ESG UQÀM

Centre d'intelligence en surveillance des marchés financiers

Les cahiers de recherche du CISMF
CISMF Research Paper Series

Detection and Prevention of Key-Compromise Related Fraudulence in Crypto-assets Through AI-Empowered Smart Contract: A Novel Framework for Asset Protection and Key-leakage Prevention

Zonglun Li [a], Hanqing Zhao[a] & Xue Liu[a]

[a] School of Computer Science, McGill University, Canada

Novembre / November 2023

Detection and Prevention of Key-Compromise Related Fraudulence in Crypto-assets Through AI-Empowered Smart Contract: – A Novel Framework for Asset Protection and Key-leakage Prevention

Zonglun Li, Hanqing Zhao, Xue Liu

School of Computer Science, McGill University

November 2023

Abstract

This paper introduces a framework based on blockchain and smart contracts to address key-leakage vulnerabilities in transaction activities on a blockchain—a growing concern that has led to significant financial losses and eroded trust in blockchain technology. In addition to traditional methods for enhancing blockchain security, this framework is unique in its ability to empower users with the flexibility to allocate assets and define their desired security level by adjusting a 'Trust Threshold,' thereby improving the security of every single transaction. A key advantage of our framework is its potential to reduce gas expenses, making it a cost-effective solution compared to existing security mechanisms in safety-critical trading environments. The paper presents a thorough analysis of the framework, emphasizing its dual focus on enhancing transaction security and operational efficiency within blockchain networks. Empirical evaluations underscore the framework's efficacy, marking a significant advancement in strengthening the stability and reliability of blockchain systems.

1 Introduction

Blockchain technology, characterized by its secure and Byzantine-tolerant consensus achievement protocol, has revolutionized multiple sectors, including finance, supply chain management, and healthcare. Its core features – decentralization, immutability, and transparency – offer substantial benefits. However, the technology also faces certain challenges, particularly in the realm of security. A significant threat to blockchain networks is key-leakage hacking incidents, which have historically led to significant financial losses.

The prevalence and impact of key-leakage hacking events are well-documented.

Notably, the Wintermute hack of 2022 exposed weaknesses in private key management, resulting in a $160 million loss [1]. Similarly, BXH exchange suffered a $139 million loss due to a compromised admin key [2], and the BNB Chain witnessed significant losses in crypto assets because of a private key leak [3]. These events underscore the vulnerability of blockchain technologies and erode public trust.

In response to these security challenges, various protective strategies have been implemented. These include multi-signature technologies such as BitGo [4], the utilization of hardware wallets [5] and the adoption of two-factor authentication systems [6]. A recently proposed protocol, ERC-4337 [7], has been proposed to further bolster crypto wallet security.

Despite these advancements, current solutions have limitations. They often restrict user flexibility in asset allocation, balancing between asset protection and liquidity. Users are also constrained in determining their preferred security level, which affects protection costs, whether in terms of time or gas expenses. ERC-4337, while offering customizable security options, imposes high gas costs, which is a significant drawback [8]. Thus, there is an urgent need for a more comprehensive solution that addresses these concerns while ensuring transaction security and efficiency on the blockchain.

This paper introduces a new framework designed to overcome these challenges. To the best of our knowledge, this is the first framework that empowers users to autonomously allocate their assets, adjust the trust threshold according to their security preference, and substantially reduce gas costs using graph learning techniques. Our framework significantly bolsters the security and efficiency of blockchain transactions, contributing to the overall stability and reliability of blockchain systems.

The effectiveness and efficiency of the proposed framework are thoroughly analyzed and validated through testing and evaluation. Our findings demonstrate that the framework markedly enhances blockchain security, offering greater resilience against key-leakage hacks. Additionally, it facilitates more cost-effective transactions, optimizing gas usage, and thereby rendering the system more economically viable for users.

2 Related Work

The security of blockchain wallets has garnered considerable attention in the research community, leading to the development of various strategies aimed at fortifying wallet security. This section reviews some of the notable advancements in this domain.

Multi-signature (multi-sig) technology is a pivotal development in enhancing wallet security. Multi-sig wallets necessitate multiple authorizations for transaction execution, thus providing a robust defence against unauthorized access. The multi-sig

mechanism is effectively implemented in various wallets, including those by Argent [9], BitGo [4], Gnosis [10], and Parity [11]. The fundamental advantage of this approach is that compromising one key does not grant access to the wallet's contents, as additional keys are required.

Two-factor authentication (2FA) represents another critical security layer for blockchain wallets. This method involves a dual verification process, where the user must provide a one-time code along with the standard password. The code, typically generated by an application like Google Authenticator or a hardware device, varies with each login, thereby significantly reducing the risk of unauthorized wallet access [12, 13].

The ERC-4337 protocol, first proposed by Vitalik Buterin, offers a substantial enhancement to the security and functionality of Ethereum wallets [7]. This proposal introduces an advanced transaction model with pseudo-transaction objects (UserOperation) and specialized actors (bundlers) for transaction processing. A critical feature of ERC-4337 is the customization of transaction verification logic, which significantly mitigates the risk of key-leakage hacking. However, its use incurs higher gas costs, as each transfer requires a contract call, making transactions more expensive compared to traditional Ethereum wallets [8].

Artificial Intelligence (AI) has also emerged as a promising tool in the realm of blockchain security, particularly in identifying illicit nodes. Deep learning models have shown high accuracy in detecting such nodes [14]. The increasing vulnerabilities in decentralized finance and the critical role of robust security measures, including AI applications, are well-documented [15]. Comprehensive surveys have highlighted the security challenges in Ethereum systems and the potential of AI in anomaly detection [16]. The misuse of smart contracts for illicit activities and the necessity for effective detection mechanisms have been explored, with tools like static analysis for Ethereum smart contracts being developed for vulnerability detection [17, 18]. Nevertheless, the field requires further research to enhance the accuracy and efficiency of these AI models.

3 System Architecture

The proposed system, as depicted in Figure 1, comprises three core components: the Identity Provider, the Users, and the Security Server. This tripartite structure collaborates to safeguard user assets within the blockchain environment.

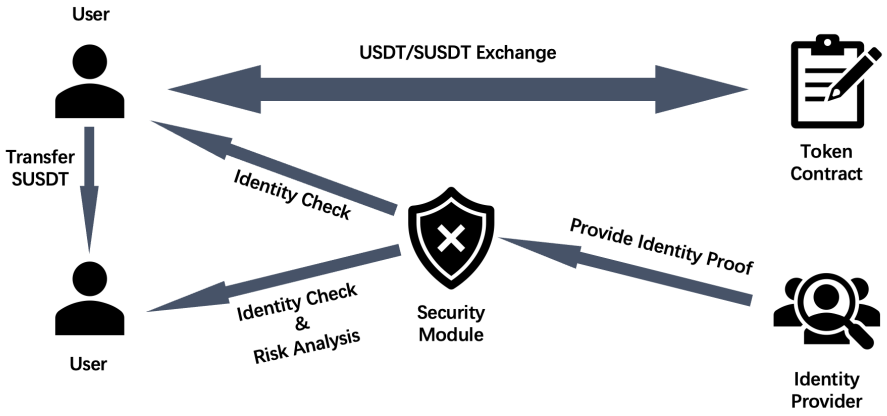

Figure 1: System Architecture of the blockchain wallet security model

3.1 Users and Asset Security

At the forefront of this system are the Users. A user can opt to deposit Tether (USDT) and receive an equivalent amount of Secure USDT (SUSDT) in return. The SUSDT is pegged to the value of USDT, maintaining a constant ratio of $1USDT = 1SUSDT$. Notably, this system is designed to be compatible with a broad range of ERC-20 tokens, ensuring its applicability across various blockchain platforms.

3.2 Identity Provider

The Identity Provider, an off-chain entity, plays a crucial role in maintaining the blockchain identity of the users. Entities such as Google, with a unique public address, act as the Identity Providers. They securely hold the corresponding secret key, thereby ensuring the integrity of the blockchain identity. This off-chain operation allows users to authenticate via conventional methods (e.g., password login or security question login), in conjunction with the public key system. This dual-layer authentication ensures that even if a user's private key is compromised, they can authenticate their identity on-chain through the Identity Provider.

3.3 Security Server

The Security Server functions as an oracle within the system. It utilizes a pre-trained graph learning model, as detailed in Luo's work [14], to provide probabilistic assessments of the maliciousness of blockchain addresses. The Security Server, similar to the Identity Provider, possesses a distinct public address that represents its on-chain identity. When a user intends to transfer SUSDT, the Security Server evaluates the recipient's address. Based on this assessment, the system determines whether additional

identity verification is necessary for the transaction.

3.4 Integration through SUSDT Smart Contract

The integration of these components is facilitated by the SUSDT smart contract. This contract is an extension of the standard ERC-20 token contract, incorporates functionalities for identity checking and inference querying. It serves as the linchpin of the system, coordinating the interactions between the Users, Identity Provider, and Security Server, thereby ensuring a secure and efficient operation of asset transfers within the blockchain network.

4 Preliminaries

This section elucidates the fundamental concepts underpinning our proposed blockchain wallet security model, including digital signatures, the Ethereum-specific cryptographic algorithm, and the ERC-20 token standard. Additionally, we discuss the application of Positive-Unlabeled learning in detecting illicit nodes in blockchain networks.

4.1 Digital Signatures

Digital signatures are integral to the integrity and authenticity of transactions in blockchain networks. They serve the critical purpose of non-repudiation, ensuring that a signer cannot disavow their signature on a document or a transmitted message [19]. In blockchain systems, digital signatures authenticate transactions and safeguard their integrity. Generated using a user's private key and verifiable by the corresponding public key, these signatures bind transactions indelibly to the owner of the private key, preventing post-issuance alterations [20]. Various digital signature schemes, each with distinct advantages and limitations regarding security, efficiency, and applicability, have been implemented in blockchain technologies [19].

4.1.1 Elliptic Curve Digital Signature Algorithm (ECDSA)

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a cornerstone in the security architecture of the Ethereum blockchain. As a variant of the Digital Signature Algorithm (DSA), ECDSA employs elliptic curve cryptography to enhance security while using shorter key lengths. In Ethereum, ECDSA is crucial for generating public and private keys, pivotal for transaction authentication and non-repudiation.

ECDSA's security hinges on the elliptic curve discrete logarithm problem's computational intractability. The algorithm creates a random private key for message

signing and a corresponding public key for signature verification. Ethereum employs the Koblitz curve secp256k1 in its ECDSA implementation, chosen for its resistance to common inversion attacks and its balance between security and performance, a necessity for Ethereum's high transaction throughput [21].

4.2 ERC-20 Standard

The ERC-20 standard represents a protocol for smart contracts on the Ethereum blockchain, specifically for token implementation. It prescribes a uniform set of rules and functions that Ethereum-based tokens must adhere to, facilitating seamless interoperability [22]. Key functions outlined in the standard include token transfer mechanisms, data access protocols, and transfer event notifications [23]. ERC-20's standardization has been pivotal in fostering the growth of Initial Coin Offerings (ICOs) by streamlining the creation of new tokens on the Ethereum network [22].

4.3 PU-Learning in Blockchain Illicit Node Detection

The detection of illicit nodes in blockchain networks is a critical security challenge. Positive-Unlabeled (PU) learning, a machine learning approach tailored for datasets with only the positive class labeled, has shown promise in this regard [14]. This technique is particularly relevant in blockchain contexts where a small subset of nodes is marked as illicit (positive), and the remainder is unlabeled.

A key assumption in PU learning is that unlabeled data comprises only negative examples. However, in blockchain transactions, this assumption may not always hold, as the unlabeled set could contain hidden positives. Consequently, PU learning models in blockchain contexts need evaluation metrics that consider this possibility.

Empirical evidence suggests that PU learning models outperform traditional machine learning techniques in identifying illicit nodes, attributed to their capability to manage hidden positive samples within the unlabeled data. These models are often coupled with graph representation learning, which offers diverse feature distributions for the same dataset, thereby enhancing the accuracy and reliability of detection outcomes [14].

5 Design Details

This section delineates the detailed design of the proposed system, focusing on the *SUSDT* contract, its operational processes, and the universal trust model.

5.1 SUSDT Contract

The *SUSDT* contract is an integral element of our system architecture, encompassing various data structures and procedures crucial for maintaining wallet security. It includes mappings for addresses to authentication information, nonce values, malicious thresholds, and lists of trusted addresses, in addition to storing the public keys of authentication parties and the malicious detector server.

Algorithm 1 SUSDT Contract

```
AuthDic := address ⟶ authCommit          ▷ Mapping from address to auth info commitment
rDic := address ⟶ r                      ▷ Mapping from address to nuance r
tDic := address ⟶ t                      ▷ Mapping from address to malicious threshold t
authPubAddress := [AuthPartyAddress]     ▷ List of authentication party's public key
trustList := address ⟶ [address]         ▷ Mapping from address to trusted addresses
SecureServer := address                  ▷ This is the malicious detector server's PubKey

procedure InitAuth(c, sig, AuthPub, t)
    if Verify(c, sig, AuthPub) == True ⋀ c ∉ AuthDic
⋀ AuthPub ∈ authPubAddress then
        AuthDic[msg.sender] ← c
        rDic[msg.sender] ← 0
        tDic[msg.sender] ← t
    else
        Abort Error

procedure CheckIdentity(sig, AuthPub)
    if Verify(AuthDic[msg.sender] ⊕ rDic[msg.Sender], sig, AuthPub) == True
⋀ AuthPub ∈ authPubAddress then
        rDic[msg.Sender] + +
        return True
    else
        return False

procedure ResetAuth(oldSig, oldAuthPub, c, newSig, newAuthPub)
    if CheckIdentiy(oldSig, oldAuthPub) == True then
        if Verify(c ⊕ rDic[msg.Sender], newSig, newAuthPub) == True ⋀
c ∉ AuthDic ⋀ AuthPub ∈ authPubAddress then
            AuthDic[msg.sender] ← c
            rDic[msg.sender] + +
        else
            Abort Error
    else
        Abort Error

procedure Deposit(amount, sig, AuthPub)
    Assert USDT.BalanceOf(msa.sender) ≥ amount
    Assert AuthPub ∈ authPubAddress
    Assert Verify(AuthDic[msg.sender] ⊕ rDic[msg.sender], sig, AuthPub) == True
    transfer USDT to address(this)
    mint amountSUSDTs to msg.sender

procedure Redeem(amount, sig, AuthPub)
```

```
39:     Assert BalanceOf(msa.sender) ≥ amount
40:     Assert AuthPub ∈ authPubAddress
41:     Assert Verify(AuthDic[msg.sender] ⊕ rDic[msg.sender], sig, AuthPub) == True
42:     burn amountSUSDT
43:     transfer amountUSDT to msg.sender

44: procedure Transfer(to, amount, prob, h, sig′, (sig, AuthPub)) ▷ if prob > t, (sig, AuthPub)
    must be valid
45:     if currentBlockHeight − h > 10 then
46:         Abort Error
47:     if Verify(prob ⊕ h ⊕ rDic[msg.sender], sig′, SecureServer) == True then
48:         if prob ≥ tDict[msg.sender] then
49:             if CheckIdentiy(sig, AuthPub) == True then
50:                 transfer amount to to
51:             else
52:                 Abort Error
53:         else
54:             transfer amount to to
55:     else
56:         Abort Error

57: procedure Trust(address, sig, AuthPub)
58:     if CheckIdentiy(sig, AuthPub) == True then
59:         trustList[msg.sender].append(address)
60:     else
61:         Abort Error

62: procedure Untrust(address, sig, AuthPub)
63:     if CheckIdentiy(sig, AuthPub) == True then
64:         trustList[msg.sender].delete(address)
65:     else
66:         Abort Error

67: procedure TrustTransfer(to, amount)
68:     if to ∈ trustList[msg.sender] then
69:         transfer amount to to
70:     else
71:         AbortError
```

- `InitAuth`: This procedure initializes authentication for a given address. It involves validating the signature of the authentication information and verifying the authorization of the authentication party's public key. Successful verification and the absence of pre-existing authentication information in the dictionary lead to the addition of authentication data and corresponding nonce value and malicious threshold.
- `CheckIdentity`: This procedure is designed to confirm the identity of a sender. It validates the signature resulting from the XOR operation of the sender's authentication information and nonce value. A successful verification leads to an increment in the nonce value and a confirmation of identity.

- `ResetAuth`: This procedure enables a sender to update their authentication information. It begins with verifying the sender's identity using their existing signature and public key. Upon confirmation, it checks the new signature against the XOR of new authentication information and the sender's nonce value. If this check passes, the authentication information is updated, and the nonce value is incremented.
- `Deposit` and `Redeem`: These procedures manage the depositing and redeeming of USDT. They confirm the sender's balance, verify their identity, and then execute the deposit or redemption operation.
- `Transfer`: This procedure facilitates the transfer of a specified amount to a recipient. It includes verifying a signature against the XOR of a probability value, current block height, and sender's nonce value. If the probability value exceeds or equals the sender's malicious threshold, the sender's identity is verified before executing the transfer. If the probability value is lower, the transfer is executed without additional identity checks.
- `Trust` and `Untrust`: These procedures enable a sender to manage their list of trusted addresses. They involve verifying the sender's identity and then adding or removing addresses from their trust list.
- `TrustTransfer`: This procedure allows for the transfer of a specified amount to a recipient who is already on the sender's list of trusted addresses, bypassing the need for further checks.

Each of these procedures plays a distinct role in ensuring the security and functionality of the *SUSDT* contract, thereby contributing to the overall integrity of the blockchain wallet security system.

5.2 Operational Processes

This section outlines the various operational processes within our system, detailing the user interactions that include setup, transfer, deposit, and redemption. These processes are designed not only to ensure security but also to enhance user experience and optimize gas expenditures.

5.2.1 Setup

The setup process is a critical initial step for users intending to secure their cryptocurrency assets, exemplified here using USDT. The process begins with the user creating an account with a reputable identity provider. This is a vital aspect of the system's security mechanism, as depicted in Figure 1.

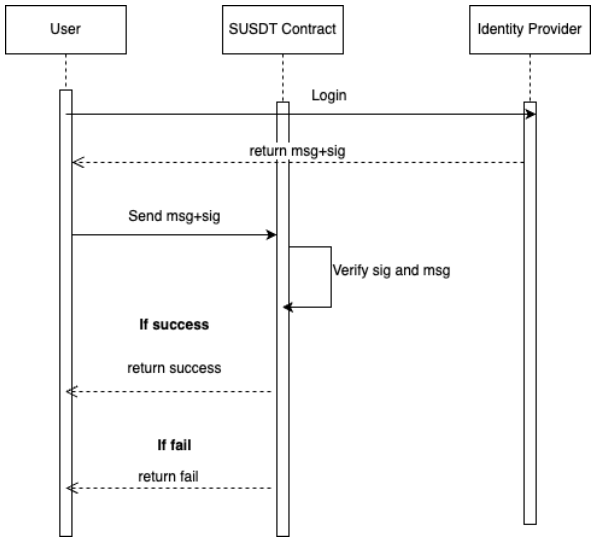

Figure 2: Setup

The user's interaction with the identity provider involves requesting the provider to sign a message denoted as $commit(auth)$, where $auth$ represents the user's account name within the identity provider's system. This step is crucial for establishing the user's identity in a secure manner.

In addition to identity verification, the user is responsible for setting a risk threshold, which can range from 0% to 100%. This threshold is a significant parameter as it aligns with the user's individual risk tolerance and dictates when the system should engage in the identity verification process. The incentive behind this setting is that a rational user would typically avoid transactions with addresses that are deemed high-risk.

Once the user has obtained the necessary information, including the signed message and risk threshold, they proceed to initiate a transaction in the system. This action involves activating the **InitAuth** function within the $SUSDT$ Contract. The user inputs the authentication account commitment c, the signature sig, the identity party's address $AuthPub$, and the risk threshold t into the contract.

The $SUSDT$ Contract then conducts a verification process to ascertain the authenticity of the provided signature. This verification step is critical to ensuring that the user's identity and settings are securely and accurately integrated into the system. If this verification is successful, it signifies the completion of the setup procedure, thereby integrating the user into the system with their specified security settings.

5.2.2 Obtain and Redeem

Obtaining and redeeming $SUSDT$ are crucial for user transactions within our system. Users acquire $SUSDT$ by depositing an equivalent amount of $USDT$ into the $SUSDT$ contract, a process visualized in Figure 3. Conversely, users can convert their $SUSDT$ back into $USDT$ by incinerating an equal amount of $SUSDT$, as outlined in Process 4.

To enhance the security of these transactions, particularly against key-compromised attacks, both depositing (obtain) and redeeming procedures incorporate a mandatory step of identity verification. This process begins with the user conducting a Remote Procedure Call (RPC), a method chosen for its absence of gas charges, to determine their current nonce value, denoted as r. This nonce acts as a dynamic component in the authentication process, ensuring that each transaction is uniquely identified and secured.

Following this, the user approaches the Identity Provider with a request to sign a message. This message is represented as $c \oplus r$, where c is the user's authentication account commitment and r is the nonce. The combination of these elements forms a unique identifier for the transaction, contributing to its security.

Having obtained the signed message, the user is then positioned to initiate the actual transaction within the blockchain network. They trigger either the **Deposit** or the **Redeem** function within the $SUSDT$ Contract. This action involves supplying the function with several key pieces of information: the amount to be deposited or redeemed ($amount$), the obtained signature (sig), and the authentication address ($AuthPub$).

The successful execution of these functions marks the completion of the obtain and redeem processes. This series of steps ensures that the transactions are not only secure against potential vulnerabilities but also user-friendly and efficient in terms of resource utilization within the blockchain network.

5.2.3 Transfer

The transfer process within our system, leveraging the $SUSDT$ token as an ERC-20 token, incorporates a unique and secure method for users to transfer $SUSDT$ to others. This process differs significantly from standard transfer protocols due to its enhanced security measures.

A key aspect of this secure transfer process is the requirement for users to obtain a security analysis of the recipient's address. This is achieved by querying the Malicious Detector Server. Typically, one might consider integrating such a server as an Oracle

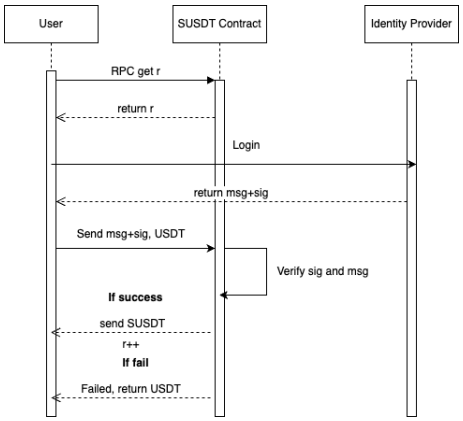

Figure 3: Obtain SUSDT

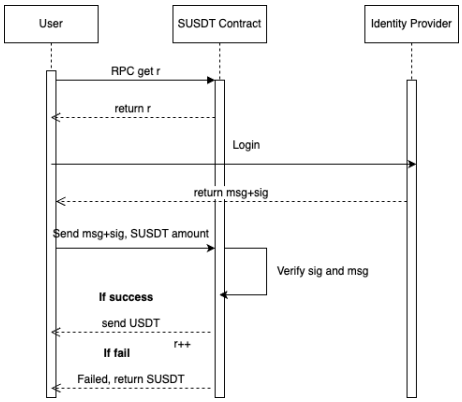

Figure 4: Redeem USDT

within the blockchain system, thereby allowing direct access to off-chain data. However, this approach often entails the need for multiple validators to authenticate the off-chain data, leading to increased operational costs.

To address this challenge and streamline the process, our system adopts an alternative method for server querying. Users initiate a security analysis request directly through an HTTP request to the Malicious Detector Server. The server, upon receiving the request, processes the query and returns the analysis result. This result includes several critical pieces of information: the malicious probability (p) associated with the target address, the current block height (h), and a corresponding signature (sig).

When initiating the transfer transaction, users call upon the **Transfer** function within the $SUSDT$ Contract and provide it with the risk analysis data obtained from the server. The contract then undertakes a two-fold verification process. Firstly, it verifies whether the provided block height falls within an acceptable range – specifically, within 10 blocks of the current block height. This check ensures the timeliness and relevance of the risk analysis result. Following this, the contract authenticates the signature of the analysis to confirm its origination from the Malicious Detector Server.

Post-verification, the contract assesses the malicious probability. If this probability falls below the user-defined threshold, indicating that the risk level of the target address is within the user's acceptable limits, the transaction is permitted to proceed. On the other hand, if the probability exceeds the threshold, suggesting a risk level beyond the user's comfort, the contract temporarily halts the transaction. In such cases, the user is prompted to perform an additional Identity Check, mirroring the process used in the Obtain and Redeem stages. This includes obtaining a signature from the Identity Provider after returning the corresponding nonce (r) to the user. Upon successful identity verification, the contract finalizes and executes the transfer.

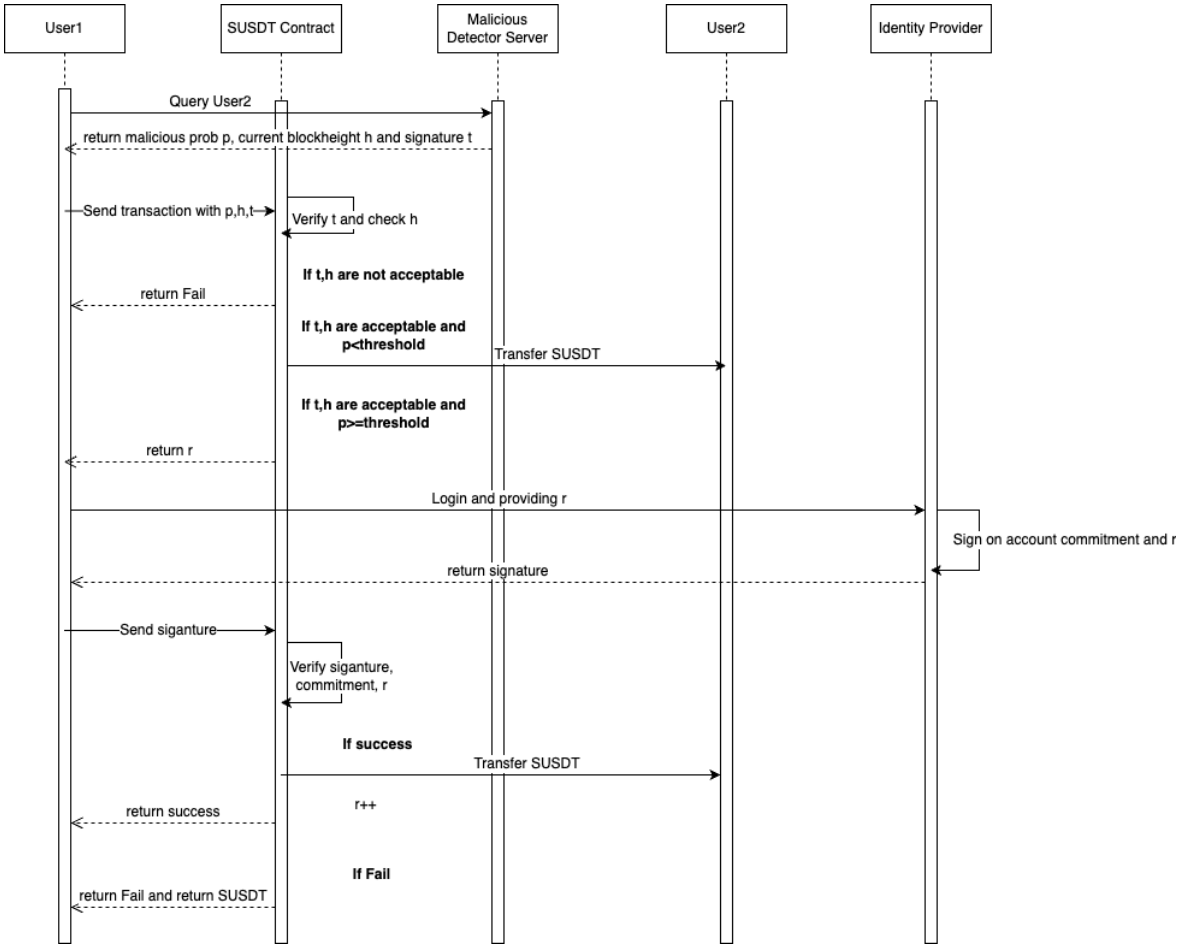

Figure 5: Illustration of the transaction execution within the SUSDT framework

5.2.4 Trusted Transfer

In the realm of blockchain transactions, particularly within the context of our *SUSDT* system, the concept of 'trusted transfers' plays a pivotal role. Users often have a list of addresses that they consider trustworthy. This list may include widely recognized addresses like the Uniswap contract address, or personal addresses belonging to family members and friends. The rationale behind trusted transfers is grounded in both convenience and efficiency, particularly in terms of gas cost optimization.

The trusted transfer process within our system is facilitated by the ability of users to define their own list of trusted addresses. Users create and manage this list through the *SUSDT* contract, utilizing a specific functionality: the **Trust** function. The process begins with the user conducting an Identity Check. This check is essential to ensuring the security and integrity of the trust list.

Upon successful completion of the Identity Check, the user employs the **Trust** function to add a specific address to their trust list. The function verifies the user's identity and, once confirmed, the specified address is added to the user's list of trusted entities.

A significant advantage of this approach is the streamlined transfer process for trusted addresses. When a user wishes to transfer *SUSDT* to an address on their trust list, the system foregoes the usual rigorous checks, such as the additional malicious check. This exemption is based on the premise that addresses on the trust list have already been vetted and deemed secure by the user. As a result, transfers to these addresses are executed without necessitating further verification, information, or proof. This not only simplifies the transaction process but also reduces gas costs, as fewer computational steps are involved in executing these transfers.

5.3 Universal Trust Model

In addition to the *SUSDT* framework, our system incorporates a novel paradigm for asset protection using the concept of Soul-Bound Tokens (SBT)[24]. SBTs represent a unique class of nontransferable NFTs, meaning once a user acquires an SBT, it cannot be transferred to another user. This intrinsic property of SBTs is leveraged to construct a Trust Network among users in our ecosystem.

The Trust Network operates through the issuance of Trusted Tokens, which are a specific type of SBT token provided by the Malicious Detector Server. Each Trusted Token comprises a message and a signature, both authenticated by the server. The message encapsulates several critical elements: the user's blockchain address, an expiration date, and a nonce—a unique number used once to ensure the freshness of the

token.

To obtain a Trusted Token, a user initiates a request to the Malicious Detector Server. This request can be formatted as a blockchain transaction or as an off-chain communication, depending on the user's preference and the context of the interaction. Upon receiving the request, the server performs an analysis (inference) of the user's address. If the inference result falls below a predetermined threshold—indicating a low risk associated with the user's address—the server issues and sends the Trusted Token to the user.

This model significantly enhances the security and trustworthiness of transactions between users. For instance, if a user, Alice, intends to send assets to another user, Bob, she can first verify whether Bob possesses a valid Trusted Token. The validity of the token is determined by two factors: the correctness of the signature and the token's adherence to its expiration date. If Bob's token meets these criteria, Alice can confidently proceed with the transaction, assured of the legitimacy and trustworthiness of Bob's address.

While the Universal Trust Model with SBTs provides an added layer of security and trust in transactions, it is important to note that this model does not offer protection against key theft attacks. The security it provides is primarily in the domain of trust verification between parties, rather than direct asset protection.

6 Evaluation and Discussion

This section presents the results of our experimental evaluation of the SUSDT system and offers a detailed discussion of its security and privacy aspects.

6.1 Experiment

Our experimental setup involved implementing the SUSDT contract using Solidity on the Ethereum Virtual Machine (EVM). We opted for the Elliptic Curve Digital Signature Algorithm (ECDSA) for signature verification, leveraging the EVM's optimized processing of ECDSA, which resulted in reduced gas consumption compared to other signature schemes.

To showcase the economic efficiency of the SUSDT system, we conducted a series of simulated user activities and recorded the gas consumption for each type of activity. The results, detailed in Table 1, highlight the gas usage across various activities within our system.

In comparison with the ERC-4337 protocol, our SUSDT system shows a marked

Table 1: Gas consumption for different activities

Activities	Gas Used	Signature Verification Instances
initAuth	75877	1
deposit	77018	1
redeem	67539	1
trust	63619	1
trustTransfer	39462	0
transfer (without identity check)	58408	1
transfer (with identity check)	63644	2

improvement in gas efficiency. For instance, a high-risk ERC-20 transfer under the ERC-4337 protocol typically consumes around $90,000$ gas [8]. In contrast, the maximum gas requirement for a SUSDT transfer, including identity and Oracle verification, is approximately $63,000$. Moreover, not all transfers necessitate an identity check, as many addresses are deemed safe by our malicious detector server.

6.2 Security & Privacy

Our security analysis is based on several assumptions. We assume the reliability and integrity of Identity Providers, implying they are secure against hacking and always provide accurate signatures. Similarly, we trust in the honesty and security of the malicious detector server, assuming it is immune to manipulation and always signs messages truthfully. Additionally, we presume that users are rational and avoid transactions with known malicious addresses, with the understanding that transactions with such addresses likely indicate a compromised private key.

In our system, users undergoing deposit or redemption processes must pass an identity check, which involves acquiring a signature from the identity provider. Due to the unforgeability of the digital signature algorithm, forging a valid signature without the correct secret key is implausible. Thus, even if a user's private key is compromised, unauthorized access to the identity provider service is effectively barred under our security model.

Moreover, our system effectively thwarts signature replay attacks by incorporating a unique nonce for each user. This nonce increments with each successful identity check, rendering previous signatures invalid for future transactions.

Regarding privacy, the SUSDT system is designed to keep users' authentication accounts confidential. Users only need to provide the on-chain commitment of their authentication account. The commitment's hiding property ensures that no other

parties can deduce the user's authentication account, thus maintaining user privacy throughout their engagement with the system.

7 Conclusion

This paper has introduced a novel framework aimed at bolstering the security of blockchain wallets, addressing some of the most pressing challenges in the field. Central to our approach is the empowerment of users to have greater control over their assets, offering them the flexibility to tailor their security measures according to their individual needs.

Key features of our framework include the ability for users to freely allocate their assets, the customization of their security level, and the opportunity to achieve significant savings in gas costs. These features collectively overcome many of the limitations inherent in existing blockchain security solutions, presenting a more holistic and user-centric approach.

The results from our comprehensive testing and evaluation underscore the framework's robustness in enhancing both the security and efficiency of blockchain transactions. Through rigorous experiments, we have demonstrated the framework's potential in mitigating key-leakage hacking incidents, a prevalent threat in the blockchain domain.

Overall, our framework not only offers an advanced solution to existing security challenges but also lays the groundwork for future research and development in blockchain security. Its implications extend beyond immediate security enhancements, potentially contributing to the broader stability and reliability of blockchain systems.

Acknowledgement

We would like to express our deepest gratitude and appreciation to the Financial Market Surveillance Intelligence Centre at Université du Québec à Montréal (UQAM), funded by the UQAM Ecole des Sciences de la Gestion (UQAM ESG). Their generous financial support has been the cornerstone of this research, providing us with the means to delve into and expand our knowledge in the realm of financial market surveillance.

Specifically, we would like to thank Prof. Ben-Abdallah Ramzi wholeheartedly for his strong support of the project. We also would like to thank our colleagues and peers for their insightful feedback and constructive suggestions, which have greatly enriched this paper.

References

[1] D. Andersen, "Well-known vulnerability in private keys likely exploited in $160m wintermute hack," Sep 2022. [Online]. Available: https://cointelegraph.com/news/well-known-vulnerability-in-private-keys-likely-exploited-in-160m-wintermute-hack

[2] E. Gkritsi, "$139m bxh exchange hack was the result of leaked admin key," May 2023. [Online]. Available: https://www.coindesk.com/tech/2021/11/01/139m-bxh-exchange-hack-was-the-result-of-leaked-admin-key/

[3] S. Sharma, "Top 11 defi cross-chain bridge attacks of 2022: Hackers bag over $2 billion," Oct 2022. [Online]. Available: https://beincrypto.com/top-11-defi-cross-chain-bridge-attacks-of-2022-hackers-bag-over-2-billion/

[4] BitGo, "Eth-multisig-v2/contracts/walletsimple.sol at master · bitgo/eth-multisig-v2." [Online]. Available: https://github.com/BitGo/eth-multisig-v2/blob/master/contracts/WalletSimple.sol

[5] A. G. Khan, A. H. Zahid, M. Hussain, and U. Riaz, "Security of cryptocurrency using hardware wallet and qr code," in *2019 International Conference on Innovative Computing (ICIC)*. IEEE, 2019, pp. 1–10.

[6] T. B. Team, "How to secure your crypto wallet [2023]: Bitpay," Feb 2023. [Online]. Available: https://bitpay.com/blog/how-to-secure-your-crypto-wallet/

[7] V. Buterin and Y. Weiss, "Erc-4337: Account abstraction using alt mempool [draft]," Sep 2021. [Online]. Available: https://eips.ethereum.org/EIPS/eip-4337

[8] J. Rising, "How expensive are erc-4337 gas fees?" 2023. [Online]. Available: https://www.stackup.sh/blog/how-much-more-expensive-is-erc-4337

[9] Argentlabs, "Argentlabs/argent-contracts: Smart contracts for argent wallet." [Online]. Available: https://github.com/argentlabs/argent-contracts

[10] Gnosis, "Multisigwallet/contracts/multisigwalletwithdailylimit.sol at master · gnosis/multisigwallet." [Online]. Available: https://github.com/gnosis/MultiSigWallet/blob/master/contracts/MultiSigWalletWithDailyLimit.sol

[11] Openethereum, "Parity-ethereum/js/src/contracts at 4d08e7b0aec46443bf26547b17d10cb302672835 · openethereum/parity-ethereum." [Online]. Available: https://github.com/openethereum/parity-ethereum/tree/4d08e7b0aec46443bf26547b17d10cb302672835/js/src/contracts

[12] "What is two-factor authentication (2fa) and why do i need it?" [Online]. Available: https://support.blockchain.com/hc/en-us/articles/4417076933524-What-is-two-factor-authentication-2FA-and-why-do-I-need-it-

[13] "All about: 2fa on crypto.com defi wallet." [Online]. Avail-

able: https://help.crypto.com/en/articles/3824773-all-about-2fa-on-crypto-com-defi-wallet

[14] J. Luo, F. Poursafaei, and X. Liu, "Towards improved illicit node detection with positive-unlabelled learning," 2023.

[15] F. Schär, "Decentralized finance: On blockchain-and smart contract-based financial markets," *FRB of St. Louis Review*, 2021.

[16] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–43, 2020.

[17] A. Juels, A. Kosba, and E. Shi, "The ring of gyges: Investigating the future of criminal smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 283–295.

[18] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "Smartcheck: Static analysis of ethereum smart contracts," in *Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain*, 2018, pp. 9–16.

[19] W. Fang, W. Chen, W. Zhang, J. Pei, W. Gao, and G. Wang, "Digital signature scheme for information non-repudiation in blockchain: a state of the art review," *Eurasip Journal on Wireless Communications and Networking*, 2020.

[20] B. Liu, L. Xiao, J. Long, M. Tang, and O. Hosam, "Secure digital certificate-based data access control scheme in blockchain," *IEEE Access*, 2020.

[21] R. Campbell, "Evaluation of post-quantum distributed ledger cryptography," *Journal of The British Blockchain Association*, vol. 2, no. 1, p. 4, 2019.

[22] G. Fenu, L. Marchesi, M. Marchesi, and R. Tonelli, "The ico phenomenon and its relationships with ethereum smart contract environment," in *2018 IEEE 1st International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, 2018.

[23] M. Fröwis, A. Fuchs, and R. Böhme, "Detecting token systems on ethereum," 2018.

[24] B. Vitalik, "Soulbound." [Online]. Available: https://vitalik.ca/general/2022/01/26/soulbound.html