIgyVUjlV39Xp6UsK43zV/1AyeYT+OOIWSk2Mm
6hsWXrAAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtI7MA

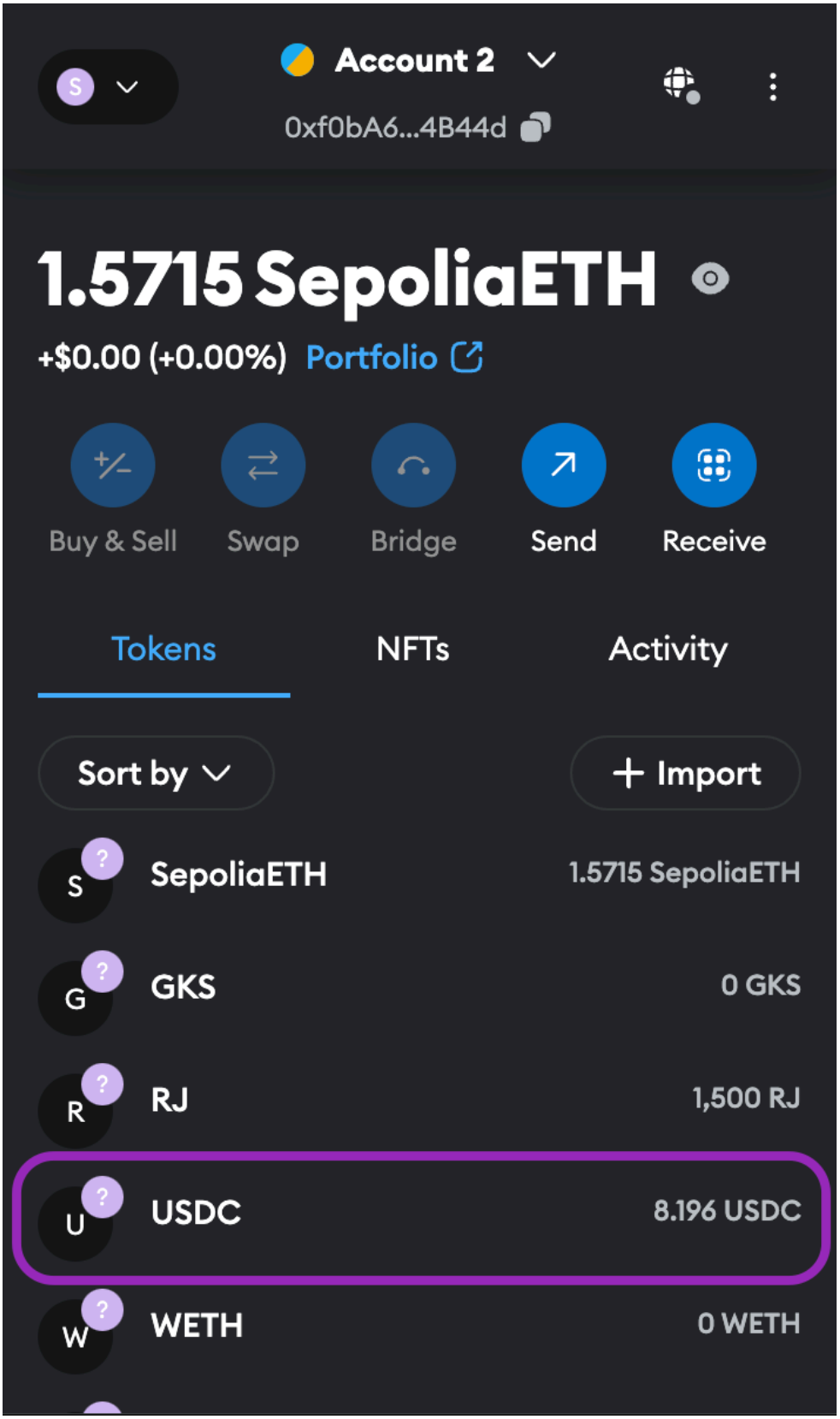
Step -7 Feeding the wrong raw vaa to the receiveEncodeMsg

```
const wrongData =
"0100000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a4f7
99e16ddabb45ea89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000
0000000260077697468647261770000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d0000000000002712434354505555
3444300000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000027100088325548e5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb0000000000000000000000000098ed3e6543f57
79d210135c319ff8dc2f1ed97b300"
const wrData =  Buffer.from(wrongData,"hex");
// console.log("This is wrong data",wrData)
// console.log(Buffer.from(wrData).toString("hex"),"This is the wrong data decoded message from algorand vaa")
const rx = await tokenManager
  .receiveEncodedMsg(
    wrData
  )
  .then(utils.wait)
// Vaa == signature
console.log((await utils.readCCTPDomain(10002)).toString());
```

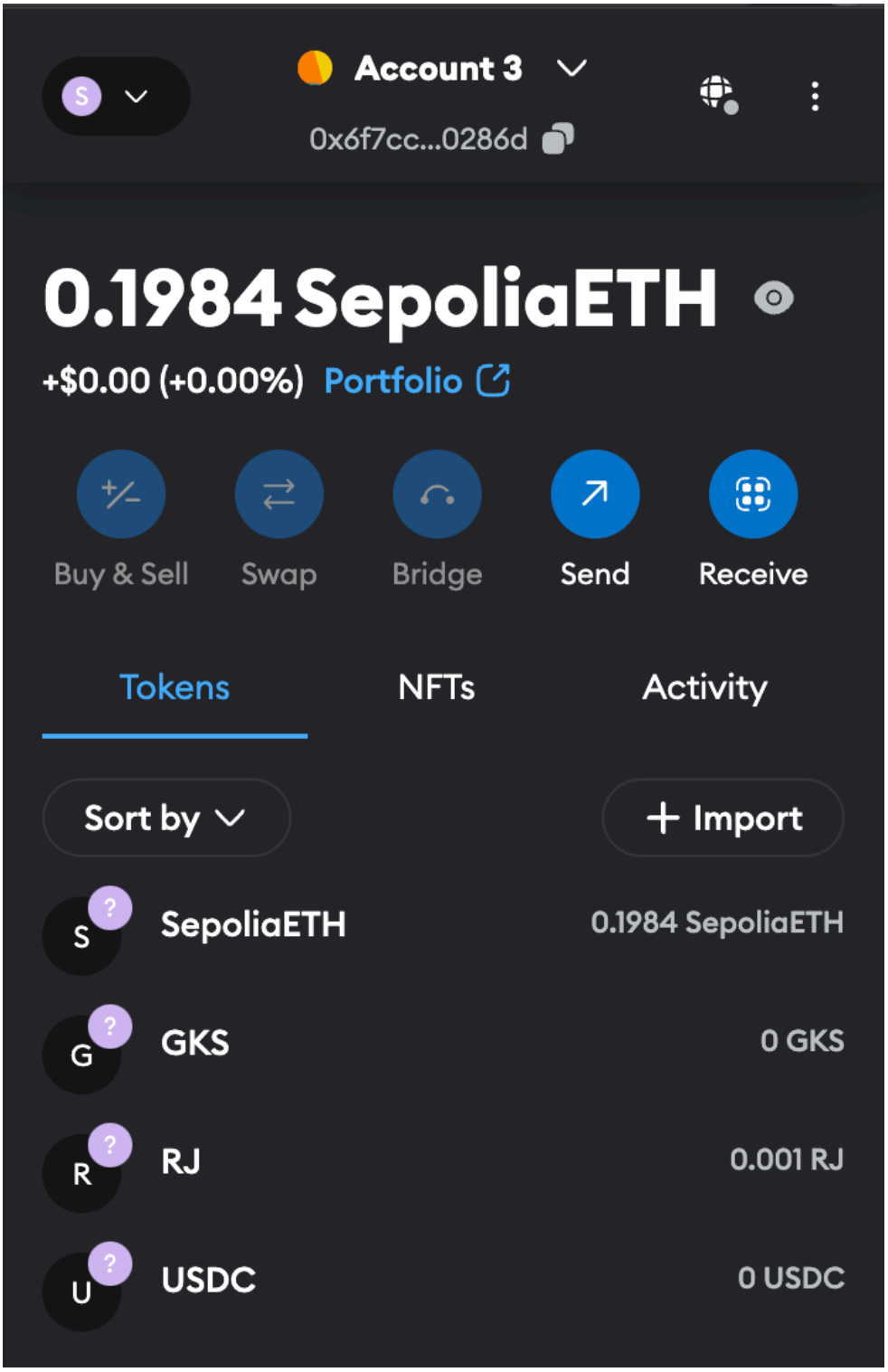
```
    error: Error: processing response error (body="{\"jsonrpc\":\"2.0\",\"id\":49,\"error\":{\"code\":3,\"message\":\"executio
n reverted: VM signature invalid\",\"data\":\"0x08c379a0000000000000000000000000000000000000000000000000000000000000020000000
0000000000000000000000000000000000000000000000000000000014564d207369676e61747572652069 6e76616c6964000000000000000000000000\"}}
\n", error={"code":3,"data":"0x08c379a0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000
0000000000000000000000000000000000000000000014564d207369676e61747572652069 6e76616c6964000000000000000000000000"}, requestBody="{\"m
ethod\":\"eth_estimateGas\",\"params\":[{\"type\":\"0x2\",\"maxFeePerGas\":\"0x3b88d7692\",\"maxPriorityFeePerGas\":\"0x59682f
00\",\"from\":\"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d\",\"to\":\"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3\",\"data\":\"
0x496d76ff0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000
0000000013501000000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a
4f799e16ddabb45ea89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
0026007769746864726177000000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d0000000000000271243435450555344430000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
5548e5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb0000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300
0000000000000000000000\"}],\"id\":49,\"jsonrpc\":\"2.0\"}", requestMethod="POST", url="https://ethereum-sepolia-rpc.publicnode
.com", code=SERVER_ERROR, version=web/5.7.1)
        at Logger.makeError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/c
ontracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:269:28)
        at Logger.throwError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/
contracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:281:20)
        at /Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/chains/ev
m/node_modules/@ethersproject/web/src.ts/index.ts:341:28
        at step (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/cha
ins/evm/node_modules/@ethersproject/web/lib/index.js:33:23)
        at Object.next (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contra
cts/chains/evm/node_modules/@ethersproject/web/lib/index.js:14:53)
        at fulfilled (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contract
s/chains/evm/node_modules/@ethersproject/web/lib/index.js:5:58)
        at processTicksAndRejections (node:internal/process/task_queues:95:5) {
      reason: 'processing response error',
      code: 'SERVER_ERROR',
      body: '{"jsonrpc":"2.0","id":49,"error":{"code":3,"message":"execution reverted: VM signature invalid","data":"0x08c379a
0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000
014564d207369676e61747572652069 6e76616c6964000000000000000000000000"}}\n',
      error: [Error],
      requestBody: '{"method":"eth_estimateGas","params":[{"type":"0x2","maxFeePerGas":"0x3b88d7692","maxPriorityFeePerGas":"0
x59682f00","from":"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d","to":"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3","data":"0x496
d76ff0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000
0000135010000000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a4f799
e16ddabb45ea89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000000260
0776974686472617700000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d0000000000000271243435450555344430000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb0000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b30000000
```

Step -8 Final balance of the user see if the balance gets uploaded

Final Balance of user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is not updated as expected



Final Balance of user 0x6f7ccF76681dA80cBD7AD4b28697A2881A70286d is not Updated as expected



Step -9 Entering the correct raw vaa to see if the balance gets updated

<u>Entering the correct VAA</u>

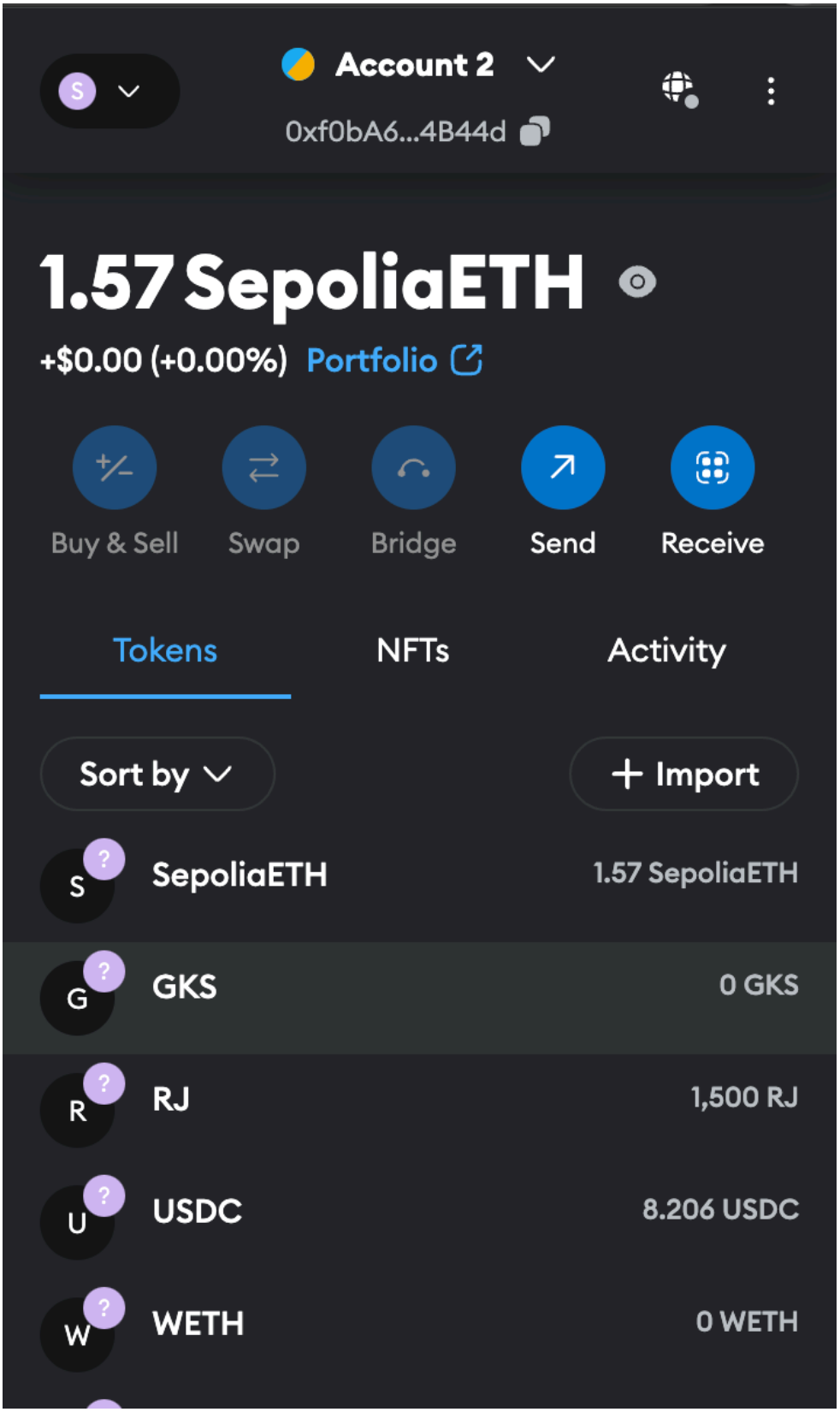
AQAAAAABAEXqdNXPMXm+pP+iBh7NtaK3uBjWmiZuP/v312N4ZyE9MLLJLyHj0fHaK7Kb1+gT+9R
6T3meFt2rtF6onbjV0sYBZ1FeAAAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCL
AAAAAAAAACYAd2I0aGRyYXcAAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQAAAAAA
ACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA
AAAAAAACcQAIgyVUjIV39Xp6UsK43zV/1AyeYT+OOIWSk2Mm6hsWXrAAAAAAAAAAAAAAAAmO
0+ZUP1d50hATXDGf+NwvHtI7MA

```
gurkiratsingh@Gurkirats-MacBook-Air evm % npm run receiveEncodedMsg

> TokenManager@1.0.0 receiveEncodedMsg
> ts-node ./ts-scripts/main.ts --receiveEncodedMsg

_proxy {
  '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
  '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
  '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
  '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
  '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
}
_proxy {
  '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
  '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
  '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
  '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
  '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
}
0
```

[This is a Sepolia **Testnet** transaction only]

Transaction Hash:	0x44483bf903a68ea2be8a15ff0e037d05a761a35c4cfdd5face1c3399492a5839
Status:	Success
Block:	7215290 3 Block Confirmations
Timestamp:	1 mins ago (Dec-05-2024 08:42:36 AM UTC)
Transaction Action:	Call Receive Encoded Msg Function by 0xf0bA633D...51784B44d on 0x98Ed3E65...2f1ed97b3
From:	0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d
Interacted With (To):	0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3
ERC-20 Tokens Transferred:	All Transfers Net Transfers From 0x98Ed3E65...2f1ed97b3 To 0xf0bA633D...51784B44d For 0.01 ERC-20: USDC (USDC)
Value:	0 ETH
Transaction Fee:	0.0014501655039964 ETH
Gas Price:	9.0188348 Gwei (0.0000000090188348 ETH)

Final balance of the user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is updated and incremented



Final balance in algorand of the user1
0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d

	(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py login chain id 10002 login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d assest Chain id 65537 assest address 434354505553444300000000000000000000000000000000000000000000 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0b\xe6 \xe0' 32 userBalance 780000

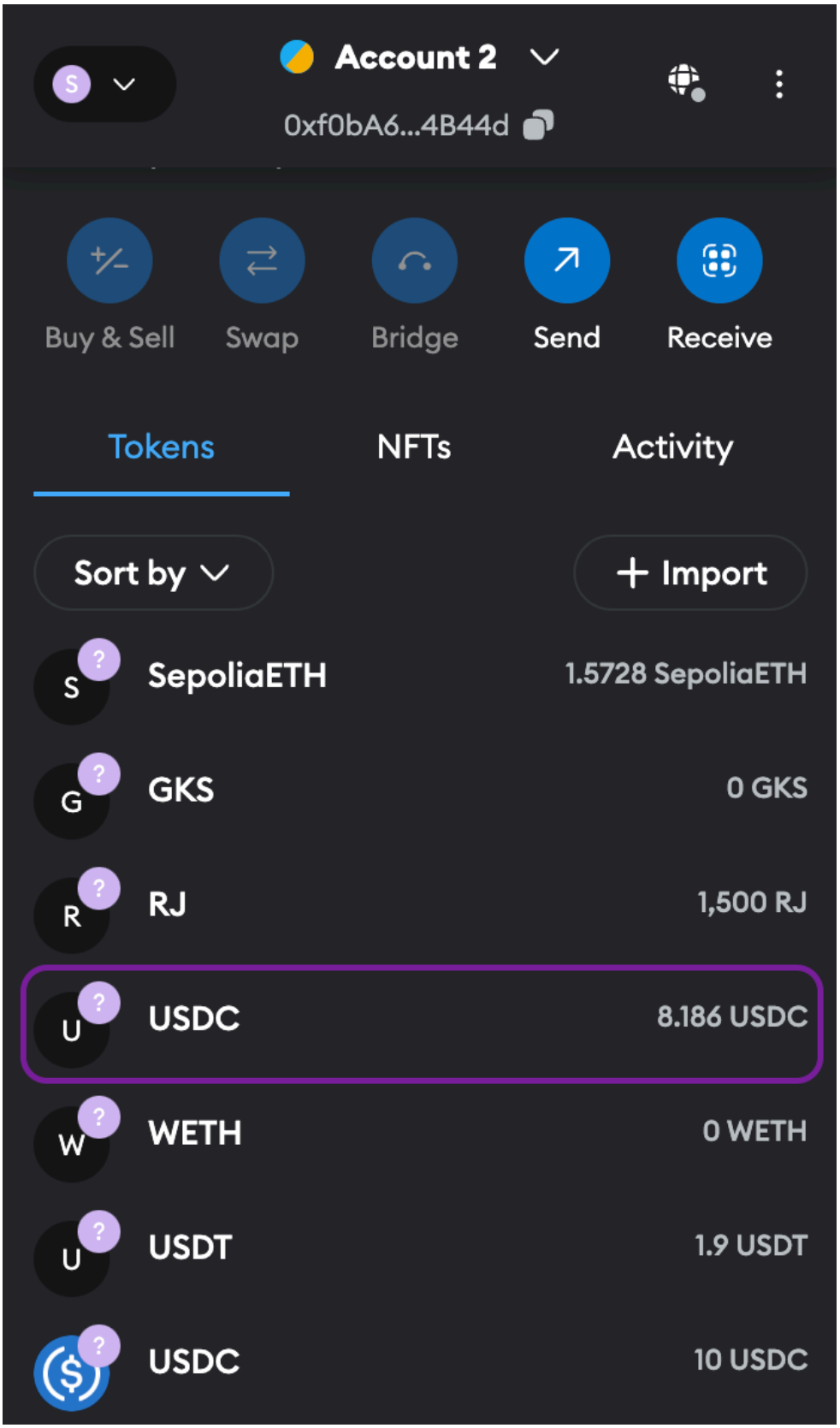
4.7 Manipulating String in VAA during Withdraw

Result	Pass
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to EVM, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect Withdraw string) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw

Steps Performed	1. **Check Initial Balances:**
	• **EOA Wallet Address**: 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d.
	• Record the initial USDC balances of the wallet and Codex before initiating withdrawal.
	2. **Execute Withdrawal**
	• Execute the withdrawal transaction on the Codex side.
	• Collect the resulting VAA for further testing.
	3. **Correct VAA Collection:**
	• Retrieve the generated VAA from Wormhole scan.
	4. **Decrypt the Correct VAA**
	• Decode the raw VAA into a readable format for inspection.
	• Decoded Data:
	  • Withdraw string: 7769746864726177
	  • Normalizer Wallet Address: 000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d
	  • Destination Chain Id: 0000000000002712
	  • Token Address Unified USDC: 434354505553444300000000000000000000000000000000000000000000000000000000
	  • Destination Chain Id: 0000000000002712
	  • Amount: 0000000000000000000000000000000000000000000000000000000000000000000002710
	  • Native Bool: 00
	  • Transaction Id: 1452f659d05e3284ee4630b04bb6c7529edd8aa7b1173fd25f8350b452124e550000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300
	5. **Manipulate VAA:**
	• Change the withdraw string from 7769746864726177 (withdraw) to 68656c6c6f000000 (hello).
	6. **Encrypt and Test Altered VAA:**
	• Re-encrypt the manipulated VAA using the provided encryption script.

	- Feed the modified encrypted VAA to the receiveEncodedMsg function and validate balances. - No changes to user balances, ensuring verification works correctly. 7. **Re-test with Correct VAA:** - Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended. - Confirm that the transaction succeeds and the user's balance updates as expected. 8. **Final Validation:** - Ensure the manipulated VAA fails and does not alter balances of user. - User's balance (0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d) increases as expected.

Observation	**Step -1 Check for both the balances** EOA wallet usdc balance if the user - 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B4



```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0c
5\x00' 32
userBalance 800000
```

Step -2 Run Withdraw Function

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
10002 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
dest_private_key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
4343545055534443000000000000000000000000000000000000000000000000 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL
KVNGXKQ4OCKMU5Y
login 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 10002
236
10002
b'\x19Ethereum Signed Message:\n549{"Login_Address": "0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d", "Login_ChainId": "10002", "
Token_Address": "4343545055534443000000000000000000000000000000000000000000000000", "Receiver_Address": "0xf0ba633d8c8b2efea8b
988fe3b4ce3951784b44d", "Receiver_ChainId": "10002", "Amount": "10000", "Withdraw_Fee": "False"}AAAAAAAAAAAAAAAA8LpjPYyLLv6ouY
j+O0zjlReEtE0AAAAAAAAnEkNDVFBVU0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0AAAAAAAAnEgAAAAAA
AAAAAAAAAAAAAAAAAAAA
```

Decrypted raw vaa
01000000000100a9a37bb4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb1122
8f7ae2f47abf9123a8650362d5f4235018ab8f42c6a20a6ece11048aff34f501675061ba0000000000
0811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000002
4007769746864726177000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce395178
4b44d0000000000002712434354505553444300000000000000000000000000000000000000
0000000000000000000000027120000000000000000000000000000000000000000000000000
0000000000000271000cf101297eacfc45a2aada32903baa3658580c7459b43282361089107b59a
715b000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Payload
7769746864726177000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b4
4d0000000000002712434354505553444300000000000000000000000000000000000000
0000000000000000000027120000000000000000000000000000000000000000000000000
0000000000271000cf101297eacfc45a2aada32903baa3658580c7459b43282361089107b59a715
b000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Withdraw string == 7769746864726177

normalizer wallet addr ==
000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d

dest chain id == 0000000000002712

token address unified USDC ==
434354505553444300000000000000000000000000000000000000000000000000

destination chain id == 0000000000002712

this is the amount ==
0000000000000000000000000000000000000000000000000000000000000002710

is native bool == 00

txn id ==
1452f659d05e3284ee4630b04bb6c7529edd8aa7b1173fd25f8350b452124e5500000000000000
000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Step -5 Changing the decrypted raw vaa into wrong vaa

Changing the string of `withdraw` == `hello` viz 7769746864726177 ==
68656c6c6f000000

Wrong Vaa Is below with changed withdraw string to hello
01000000000100a9a37bb4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb1122
8f7ae2f47abf9123a8650362d5f4235018ab8f42c6a20a6ece11048aff34f501675061ba0000000000
0811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000002
40068656c6c6f000000000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce39517
84b44d0000000000002712434354505553444300000000000000000000000000000000000000
0000000000000000000000027120000000000000000000000000000000000000000000000000
0000000000000271000cf101297eacfc45a2aada32903baa3658580c7459b43282361089107b59
a715b000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Step -6 Encrypting the wrong vaa with changed chain id

Encrypting wrong vaa
AQAAAAABAKmje7Q4B3hjMwTV92zvj8DGuHvhE90KmknnXQCAzusRIo964vR6v5EjqGUDYtX0I1A
Yq49CxqIKbs4RBIr/NPUBZ1BhugAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCL
AAAAAAAAACQAaGVsbG8AAAAAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQAAAAA
AACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcQAM8QEpfqz8RaKq2jKQO6o2WFgMdFm0MoI2EI
kQe1mnFbAAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

Step - 7 Feeding the wrong raw vaa to the receiveEncodeMsg

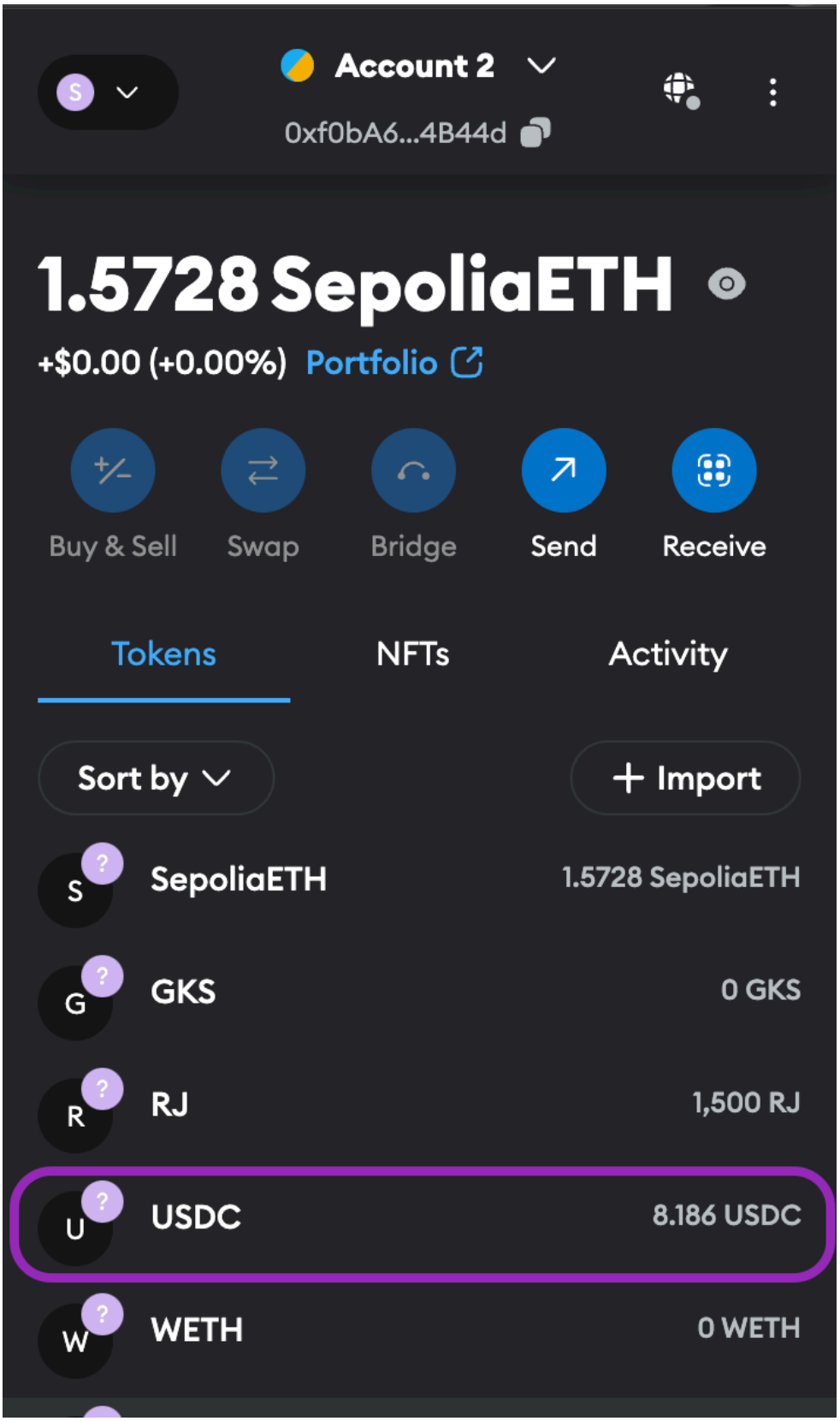
```
const wrongData =
"0100000000100a9a37bb4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb11228f7ae2f47abf9123a8650362d5f4235018ab8
f42c6a20a6ece11048aff34f501675061ba00000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000
0000000240068656c6c6f0000000000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000002712434354505 55
3444300000000000000000000000000000000000000000000000000000000000000271200000000000000000000000000000000000000000000000
00000000000271000cf101297eacfc45a2aada32903baa3658580c7459b43282361089107b59a715b0000000000000000000000098ed3e6543f57
79d210135c319ff8dc2f1ed97b300"
// const wrData =  Buffer.from(wrongData,"hex");
// console.log("This is wrong data",wrData)
// console.log(Buffer.from(wrData).toString("hex"),"This is the wrong data decoded message from algorand vaa")
const rx = await tokenManager
  .receiveEncodedMsg(
    wrongData
  )
  .then(utils.wait)
// Vaa == signature
console.log((await utils.readCCTPDomain(10002)).toString());
```

```
gurkiratsingh@Gurkirats-MacBook-Air evm % npm run receiveEncodedMsg

> TokenManager@1.0.0 receiveEncodedMsg
> ts-node ./ts-scripts/main.ts --receiveEncodedMsg

_proxy {
  '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
  '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10002': '0x98Ed3E6543F5779D210135C319ff8dc2f1ed97b3',
  '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
  '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
  '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
}
<ref *1> Error: cannot estimate gas; transaction may fail or may require manual gas limit [ See: https://links.ethers.org/v5-e
rrors-UNPREDICTABLE_GAS_LIMIT ] (error={"reason":"execution reverted: VM signature invalid","code":"UNPREDICTABLE_GAS_LIMIT","
method":"estimateGas","transaction":{"from":"0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d","maxPriorityFeePerGas":{"type":"BigNu
mber","hex":"0x59682f00"},"maxFeePerGas":{"type":"BigNumber","hex":"0x04168b6eec"},"to":"0x98Ed3E6543F5779D210135C319ff8dc2f1e
d97b3","data":"0x496d76ff0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000
0000000000000000000000001350100000000100a9a37bb4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb11228f7ae2f47abf9123a86
50362d5f4235018ab8f42c6a20a6ece11048aff34f501675061ba00000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec
08b0000000000000002400686 56c6c6f0000000000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000002712434354505
553444300000000000000000000000000000000000000000000000000000000000000271200000000000000000000000000000000000000000000000000
00000271000cf101297eacfc45a2aada32903baa3658580c7459b43282361089107b59a715b0000000000000000000000098ed3e6543f5779d210135c319f
f8dc2f1ed97b3000000000000000000000000","type":2,"accessList":null},"error":{"reason":"processing response error","code":"SERVE
R_ERROR","body":{\"jsonrpc\":\"2.0\",\"id\":49,\"error\":{\"code\":3,\"message\":\"execution reverted: VM signature invalid\"
,\"data\":\"0x08c379a0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000
0000000000000000000000014564d207369676e617475726520696e76616c696400000000000000000000000\"}}\n","error":{"code":3,"data":"0x08
c379a0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000
00000014564d207369676e617475726520696e76616c696400000000000000000000000\"},"requestBody":"{\"method\":\"eth_estimateGas\",\"pa
rams\":[{\"type\":\"0x2\",\"maxFeePerGas\":\"0x4168b6eec\",\"maxPriorityFeePerGas\":\"0x59682f00\",\"from\":\"0xf0ba633d8c8b2e
fea8b988fe3b4ce3951784b44d\",\"to\":\"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3\",\"data\":\"0x496d76ff0000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
b4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb11228f7ae2f47abf9123a8650362d5f4235018ab8f42c6a20a6ece11048aff34f50167
5061ba00000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000002400686 56c6c6f0000000000000000
00000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000002712434354505553444300000000000000000000000000000000000
0000000000000000000027120000000000000000000000000000000000000000000000000000000000000000000000000000000000000271000cf101297eacfc45a2aada32903baa3658580
c7459b43282361089107b59a715b0000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b3000000000000000000000000\"}],\"id\
":49,\"jsonrpc\":\"2.0\"}","requestMethod":"POST","url":"https://ethereum-sepolia-rpc.publicnode.com"}}, tx={"data":"0x496d76f
f0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000
1350100000000100a9a37bb4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb11228f7ae2f47abf9123a8650362d5f4235018ab8f42c6a
20a6ece11048aff34f501675061ba00000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000002400686
56c6c6f0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
45a2aada32903baa3658580c7459b43282361089107b59a715b0000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300000000000000
0000000000000","to":{},"from":"0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d","type":2,"maxFeePerGas":{"type":"BigNumber","hex":"
0x04168b6eec"},"maxPriorityFeePerGas":{"type":"BigNumber","hex":"0x59682f00"},"nonce":{},"gasLimit":{},"chainId":{}}, code=UNP
REDICTABLE_GAS_LIMIT, version=abstract-signer/5.7.0)
    at Logger.makeError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contr
acts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:269:28)
    at Logger.throwError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/cont
racts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:281:20)
```

```
    error: [illegible]onse error (body="{\"jsonrpc\":\"2.0\",\"id\":49,\"error\":{\"code\":3,\"message\":\"executio
n reverte[illegible] VM signature invalid\",\"data\":\"0x08c379a0000000000000000000000000000000000000000000000000000000000000000200000000
0000000000[illegible]0000000000000000000000014564d207369676e617475726520696e76616c696400000000000000000000000\"}}
\n", error={"code":3,"data":"0x08c379a0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000
0000000000000000000000014564d207369676e617475726520696e76616c696400000000000000000000000\"}, requestBody="{\"m
ethod\":\"eth_estimateGas\",\"params\":[{\"type\":\"0x2\",\"maxFeePerGas\":\"0x4168b6eec\",\"maxPriorityFeePerGas\":\"0x59682f
00\",\"from\":\"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d\",\"to\":\"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3\",\"data\":\"
0x496d76ff0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000
0000000001350100000000100a9a37bb4380778633304d5f76cef8fc0c6b87be113dd0a9a49e75d0080ceeb11228f7ae2f47abf9123a8650362d5f4235018
ab8f42c6a20a6ece11048aff34f501675061ba00000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
00240068656c6c6f0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
1297eacfc45a2aada32903baa3658580c7459b43282361089107b59a715b0000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300
0000000000000000000000\"}],\"id\":49,\"jsonrpc\":\"2.0\"}", requestMethod="POST", url="https://ethereum-sepolia-rpc.publicnode
.com", code=SERVER_ERROR, version=web/5.7.1)
        at Logger.makeError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/c
ontracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:269:28)
        at Logger.throwError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/
contracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:281:20)
        at /Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/chains/ev
m/node_modules/@ethersproject/web/src.ts/index.ts:341:28
        at step (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/cha
ins/evm/node_modules/@ethersproject/web/lib/index.js:33:23)
        at Object.next (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contra
cts/chains/evm/node_modules/@ethersproject/web/lib/index.js:14:53)
        at fulfilled (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contract
s/chains/evm/node_modules/@ethersproject/web/lib/index.js:5:58)
        at processTicksAndRejections (node:internal/process/task_queues:95:5) {
      reason: 'processing response error',
      code: 'SERVER_ERROR',
      body: '{"jsonrpc":"2.0","id":49,"error":{"code":3,"message":"execution reverted: VM signature invalid","data":"0x08c379a
0000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000
014564d207369676e617475726520696e76616c696400000000000000000000000\"}}\n',
      error: [Error],
      requestBody: '{"method":"eth_estimateGas","params":[{"type":"0x2","maxFeePerGas":"0x4168b6eec","maxPriorityFeePerGas":"0
x59682f00","from":"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d","to":"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3","data":"0x496
```

Step -8 Final balance of the user see if the bal gets uploaded

Final Balance of user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is not updated as expected



Step -9 Entering the correct raw vaa to see if balance gets updated

Entering the correct VAA

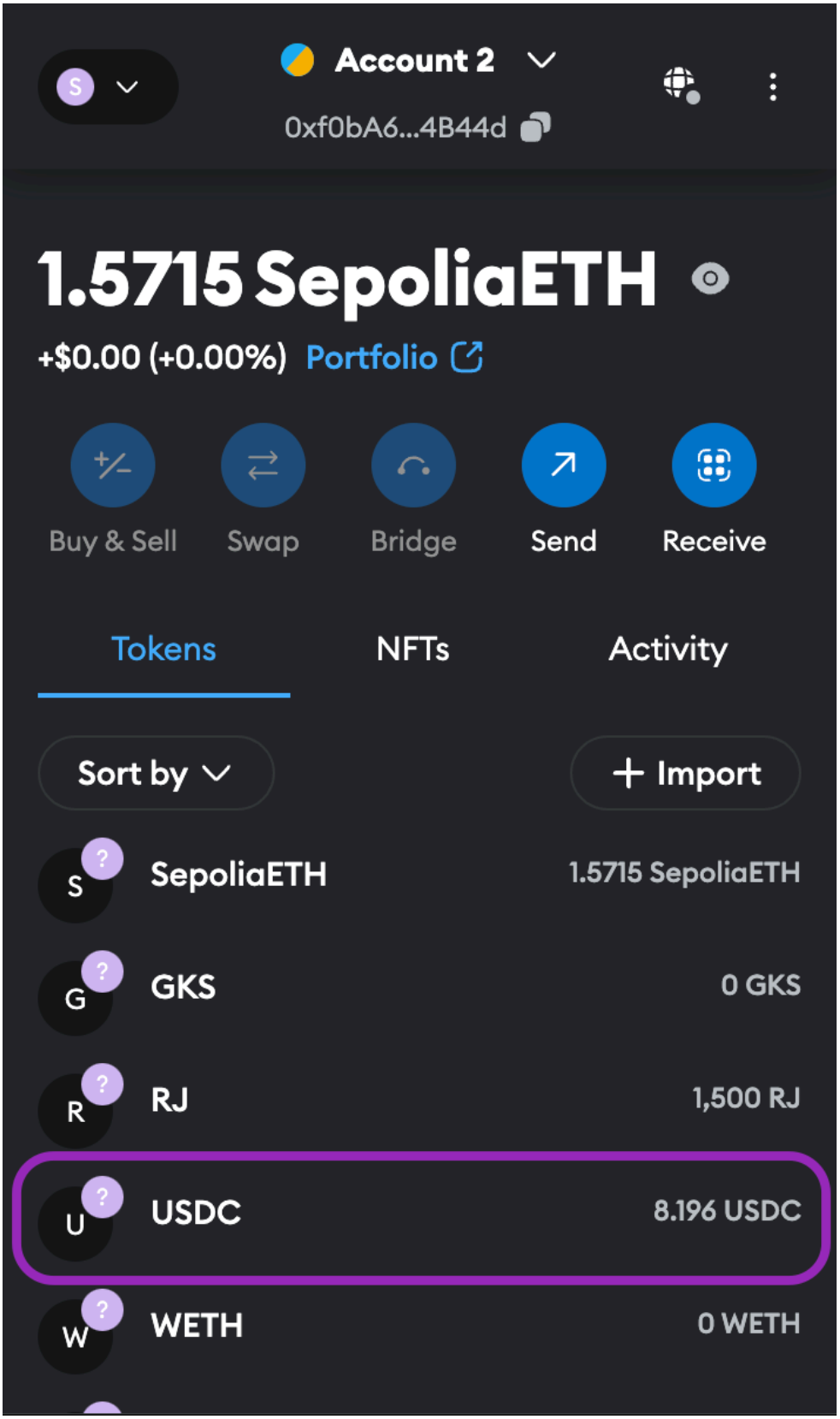
AQAAAAABAKmje7Q4B3hjMwTV92zvj8DGuHvhE90KmknnXQCAzusRIo964vR6v5EjqGUDYtX0I1A
Yq49CxqIKbs4RBIr/NPUBZ1BhugAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCL
AAAAAAAAACQAd2I0aGRyYXcAAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQAAAAAA
ACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcQAM8QEpfqz8RaKq2jKQO6o2WFgMdFm0MoI2EIk
Qe1mnFbAAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

```
● gurkiratsingh@Gurkirats-MacBook-Air evm % npm run receiveEncodedMsg

  > TokenManager@1.0.0 receiveEncodedMsg
  > ts-node ./ts-scripts/main.ts --receiveEncodedMsg

  _proxy {
    '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
    '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
    '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
    '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
    '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
  }
  _proxy {
    '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
    '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
    '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
    '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
    '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
  }
  0
```

[This is a Sepolia **Testnet** transaction only]

Transaction Hash:	0x70036e2794db2a075b22ed2310debf62e62dc3c9ba2d919d8cfcfd83f7f6ebb3
Status:	Success
Block:	7215026 16 Block Confirmations
Timestamp:	3 mins ago (Dec-05-2024 07:45:48 AM UTC)
Transaction Action:	Call Receive Encoded Msg Function by 0xf0bA633D...51784B44d on 0x98Ed3E65...2f1ed97b3
From:	0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d
Interacted With (To):	0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3
ERC-20 Tokens Transferred:	All Transfers Net Transfers From 0x98Ed3E65...2f1ed97b3 To 0xf0bA633D...51784B44d For 0.01 ERC-20: USDC (USDC)
Value:	0 ETH
Transaction Fee:	0.001359679334094445 ETH
Gas Price:	8.456085365 Gwei (0.000000008456085365 ETH)

Final bal of the user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is updated and incremented

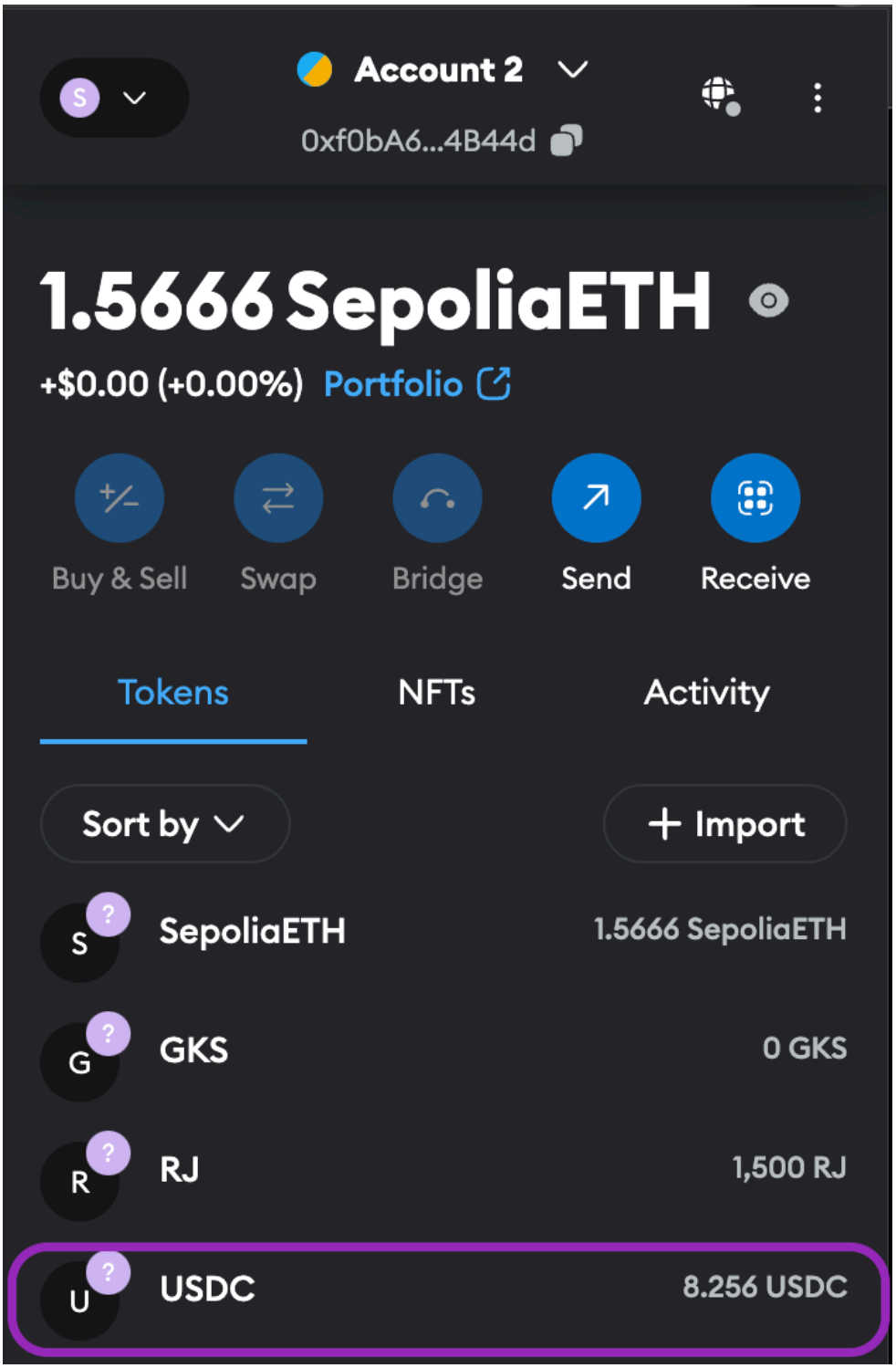


Final balance in algorand of user1
0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d

	(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py login chain id 10002 login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d assest Chain id 65537 assest address 434354505553444300000000000000000000000000000000000000000000000 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0c\r\x f0' 32 userBalance 790000

4.8 Manipulating Amount in VAA during Withdraw

Result	**Pass**
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to EVM, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect amount) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw
Steps Performed	1. **Check Initial Balances:** • **EOA Wallet Address**: 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d. • Record the initial USDC balances of the wallet and Codex before initiating withdrawal. 2. **Execute Withdrawal** • Execute the withdrawal transaction on the Codex side. • Collect the resulting VAA for further testing. 3. **Correct VAA Collection:** • Retrieve the generated VAA from Wormhole scan. 4. **Decrypt the Correct VAA** • Decode the raw VAA into a readable format for inspection. 5. **Manipulate VAA:** • Change the Amount. • **Original Value**: 000000000000000000000000000000000000000000000000000000000000c350 == 50000 • **Changed Value**: 0000000000000000000000000000000000000000000000000000000000004e20 == 20000 6. **Encrypt and Test Altered VAA:** • Re-encrypt the manipulated VAA using the provided encryption script. • Feed the modified encrypted VAA to the receiveEncodedMsg function and validate balances. • No changes to user balances, ensuring verification works correctly. 7. **Re-test with Correct VAA:** • Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended. • Confirm that the transaction succeeds and the user's balance updates as expected. 8. **Final Validation:** • Ensure the manipulated VAA fails and does not alter balances of the user. • User's balance (0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d) increases as expected.

Observation	**Step -1 Check for both the balances** EOA wallet usdc balance of the user - 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d  Initial balance in algorand of the user1 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d

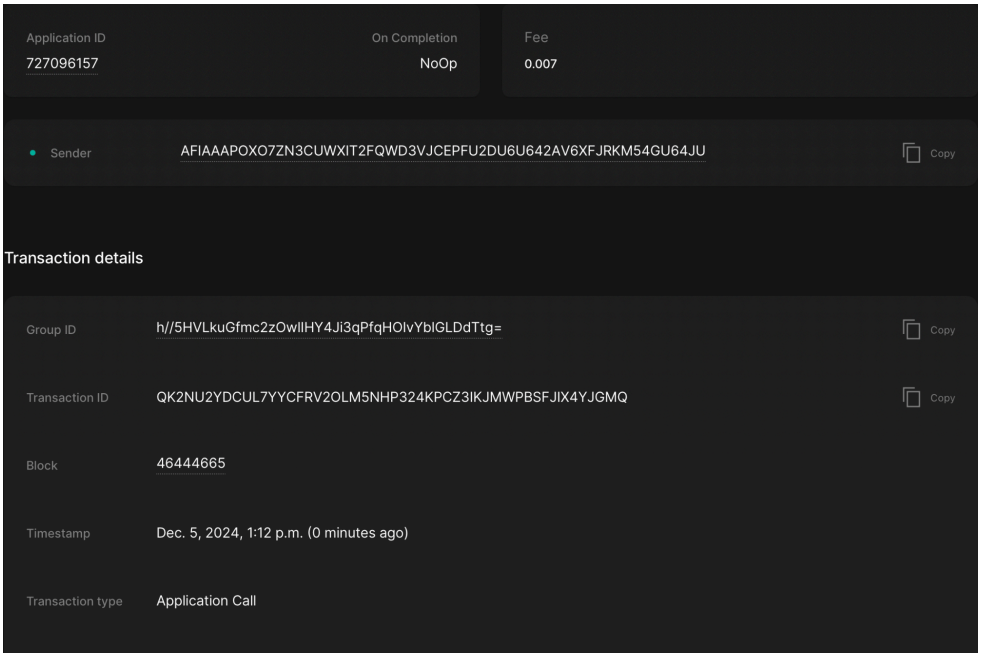
```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0b#\x9
0' 32
userBalance 730000
```

Step -2 Run Withdraw Function

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
10002 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
dest_private_key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
4343545055534443000000000000000000000000000000000000000000000000 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL
KVNGXKQ4OCKMU5Y
login 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 10002
236
10002
b'\x19Ethereum Signed Message:\n549{"Login_Address": "0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d", "Login_ChainId": "10002", "
Token_Address": "4343545055534443000000000000000000000000000000000000000000000000", "Receiver_Address": "0xf0ba633d8c8b2efea8b
988fe3b4ce3951784b44d", "Receiver_ChainId": "10002", "Amount": "50000", "Withdraw_Fee": "False"}AAAAAAAAAAAAAAAA8LpjPYyLLv6ouY
j+O0zjlReEtE0AAAAAAAAnEkNDVFBVU0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0AAAAAAAAnEgAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
userBalance 680000
after algo amount of codex 2000000
after algo amount of msg processor 2000000
```

Step -3 Collecting txn id from algorand scan

Txn hash -
QK2NU2YDCUL7YYCFRV2OLM5NHP324KPCZ3IKJMWPBSFJIX4YJGMQ



Wormhole scan link
https://wormholescan.io/#/tx/QK2NU2YDCUL7YYCFRV2OLM5NHP324KPCZ3IKJMWPBSFJIX4YJGMQ?network=Testnet

Step -4 Decrypting correct raw Vaa

Correct Vaa
AQAAAAABALHciGeBKdbYbHUEa8LC8YruGSBWRCVnUSFVhTqV9QHVG6T+60verFkpCB8vWzWlAmb+MBuMwAPkAjVRjh9oo9YBZ1GmpwAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAAACgAd2l0aGRyYXcAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQAAAAAACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAIK02msDFRf8YEWNdOWzrTv3riniztCks
s8MipRfmEmZAAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

Decrypted raw vaa

01000000000100b1dc88678129d6d86c75046bc2c2f18aee19205644256751215585 3a95f501d51b
a4feeb4bdeac5929081f2f5b35a50266fe301b8cc003e40235518e1f68a3d6016751a6a700000000
000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
0028007769746864726177000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce39
51784b44d0000000000002712434354505553444300000000000000000000000000000000000000
000000000000000000000000000271200000000000000000000000000000000000000000000000
0000000000000000c3500082b4da6b031517fc60458d74e5b3ad3bf7ae29e2ced0a4b2cf0c8a94
5f984999000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Payload

7769746864726177000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b4
4d0000000000002712434354505553444300000000000000000000000000000000000000000000
000000000000000000000271200000000000000000000000000000000000000000000000000000
0000000000c3500082b4da6b031517fc60458d74e5b3ad3bf7ae29e2ced0a4b2cf0c8a945f98499
9000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Step -5 Changing the decrypted raw vaa into wrong vaa

Changing the chain id

0000000000000000000000000000000000000000000000000000000000000000c350
==0000000000000000000000000000000000000000000000000000000000000004e20

Wrong Vaa with changed amount

01000000000100b1dc88678129d6d86c75046bc2c2f18aee19205644256751215585 3a95f501d51b
a4feeb4bdeac5929081f2f5b35a50266fe301b8cc003e40235518e1f68a3d6016751a6a700000000
000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
0028007769746864726177000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce39
51784b44d0000000000002712434354505553444300000000000000000000000000000000000000
000000000000000000000000000271200000000000000000000000000000000000000000000000
0000000000000000004e200082b4da6b031517fc60458d74e5b3ad3bf7ae29e2ced0a4b2cf0c8a94
5f984999000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Step -6 Encrypting wrong vaa with changed chain id

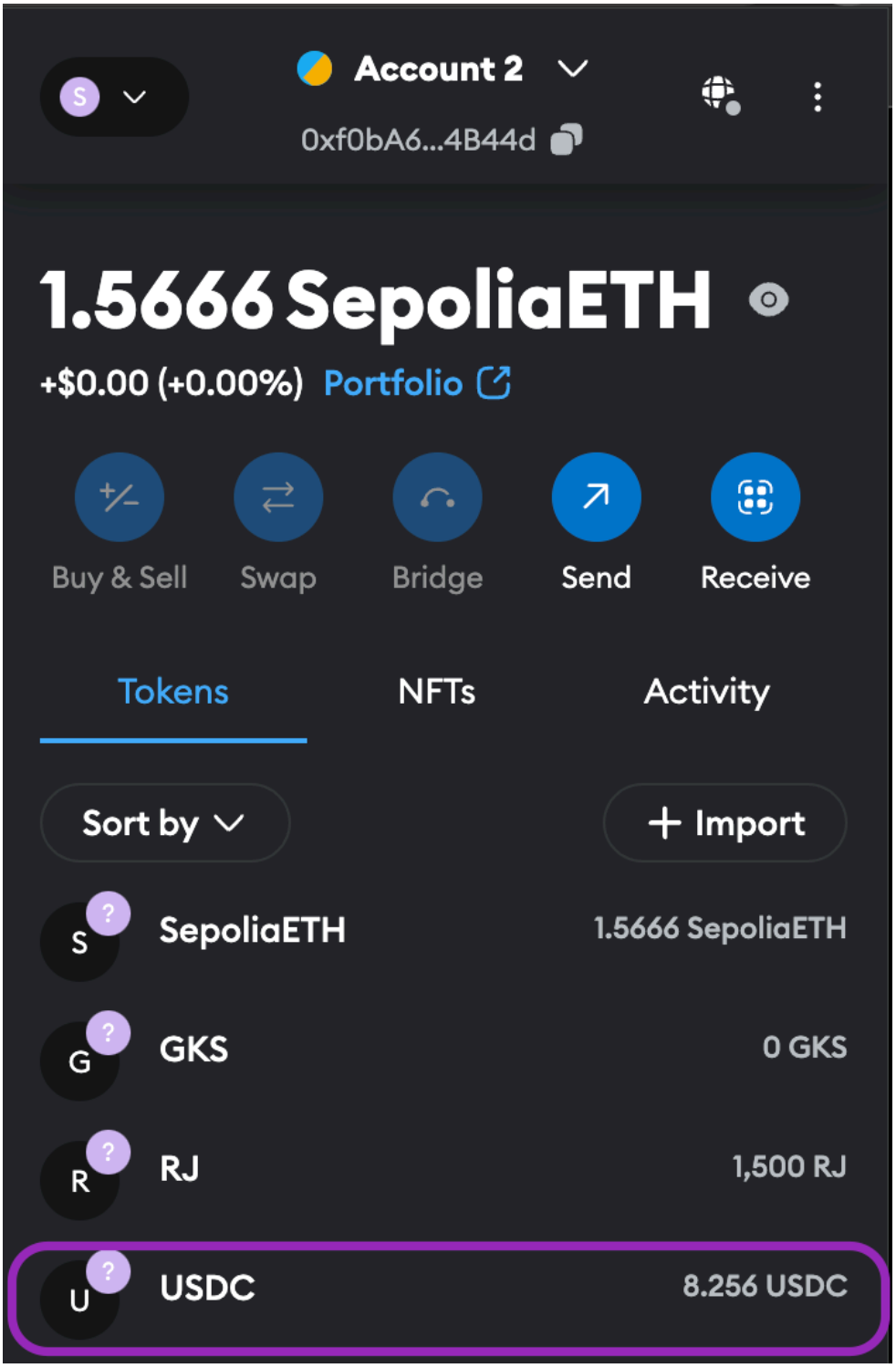
Encrypted wrong vaa

AQAAAAABALHciGeBKdbYbHUEa8LC8YruGSBWRCVnUSFVhTqV9QHVG6T+60verFkpCB8vWzWl
Amb+MBuMwAPkAjVRjh9oo9YBZ1GmpwAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY
/9o/sCLAAAAAAAAACgAd2l0aGRyYXcAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQA
AAAAACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE4gAIK02msDFRf8YEWNdOWzrTv3riniztCkss
8MipRfmEmZAAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

Step -7 Feeding the wrong raw vaa to the receiveEncodeMsg

```
const wrongData =
"01000000000100b1dc88678129d6d86c75046bc2c2f18aee19205644256751215585 3a95f501d51ba4feeb4bdeac5929081f2f5b35a50266fe301
b8cc003e40235518e1f68a3d6016751a6a700000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b00000000
0000000280077697468647261770000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44d0000000000002712434354505553
3444300000000000000000000000000000000000000000000000000000000000000000000000000000271200000000000000000000000000000000000000000000000000000
000000000004e200082b4da6b031517fc60458d74e5b3ad3bf7ae29e2ced0a4b2cf0c8a945f984999000000000000000000000000098ed3e6543f57
79d210135c319ff8dc2f1ed97b300"
const wrData =  Buffer.from(wrongData,"hex");
// console.log("This is wrong data",wrData)
// console.log(Buffer.from(wrData).toString("hex"),"This is the wrong data decoded message from algorand vaa")
const rx = await tokenManager
  .receiveEncodedMsg(
    wrData
  )
  .then(utils.wait)
// Vaa == signature
console.log((await utils.readCCTPDomain(10002)).toString());
```

```
    error: Error: processing response error (body="{\"jsonrpc\":\"2.0\",\"id\":49,\"error\":{\"code\":3,\"message\":\"executio
n reverted: VM signature invalid\",\"data\":\"0x08c379a0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000014564d207369676e617475726520696e76616c696400000000000000000000000000\"}}
\n", error={"code":3,"data":"0x08c379a0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000014564d207369676e617475726520696e76616c696400000000000000000000000000\"}, requestBody="{\"m
ethod\":\"eth_estimateGas\",\"params\":[{\"type\":\"0x2\",\"maxFeePerGas\":\"0x1307e123ae\",\"maxPriorityFeePerGas\":\"0x59682
f00\",\"from\":\"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d\",\"to\":\"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3\",\"data\":\
"0x496d76ff0000000000000000000000000000000000000000000000000000000000
```



Step -9 Entering the correct raw vaa to see if the balance gets updated

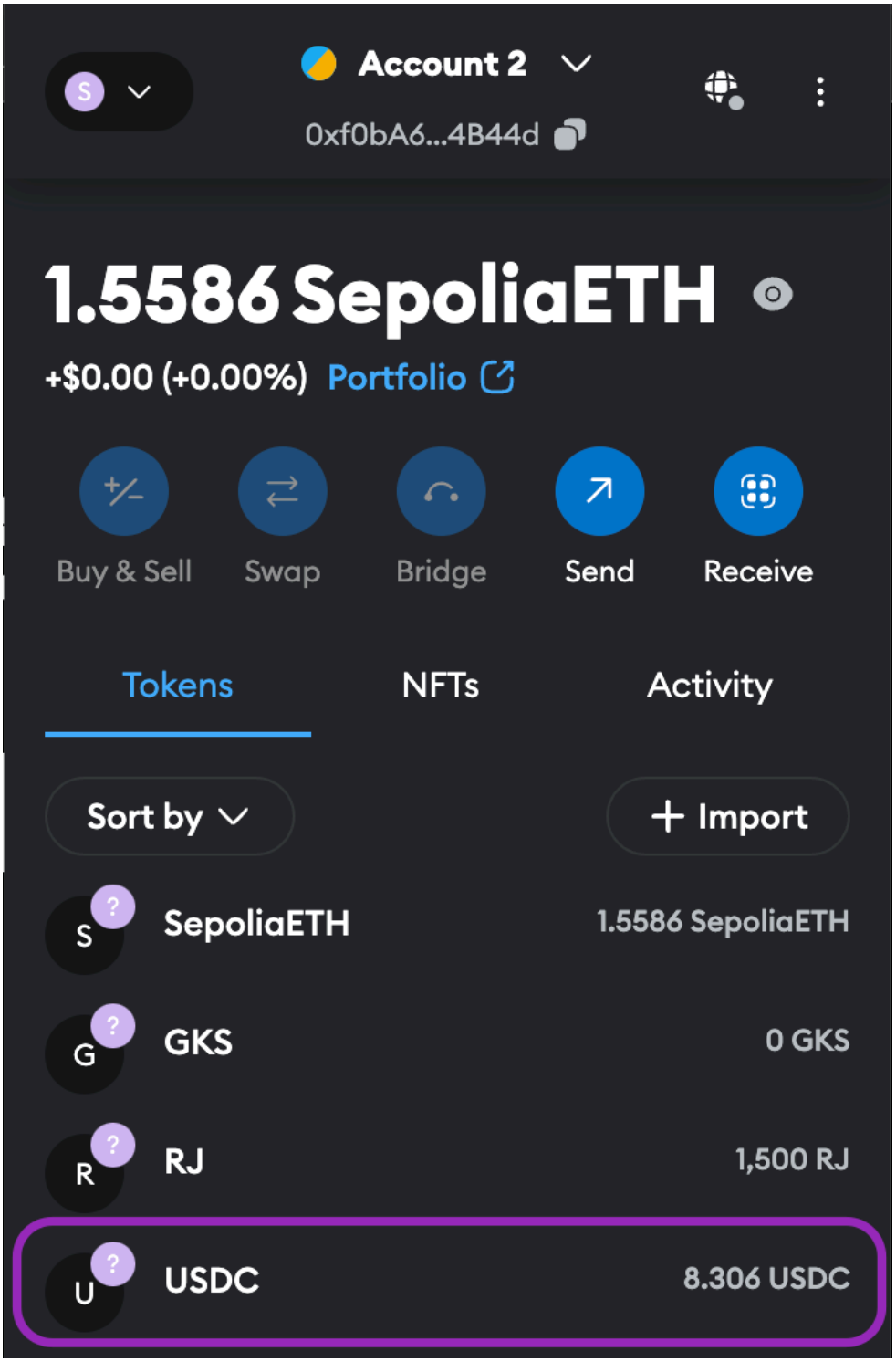
<u>Entering correct VAA</u>

AQAAAAABALHciGeBKdbYbHUEa8LC8YruGSBWRCVnUSFVhTqV9QHVG6T+60verFkpCB8vWzWI
Amb+MBuMwAPkAjVRjh9oo9YBZ1GmpwAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY
/9o/sCLAAAAAAAAACgAd2I0aGRyYXcAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQA
AAAAAACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAIK02msDFRf8YEWNdOWzrTv3riniztCkss
8MipRfmEmZAAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtI7MA

```
gurkiratsingh@Gurkirats-MacBook-Air evm % npm run receiveEncodedMsg

> TokenManager@1.0.0 receiveEncodedMsg
> ts-node ./ts-scripts/main.ts --receiveEncodedMsg

_proxy {
  '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
  '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
  '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
  '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
  '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
}
_proxy {
  '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
  '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
  '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
  '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
  '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
  '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
}
0
```

[This is a Sepolia **Testnet** transaction only]

Transaction Hash:	0x71aec0c4e2a524558b38503b3d478ddc81718045d862019755ddf278b6529b07
Status:	Success
Block:	7216686 4 Block Confirmations
Timestamp:	53 secs ago (Dec-05-2024 01:38:36 PM UTC)
Transaction Action:	Call Receive Encoded Msg Function by 0xf0bA633D...51784B44d on 0x98Ed3E65...2f1ed97b3
From:	0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d
Interacted With (To):	0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3
ERC-20 Tokens Transferred:	All Transfers Net Transfers From 0x98Ed3E65...2f1ed97b3 To 0xf0bA633D...51784B44d For 0.05 ERC-20: USDC (USDC)
Value:	0 ETH
Transaction Fee:	0.00803161973908806 ETH
Gas Price:	49.946330892 Gwei (0.000000049946330892 ETH)



Final balance in algorand of user1
0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\n`@' 32
userBalance 680000
```

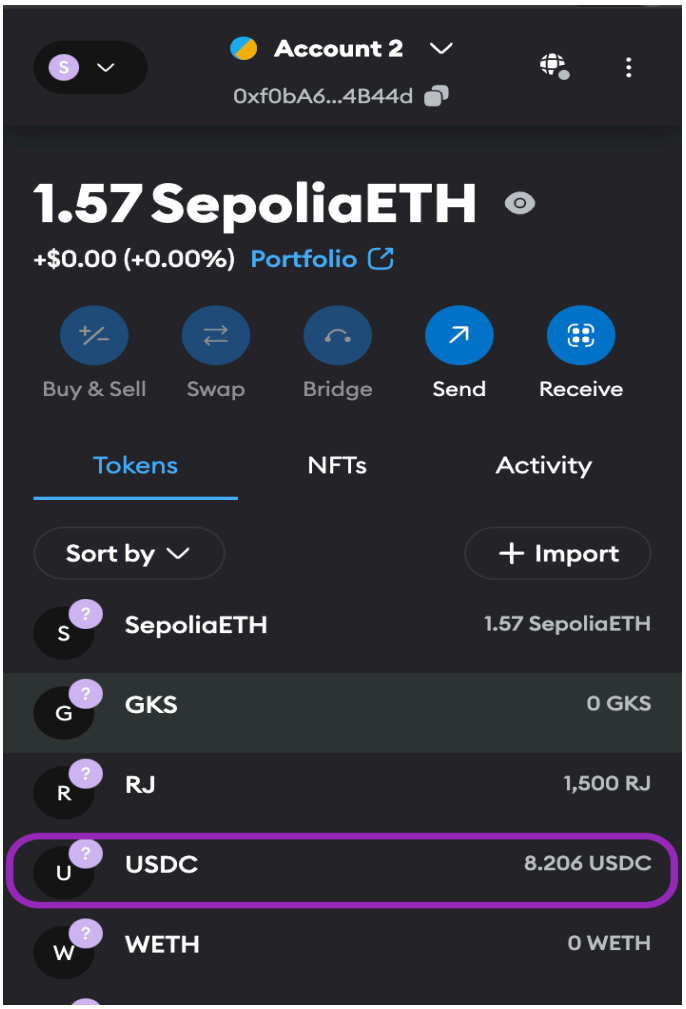
4.9 Manipulating ChainId in VAA during Withdraw

Result	Pass
Description	This process tests the robustness of a cross-chain token withdrawal mechanism from Codex to EVM, particularly in scenarios involving Wormhole VAA manipulation. The aim is to verify whether a maliciously altered VAA (with incorrect chain id) can bypass verification and affect user balances, ensuring system resilience against such exploits.
Function	Withdraw
Steps Performed	1. **Check Initial Balances:** - **EOA Wallet Address**: 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d. - Record the initial USDC balances of the wallet and Codex before initiating withdrawal. 2. **Execute Withdrawal** - Execute the withdrawal transaction on the Codex side. - Collect the resulting VAA for further testing. 3. **Correct VAA Collection:** - Retrieve the generated VAA from Wormhole scan. 4. **Decrypt the Correct VAA** - Decode the raw VAA into a readable format for inspection. 5. **Manipulate VAA:** - **Scenario 1:** Change the chain ID in **one location** within the VAA payload. - Change 0000000000002712 to 0000000000002714. - **Scenario 2:** Change the chain ID in **two locations** to simulate a more complex manipulation. 6. **Encrypt and Test Altered VAA:** - Re-encrypt the manipulated VAA using the provided encryption script. - Feed the modified encrypted VAA to the receiveEncodedMsg function and validate balances. - No changes to user balances, ensuring verification works correctly. 7. **Re-test with Correct VAA:** - Re-submit the original, unaltered VAA to verify the withdrawal transaction proceeds as intended. - Confirm that the transaction succeeds and the user's balance updates as expected. 8. **Final Validation:** - Ensure the manipulated VAA fails and does not alter balances of user. - User's balance (0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d) increases as expected.

Observation

Step -1 Check for both the balances

EOA wallet usdc balance of the user - 0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d



Initial balance in algorand of the user1 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d

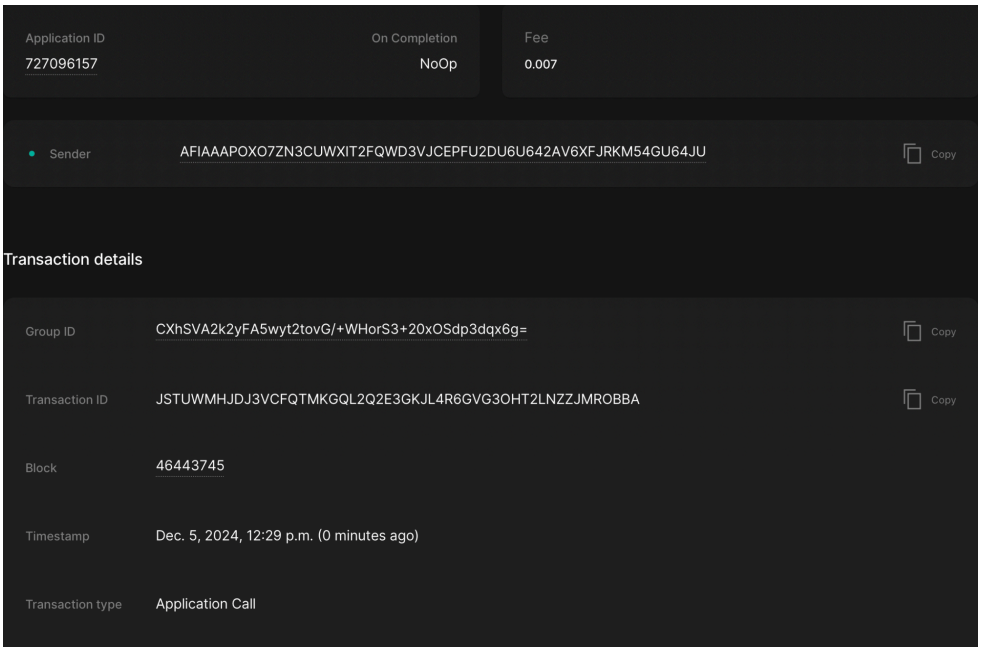
```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/run.py
login chain id 10002
login private key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
login address 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
assest Chain id 65537
assest address 4343545055534443000000000000000000000000000000000000000000000000
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0b\xe6
\xe0' 32
userBalance 780000
```

Step -2 Run Withdraw Function

```
(venv) gurkiratsingh@Gurkirats-MacBook-Air codex % python3 scripts/runWithdraw.py
AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU
727096157
before algo amount of codex 2000000
codex min balance 409100
before algo amount of msg processor 2000000
10002 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d
dest_private_key 80a92e95c5aea441ce86fe5ece9d6f6db366e80919a8ed1cb1167e908e75087d
4343545055534443000000000000000000000000000000000000000000000000 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
emitterId from 727096228 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b
sig_addr 86525623 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b FOVXINPB2TH3CULASIBM4QTF2CKPLESZXE6WOSWRIQL
KVNGXKQ4OCKMU5Y
login 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d 10002
super AFIAAAPOXO7ZN3CUWXIT2FQWD3VJCEPFU2DU6U642AV6XFJRKM54GU64JU 10002
236
10002
b'\x19Ethereum Signed Message:\n549{"Login_Address": "0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d", "Login_ChainId": "10002", "
Token_Address": "4343545055534443000000000000000000000000000000000000000000000000", "Receiver_Address": "0xf0ba633d8c8b2efea8b
988fe3b4ce3951784b44d", "Receiver_ChainId": "10002", "Amount": "50000", "Withdraw_Fee": "False"}AAAAAAAAAAAAAAAAA8LpjPYyLLv6ouY
j+O0zjlReEtE0AAAAAAAAAnEkNDVFBVU0RDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8LpjPYyLLv6ouYj+O0zjlReEtE0AAAAAAAAnEgAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
userBalance 730000
after algo amount of codex 2000000
after algo amount of msg processor 2000000
```

Step -3 Collecting txn id from algorand scan

Txn hash -
JSTUWMHJDJ3VCFQTMKGQL2Q2E3GKJL4R6GVG3OHT2LNZZJMROBBA



Wormhole scan link
https://wormholescan.io/#/tx/JSTUWMHJDJ3VCFQTMKGQL2Q2E3GKJL4R6GVG3OHT2LNZZJMROBBA?network=Testnet

Step -4 Decrypting correct raw Vaa

Correct Vaa
AQAAAAABABLae9ImiS53Icr24liymBN9vv29Eayv+mxQ5OPh2RjveDjqg7C4JcV8a1ZL7OWLAG9d1SZ4902z//6dIYt1OABZ1GcxgAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAACcAd2I0aGRyYXcAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQAAAAAACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAEynSzDpGndRFhNijQXqGibMpK+R8aptuPPS25ylkXBCAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

Decrypted raw vaa
0100000000010012da7bd226892e7721caf6e258b298137dbefdbd11acaffa6c50e4e3e1d918ef7838ea83b0b825c57c6b566f2fb3962c01bd775499e3dd36cffffa74862dd4e00167519cc6000000000008 11fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc6df0e63ff68fec08b0000000000000027007 76974686472617700000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b44 d000000000000271243435450555344430000000000000000000000000000000000000000000 000000000000000000027120000000000000000000000000000000000000000000000000000 0000000c350004ca74b30e91a77511613628d05ea1a26cca4af91f1aa6db8f3d2db9ca59170420000 000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Payload
76974686472617700000000000000000000000000f0ba633d8c8b2efea8b988fe3b4ce3951784b4 4d000000000000271243435450555344430000000000000000000000000000000000000000000 000000000000000000027120000000000000000000000000000000000000000000000000000 00000000c350004ca74b30e91a77511613628d05ea1a26cca4af91f1aa6db8f3d2db9ca5917042000 000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

1. Scenario — (Changing the chain id in one place)

Step -5 Replacing decrypted raw vaa with wrong vaa
Changing the Chain id
0000000000002712 == 0000000000002714

Wrong Vaa with changed chain id
0100000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2
c92f21e3d1f1da2bb29bd7e813fbd47a4f799e16ddabb45ea89db8d5d2c60167515e0000000000000
811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000026
007769746864726177000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a702
86d0000000000002714434354505553444300000000000000000000000000000000000000000
000000000000000000000271200000000000000000000000000000000000000000000000000000
000000000271000883255 48e5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb0
000000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300

Step -6 Encrypting the wrong vaa with changed chain id

Encrypted wrong vaa
vaaAQAAAAABAEXqdNXPMXm+pP+iBh7NtaK3uBjWmiZuP/v312N4ZyE9MLLJLyHj0fHaK7Kb1+gT+
9R6T3meFt2rtF6onbjV0sYBZ1FeAAAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sC
LAAAAAAAAACYAd2I0aGRyYXcAAAAAAAAAAAAAAABvfM92aB2oDL161LKGI6KIGnAobQAAAAAA
ACcUQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACcQAIgyVUjlV39Xp6UsK43zV/1AyeYT+OOIWSk2Mm6
hsWXrAAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

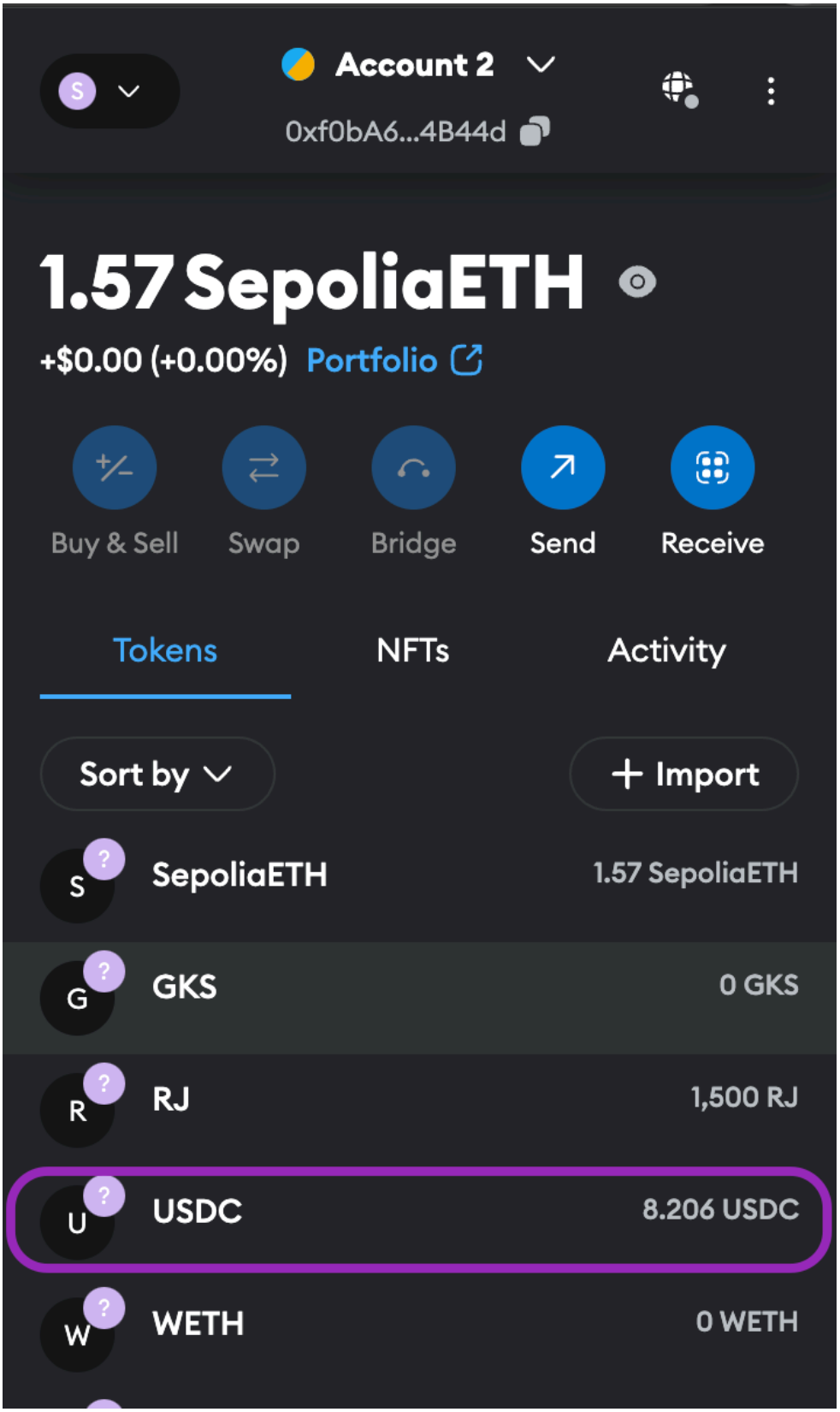
Step -7 Feeding the wrong raw vaa to the receiveEncodeMsg

```
const wrongData =
"0100000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a4f7
99e16ddabb45ea89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000
0000000260077697468647261770000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d0000000000002714434354505 55
3444300000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000027100088325548e5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb0000000000000000000000000098ed3e6543f57
79d210135c319ff8dc2f1ed97b300"
const wrData =  Buffer.from(wrongData,"hex");
// console.log("This is wrong data",wrData)
// console.log(Buffer.from(wrData).toString("hex"),"This is the wrong data decoded message from algorand vaa")
const rx = await tokenManager
  .receiveEncodedMsg(
    wrData
  )
  .then(utils.wait)
// Vaa == signature
console.log((await utils.readCCTPDomain(10002)).toString());
```

```
    error: Error: processing response error (body="{\"jsonrpc\":\"2.0\",\"id\":49,\"error\":{\"code\":3,\"message\":\"executio
n reverted: VM signature invalid\",\"data\":\"0x08c379a0000000000000000000000000000000000000000000000000000000000000000020000000
0000000000000000000000000000000000000000000000000000014564d207369676e617475726520696e76616c696400000000000000000000000000\"}}
\n", error={"code":3,"data":"0x08c379a0000000000000000000000000000000000000000000000000000000000000000020000000000000000000000
0000000000000000000000000000000000000000000014564d207369676e617475726520696e76616c696400000000000000000000000000"}, requestBody="{\"m
ethod\":\"eth_estimateGas\",\"params\":[{\"type\":\"0x2\",\"maxFeePerGas\":\"0x977584cd6\",\"maxPriorityFeePerGas\":\"0x59682f
00\",\"from\":\"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d\",\"to\":\"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3\",\"data\":\"
0x496d76ff0000000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000
0000000013501000000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a
4f799e16ddabb45ea89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000
0026007769746864726177000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d0000000000002714434354505553444300000000
0000000000000000000000000000000000000000000000000000000027120000000000000000000000000000000000000000000000000000000000000000027100088832
5548e5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb0000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b300
0000000000000000000000\"}],\"id\":49,\"jsonrpc\":\"2.0\"}", requestMethod="POST", url="https://ethereum-sepolia-rpc.publicnode
.com", code=SERVER_ERROR, version=web/5.7.1)
        at Logger.makeError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/c
ontracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:269:28)
        at Logger.throwError (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/
contracts/chains/evm/node_modules/@ethersproject/logger/src.ts/index.ts:281:20)
        at /Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/chains/ev
m/node_modules/@ethersproject/web/src.ts/index.ts:341:28
        at step (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contracts/cha
ins/evm/node_modules/@ethersproject/web/lib/index.js:33:23)
        at Object.next (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contra
cts/chains/evm/node_modules/@ethersproject/web/lib/index.js:14:53)
        at fulfilled (/Users/gurkiratsingh/Downloads/CODEX-dev_e623676a0a6e8d69337019137d0f6dc1b8e4d1a2/CODEX-dev/dev/contract
s/chains/evm/node_modules/@ethersproject/web/lib/index.js:5:58)
        at processTicksAndRejections (node:internal/process/task_queues:95:5) {
      reason: 'processing response error',
      code: 'SERVER_ERROR',
      body: '{"jsonrpc":"2.0","id":49,"error":{"code":3,"message":"execution reverted: VM signature invalid","data":"0x08c379a
0000000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000
014564d207369676e617475726520696e76616c696400000000000000000000000000"}}\n',
      error: [Error],
      requestBody: '{"method":"eth_estimateGas","params":[{"type":"0x2","maxFeePerGas":"0x977584cd6","maxPriorityFeePerGas":"0
x59682f00","from":"0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d","to":"0x98ed3e6543f5779d210135c319ff8dc2f1ed97b3","data":"0x496
d76ff0000000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000
0000135010000000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a4f799
e16ddabb45ea89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b000000000000000260
0776974686472617700000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d0000000000002714434354505553444300000000000000
0000000000000000000000000000000000000000000000000000000027120000000000000000000000000000000000000000000000000000000000000000027100088325548e
5577f57a7a52c2b8df357fd40c9e613f8e388592936326ea1b165eb0000000000000000000000098ed3e6543f5779d210135c319ff8dc2f1ed97b30000000
0000000000000000"}],"id":49,"jsonrpc":"2.0"}',
```

Step -8 Final balance of the user

Final Balance of user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is not updated as expected



2. Scenario — (Changing the chain id in both the places)

Step -5 Changing the decrypted raw vaa into wrong vaa in both the places

Changing the Chain id

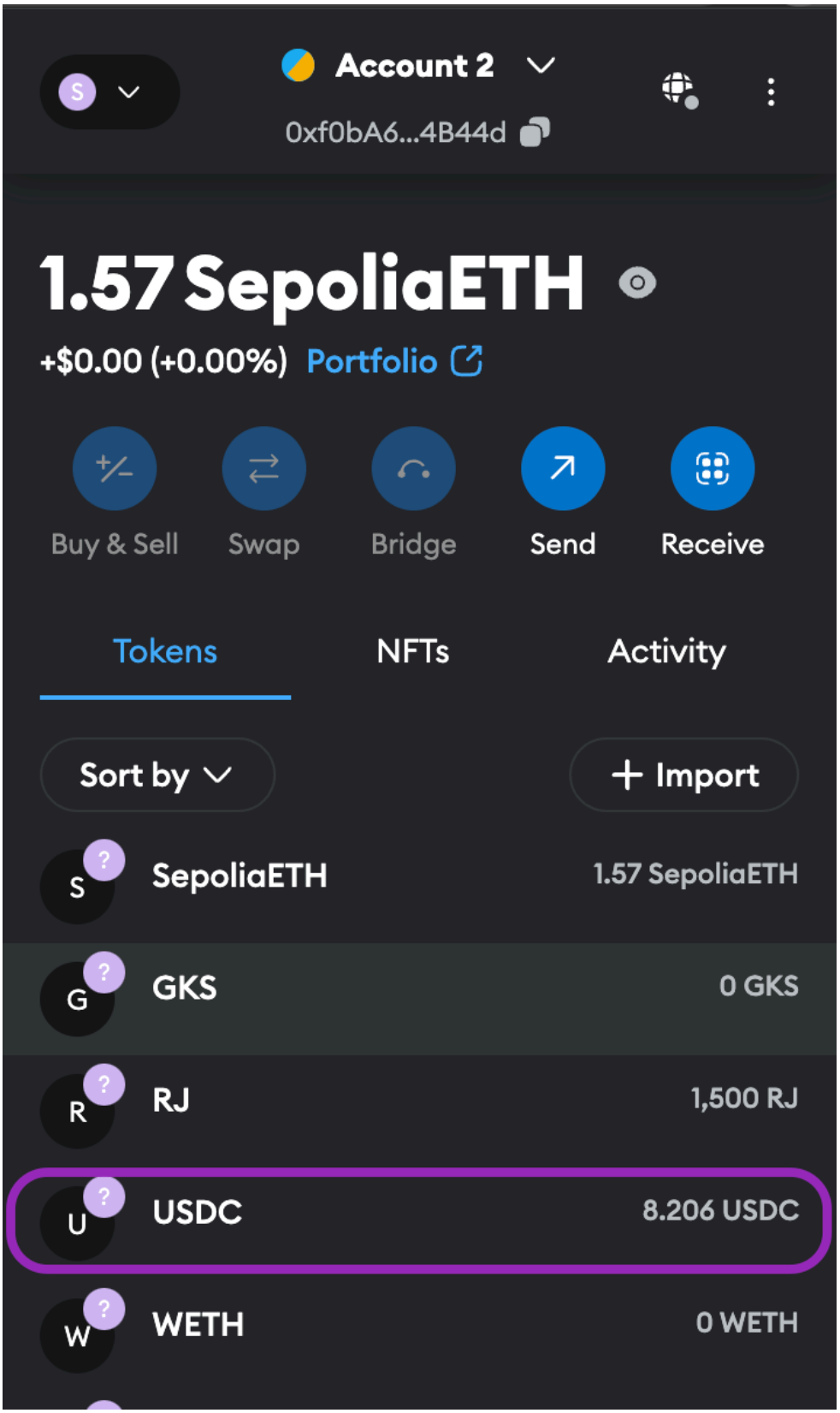
0000000000002712 == 0000000000002714

Wrong Vaa with changed chain id

0100000000010045ea74d5cf3179bea4ffa2061ecdb5a2b7b818d69a266e3ffbf7d7637867213d30b2c92f21e3d1f1da2bb29bd7e813fbd47a4f799e16ddabb45ea89db8d5d2c60167515e0000000000000811fcc2bbdbb39ce1d500ac068b234eccf21bb0d8f421cc69df0e63ff68fec08b0000000000000026007769746864726177000000000000000000000006f7ccf76681da80cbd7ad4b28697a2881a70286d0000000000000271443435450555344430000000000000000000000000000000000000000000000000000000000000000000000271400000000000000000000000000000000000000000000000000000000000

Step -8 Final balance of the user see if the bal gets uploaded

Final Balance of user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is not updated as expected



Step -9 Entering correct raw vaa to see if the balance gets updated

<u>Entering correct VAA</u>

AQAAAAABABLae9ImiS53Icr24liymBN9vv29Eayv+mxQ5OPh2RjveDjqg7C4JcV8a1ZvL7OWLAG9d1SZ4902z//6dIYt1OABZ1GcxgAAAAAACBH8wrvbs5zh1QCsBosjTszyG7DY9CHMad8OY/9o/sCLAAAAAAAACcAd2I0aGRyYXcAAAAAAAAAAAAAADwumM9jIsu/qi5iP47TOOVF4S0TQAAAAAACcSQ0NUUFVTREMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAnEgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMNQAEynSzDpGndRFhNijQXqGibMpK+R8aptuPPS25ylkXBCAAAAAAAAAAAAAAAmO0+ZUP1d50hATXDGf+NwvHtl7MA

```
● gurkiratsingh@Gurkirats-MacBook-Air evm % npm run receiveEncodedMsg

  > TokenManager@1.0.0 receiveEncodedMsg
  > ts-node ./ts-scripts/main.ts --receiveEncodedMsg

  _proxy {
    '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
    '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
    '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
    '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
    '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
  }
  _proxy {
    '4': '0x718CaC2Ff57123D4328C6F9e49c3A90126Da8c69',
    '6': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10002': '0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3',
    '10003': '0x3FAFc9cd92b57eb3b2C8Abd80aF38a440C2a6d4e',
    '10004': '0x938a9036a984E23f30e2861799358A919Dc77e62',
    '10005': '0xC18F0D7153DE49C3841DC32D29F20C384396839d',
    '10007': '0x99737Ec4B815d816c49A385943baf0380e75c0Ac'
  }
  0
```

[This is a Sepolia **Testnet** transaction only]

Transaction Hash:	0x4b34365b6e1ec829fbc857c01a602e7da0a46e23bde56030ab39680383c44cf0
Status:	Success
Block:	7216528 3 Block Confirmations
Timestamp:	1 mins ago (Dec-05-2024 01:05:12 PM UTC)
Transaction Action:	Call Receive Encoded Msg Function by 0xf0bA633D...51784B44d on 0x98Ed3E65...2f1ed97b3
From:	0xf0bA633D8C8b2EfEA8b988fE3B4Ce3951784B44d
Interacted With (To):	0x98Ed3E6543F5779D210135C319fF8dc2f1ed97b3
ERC-20 Tokens Transferred:	All Transfers Net Transfers From 0x98Ed3E65...2f1ed97b3 To 0xf0bA633D...51784B44d For 0.05 ERC-20: USDC (USDC)
Value:	0 ETH
Transaction Fee:	0.00339921988154832 ETH
Gas Price:	21.138769824 Gwei (0.000000021138769824 ETH)

Final balance of the user 0xf0ba633d8c8b2efea8b988fe3b4ce3951784b44d is updated and incremented

7. Auditing Approach and Methodologies Applied

The Deposit and withdrawal transactions were tested in a comprehensive approach to ensure the highest level of security and reliability. Careful attention was given to the following key areas to ensure the overall quality of code:

- **Code quality and structure**: We conducted a detailed review of the codebase to identify any potential issues related to code structure, readability, and maintainability. This included analyzing the overall architecture of the solidity smart contract and reviewing the code to ensure it follows best practices and coding standards.
- **Security vulnerabilities**: Our team used manual techniques to identify any potential security vulnerabilities that could be exploited by attackers. This involved a thorough analysis of the code to identify any potential weaknesses, such as buffer overflows, injection vulnerabilities, Signatures, and deprecated functions.
- **Documentation and comments**: Our team reviewed the code documentation and comments to ensure they accurately describe the code's intended behavior and logic. This helps developers to better understand the codebase and make modifications without introducing new issues.
- **Compliance with best practices**: We checked that the code follows best practices and coding standards that are recommended by the solidity community and industry experts. This ensures that the solidity smart contract is secure, reliable, and efficient.

Our audit team followed OWASP and Ethereum(Solidity) community security guidelines for this audit. As a result, we were able to identify potential issues and provide recommendations to improve the smart contract security and performance.

Throughout the audit of the smart contract, our team placed great emphasis on ensuring the overall quality of the code and the use of industry best practices. We meticulously reviewed the codebase to ensure that it was thoroughly documented and that all comments and logic aligned with the intended behavior. Our approach to the audit was comprehensive, methodical, and aimed at ensuring that the smart contract was secure, reliable, and optimized for performance.

8. Limitations on Disclosure and Use of this Report

This report contains information concerning potential details of the Ultrade EVM Project and methods for exploiting them. Entersoft recommends that special precautions be taken to protect the confidentiality of both this document and the information contained herein. Security Assessment is an uncertain process, based on past experiences, currently available information, and known threats. All information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, while Entersoft considers the major security vulnerabilities of the analyzed systems to have been identified, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures. In addition, the analysis set forth herein is based on the technologies and known threats as of the date of this report. As technologies and risks change over time, the vulnerabilities associated with the operation of the Ultrade EVM solidity smart contract described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities will also change. Entersoft makes no undertaking to supplement or update this report based on changed circumstances or facts of which Entersoft becomes aware after the date hereof, absent a specific written agreement to perform the supplemental or updated analysis. This report may recommend that Entersoft use certain software or hardware products manufactured or maintained by other vendors. Entersoft bases these recommendations upon its prior experience with the capabilities of those products. Nonetheless, Entersoft does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended. This report was prepared by Entersoft for the exclusive benefit of Ultrade EVM and is proprietary information. The Non-Disclosure Agreement (NDA) in effect between Entersoft and Ultrade EVM governs the disclosure of this report to all other parties including product vendors and suppliers.