Enhancing DeFi by Improving ERC-20 Token Security and Addressing Leveraged Token Shortcomings

Mohammadreza Rahimian

A Thesis
In the Concordia Institute for
Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Information and Systems Engineering)
at Concordia University
Montréal, Québec, Canada

February 2025

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Mohammadreza Rahimian**

Entitled: **Enhancing DeFi by Improving ERC-20 Token Security and Addressing Leveraged Token Shortcomings**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Steve Shih, Ph.D.	Chair
Rei Safavi-Naini, Ph.D.	External Examiner
Gengrui Zhang, Ph.D.	Arms-Length Examiner
Amr Youssef, Ph.D.	Examiner
Mohammad Mannan, Ph.D.	Examiner
Jeremy Clark, Ph.D.	Supervisor

Approved by ______________________
Farnoosh Naderkhani, Ph.D., Graduate Program Director (CIISE)

01 March 2025 ______________________
Mourad Debbabi, Ph.D., Dean (GCS)

Abstract

Enhancing DeFi by Improving ERC-20 Token Security and Addressing Leveraged Token Shortcomings

Mohammadreza Rahimian, Ph.D.

Concordia University, 2025

ERC-20 tokens have become widely adopted as tools for representing real-world assets on the blockchain. They function as code, running through smart contracts. However, the development of smart contracts has proven to be error-prone, often leading to security vulnerabilities. This study addresses these issues by systematizing 82 known vulnerabilities and best practices. We then introduce a new ERC-20 implementation, TokenHook, which considers all these security aspects. This improved model outperforms widely used ERC-20 templates in terms of security and reliability.

As the blockchain ecosystem evolves, the rise of leveraged tokens (LVTs) presents additional challenges. They extend the features of ERC-20 by adding decentralized finance (DeFi) functionality. Users can buy and sell LVTs like cryptocurrencies but with amplified returns. However, an analysis of over 1,600 LVTs from 10 issuers reveals critical deficiencies due to the absence of a standardized framework, compromising their return on investment. To protect investors, we introduce LeverEdge, a fully decentralized model designed for deploying LVTs on the blockchain. Unlike existing implementations, LeverEdge operates entirely on-chain, overcoming limitations such as transparency, latency, and gas fees through a hybrid L1-L2 approach. With its security carefully tested, LeverEdge provides a potential reference framework for future decentralized LVT deployments.

This progression, from enhancing the security of ERC-20 with TokenHook to developing LeverEdge as a decentralized LVT, contributes to a more secure, transparent, and decentralized approach to DeFi ecosystem.

Acknowledgments

I would like to extend my gratitude to my advisor, Dr. Jeremy Clark, for his guidance and support throughout my PhD journey. His insight and encouragement in navigating the complexities of research, coupled with his constructive feedback, have been crucial in overcoming challenges and maintaining focus. I am grateful for the numerous opportunities he provided to enhance my research skills, shape my ideas, and refine my methodologies, ensuring that my work contributes meaningfully to the field. His detailed feedback on my writing, assistance with publishing, and access to vital research tools have been invaluable in helping me achieve key academic milestones.

I would also like to express my thanks to my committee members, Dr. Mohammad Mannan and Dr. Amr Youssef for their insightful suggestions and thoughtful questions, which improved the quality and depth of my research. Their expertise and feedback challenged me to think critically and broaden my perspective on my work.

Finally, I want to thank my peers in the Madiba Security Research Group. The productive discussions we shared enriched my experience, and I am grateful for the support I found within this research group.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Decentralized Finance (DeFi) refers to a blockchain-based financial system that operates without intermediaries like banks, where transactions are correctly (under reasonable security assumptions) executed by a peer-to-peer network of validators. In blockchains like Ethereum, anyone can participate in the system at any time as either a user and/or a validator without any registration or permission. For this reason, blockchains are commonly described as permissionless. Thus DeFi is permissionless finance where anyone can offer new services or use the services that have been offered. This does not mean it is legal to use in all cases and jurisdictions, however, without validation of identities or citizenships, any laws or regulations can generally be circumvented without much technical effort.

As one might imagine, an arena of permissionless financial services is likely to offer both innovation and high risk. This dissertation focuses on the security, scalability, risks, and the impacts of DeFi. Our work addresses various subtopics, including smart contract vulnerabilities, ERC-20 token security, and the user risks associated with leveraged tokens (LVTs) as complex financial instruments.

The dissertation theme is centered on protecting users at a technical level, whether they are holding ERC-20 tokens or using LVTs to amplify their short-term investments. Our measures are complementary to any additional protections that might be put into place by regulatory authorities.

1.1 Motivation

1.1.1 Blockchain and DeFi: Adoption and Risk

Blockchain and DeFi are phenomena we study because they are widely used, not because of their inherent virtues or flaws. As of October 2024, the total value locked (TVL) in DeFi has reached approximately \$94.9 billion. This represents a significant increase from earlier in the year, when it was around \$54.2 billion, marking over 75% year-to-date rise [25, 147]. Whether blockchain is viewed as revolutionary or controversial, and whether DeFi is considered liberating or risky, the adoption and utilization of DeFi services across various sectors—from finance to technology—make them important subjects for academic research.

Our research primarily focuses on two aspects of blockchain and DeFi: (i) ERC-20 tokens, which have become integral to DeFi ecosystems (see Figure 1.1), and (ii) leveraged tokens (LVTs) as extension of ERC-20 tokens which provide traders with amplified exposure but come with inherent risks.

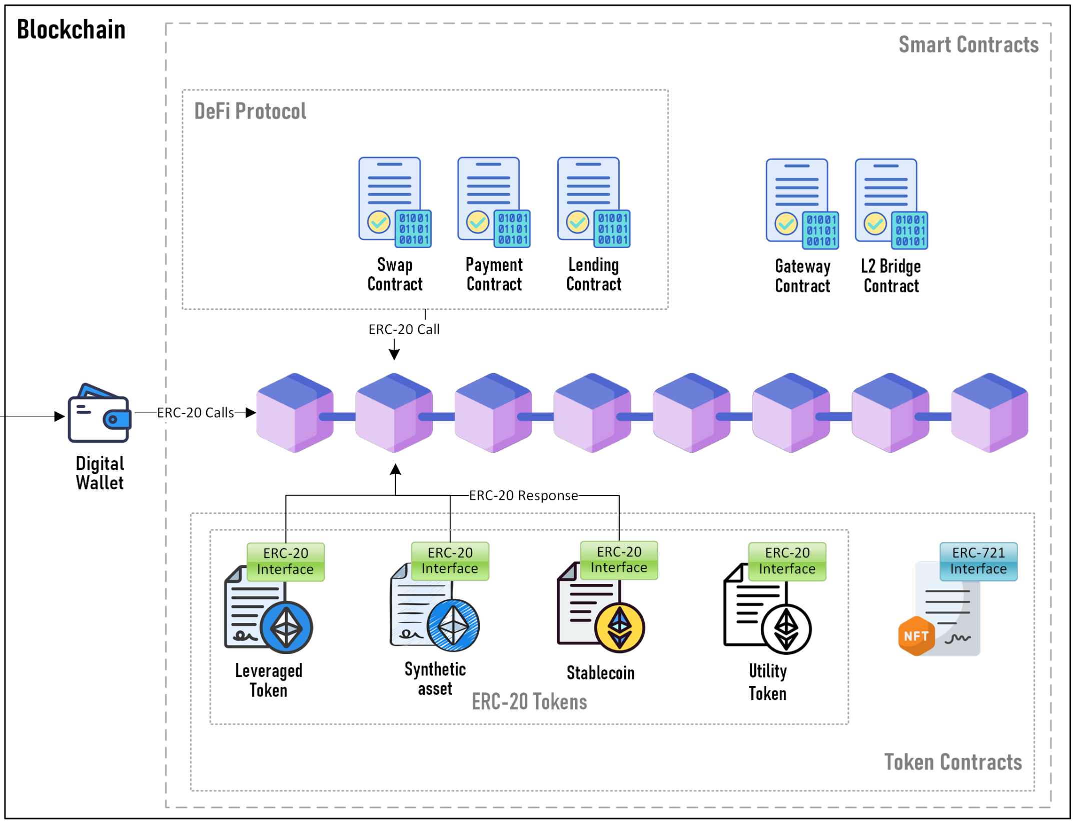


Figure 1.1: Place of ERC-20 tokens and LVTs within the Ethereum and DeFi ecosystem. At the core, there is the Ethereum Blockchain layer, which supports smart contracts. ERC-20 tokens are divided into categories such as leveraged tokens, stablecoins, and utility tokens, connecting to DeFi protocols like lending platforms and decentralized exchanges. Most interactions with fungible tokens follow the interface defined by the ERC-20 standard.

1.1.2 ERC-20 Tokens: A Key Component of DeFi

ERC-20 tokens contribute significantly to the functionality of the Ethereum blockchain and DeFi. While numerous core aspects of blockchain deserve study, ERC-20 tokens stand out for several reasons. They represent the fundamental building blocks of DeFi. Nearly all decentralized applications (dApps) rely on this token standard for asset tokenization, encompassing stablecoins, governance tokens, and utility tokens. As shown in gray in Figure 1.2, majority of Ethereum-based tokens follow this stan-

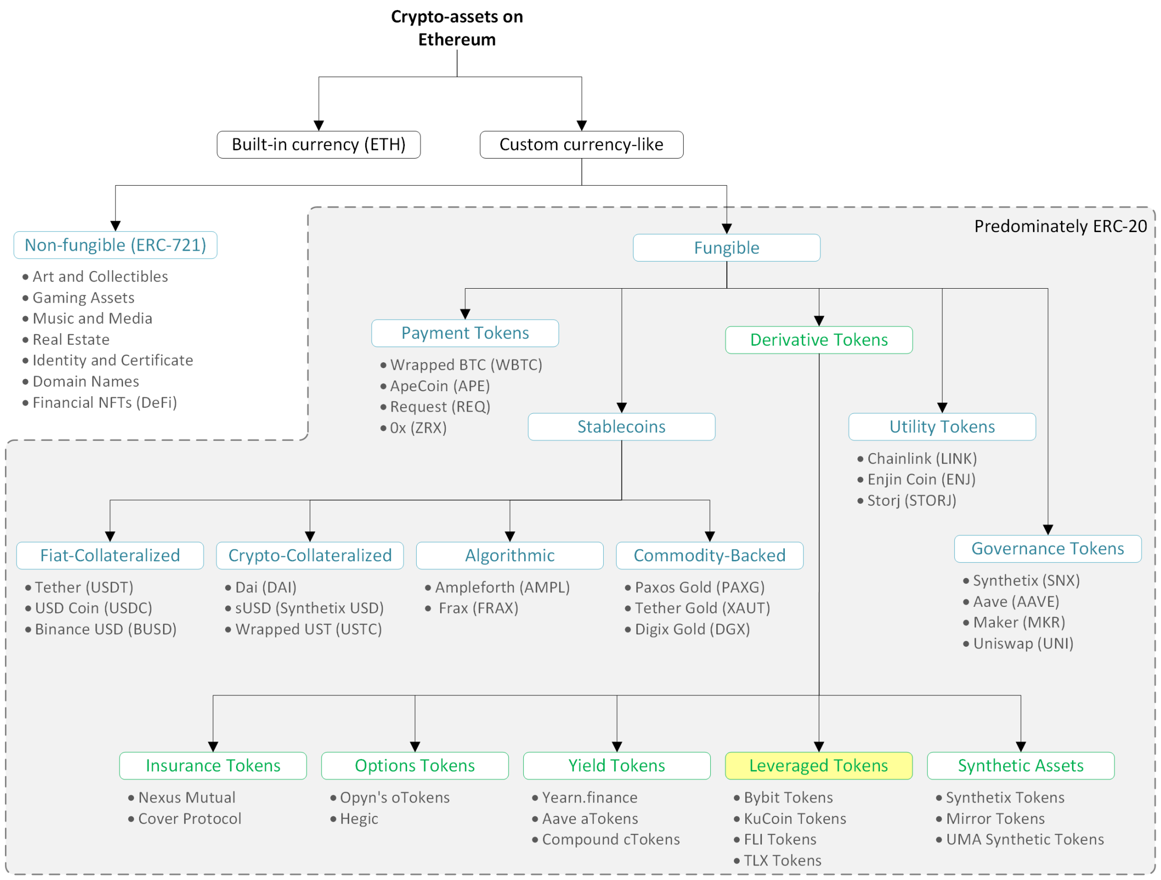


Figure 1.2: Categorization of the different types of tokens based on their functionality, standard protocols, and use cases within the Ethereum ecosystem. More than 98% of fungible tokens follow ERC-20 standard.

dard. Moreover, ERC-20 tokens power key DeFi components such as liquidity pools, lending protocols, decentralized exchanges, and governance systems.

ERC-20 tokens are technically smart contracts that follow a standardized interface, defining a set of rules and functions that each token must implement to ensure compatibility with other smart contracts and dApps. Like any programming code, they are prone to security vulnerabilities. By examining ERC-20 tokens, we gain insights into the core mechanics of decentralized asset representation and movement, which are required for enhancing security of these instruments. We explore the secu-

Risk to Users	**Technical Reason**	**Protection Mechanism**
Inability to custody	Assets are locked in the exchange	Implementation on the blockchain
Transparency in total supply	Total supply are not published	
Transparency in transactions	Transaction are not publicly disclosed	
Transparency in token holders	Ownership is confidential	
Interoperability with DeFi	Token works in isolation	
Challenges in auditing	Code of the token is not public	
Inadequate financial backing	Futures issued after the launch of the token	
Possibility of Front-running	Exploited fund trade during well-known events	Randomized rebalancing or iceberg orders
Higher tracking error	Inefficient rebalancing algorithm	Optimization of rebalancing algorithms
Higher management fees	Inefficient cost management	Cross-trading or implementation on layer 2 chains
Multiple Withdrawal Attack	Withdraw from user wallet more than approved	Securing transferFrom method

Table 1.1: Protection mechanisms for users holding ERC-20 tokens or LVTs

rity challenges associated with ERC-20 tokens to identify potential enhancements and ensure compliance with best practices. This contribution strengthens the security of ERC-20 tokens, reinforcing their pivotal role in DeFi ecosystem.

1.1.3 Leveraged Tokens: Complexity and Investor Safeguards

Unlike ERC-20 tokens, leveraged tokens (LVTs) are not a critical infrastructure component of the blockchain. However, they present a layer of complexity that makes them worth exploring in academic research. Many users are attracted to LVTs due to their potential for amplified returns but often fail to fully grasp the technical details, particularly the risks associated with volatility decay, rebalancing mechanisms, or the impact of market movements in adverse directions. This lack of understanding may lead to common missteps, where users overestimate the gains of these tokens or underestimate their exposure to losses. By researching LVTs, we aim to shed light on these intricacies (see Table 1.1), helping both developers and users better comprehend the mechanisms at play and mitigating the associated risks.

1.2 Contributions and Outline

The structure of this dissertation follows an incremental research approach [182], where the latest contributions are built on earlier findings, proposes solutions, and validates them through comparative analysis and integration. It is organized in the following chapters.

1.2.1 Chapter 2: Background

We provide an overview of key blockchain technologies, including the ERC-20 token standard, security auditing tools, and practices used to assess smart contract vulnerabilities. Additionally, we explore the concept of financial leverage and its application in crypto markets, such as lending platforms and futures markets. This chapter establishes the fundamental principles and protocols necessary for understanding the decentralized design, security considerations, and market dynamics discussed in subsequent chapters.

1.2.2 Chapter 3: Resolving the Multiple Withdrawal Attack on ERC20 Tokens

We focus on Ethereum tokens, particularly ERC-20 tokens, which are integral to dApps deployed on Ethereum and other EVM-compatible chains. The ERC-20 standard is widely used and interoperable with numerous dApps, user interface platforms, and web applications. A key security issue, the *"Multiple Withdrawal Attack"*, has existed since 2016 and involves the `approve` method, allowing malicious users to exploit token approvals. Understanding how the attack works aids in the process of mitigating it. Moreover, evaluating the proposed mitigations helps determine the extent to which they address the attack. If none are satisfactory, a new proposal will be required. In this regard, we answer RQ3 as the primary research question, followed by 5 guiding questions:

(RQ3) Are there any open security vulnerabilities in the ERC-20 protocol that remain unaddressed?

(RQ3.1) How does the *"Multiple Withdrawal Attack"* work, and why is its mitigation important?

(RQ3.2) Where should this attack be prevented, and what does an ideal solution looks like?

(RQ3.3) What efforts have been made to address this vulnerability, and to what extent have they been effective?

(RQ3.4) Can the *"Multiple Withdrawal Attack"* be mitigated while ensuring backward compatibility and adherence to the ERC-20 standard?

(RQ3.5) How would a potential solution impact the token's performance or gas requirements?

Contributions. We evaluate 10 proposed mitigations for the *"Multiple Withdrawal Attack"* and develop a set of criteria that encompass (i) backward compatibility, (ii) DeFi interoperability, (iii) adherence to the ERC-20 standard, and (iv) attack mitigation. Since no mitigation is fully satisfactory, we study in detail possible implementations of ERC-20's `approve` and `transferFrom` methods. We then develop two additional solutions: one deploys a secure `approve` method but does not adhere to the specifications of the ERC-20 standard. The second, mitigates the attack by securing the `transferFrom` method and fully satisfies the ERC-20 standard specifications. This solution requires 37% more gas compared to the non-secure implementation, which we believe is justifiable to protect users investing in ERC-20 tokens.

1.2.3 Chapter 4: TokenHook: Secure ERC-20 Smart Contract

Ethereum has undergone numerous security attacks, collectively causing more than US$100M in financial losses [74, 128, 127, 152, 140, 11]. Drawing from the *"Multiple Withdrawal Attack"* work [143], it seems reasonable to investigate all potential security vulnerabilities affecting ERC-20 tokens and systematize them. Additionally, assessing the effectiveness of popular static analysis tools designed for smart contracts can identify potential inconsistencies and false positives, highlighting areas for improvement. Although prior research has addressed smart contract vulnerabilities [97], there are still concerns on the security of ERC-20 tokens that can be answered by the primary research question RQ4 followed by 3 guiding questions:

(RQ4) How can known ERC-20 security risks be classified to improve overall token security?

(RQ4.1) What are other known vulnerabilities in ERC-20 tokens, and how can they be systematically categorized to improve security practices?

(RQ4.2) Can a new ERC-20 implementation, enhance security and software diversity compared to existing Solidity and Vyper implementations?

(RQ4.3) How effective are widely-used auditing tools in detecting security vulnerabilities in ERC-20 token implementations, and can they replace the need for expert human security reviews?

Contributions. We study all known vulnerabilities and cross-check their relevance to ERC-20 token contracts, systematizing a comprehensive set of 82 distinct vulnerabilities and best practices. We then use our specialized domain knowledge to provide a new ERC-20 implementation, TokenHook, which is open source and freely available in both Vyper and Solidity. It aims to increase software diversity, as currently no Vyper ERC-20 implementation is considered a reference, and only one Solidity implementation is actively maintained. Compared to this implementation, TokenHook offers enhanced security properties and stronger compliance with best practices. Finally, we use TokenHook as a benchmark to assess the completeness and precision of seven widely used auditing tools for detecting security vulnerabilities. We conclude that while these tools offer some value, they cannot replace the expertise of a security professional in developing and reviewing smart contract code.

1.2.4 Chapter 5: A Shortfall in Investor Expectations of Leveraged Tokens

We examine leveraged tokens (LVTs) as emerging crypto-assets primarily issued by centralized exchanges. These tokens are modeled after the concept of leveraged ETFs (LETFs) in traditional markets, offering amplified gains and losses relative to the underlying asset. Since 2019, more than 1,600 leveraged tokens have been introduced. By analyzing key aspects such as underlying assets, blockchain interaction, types of leveraged products, and fund management algorithms, we can formalize LVT dynamics and mechanics of their constituent components. Analyzing these aspects clarifies how the tokens achieve leverage and maintain their structure over time. Additionally, it helps investors to understand the functionality of leveraged funds, rebalancing mechanisms, and potential risks. This is crucial for making informed investment decisions that we address through the primary research question RQ5 followed by 4 guiding questions:

(RQ5) What are Leveraged Tokens (LVT), and what limitations exist in their current operations?

(RQ5.1) What information is accessible to LVT traders, and how transparent are these investment vehicles in terms of their structure and risk exposure?

(RQ5.2) Are LVTs adequately financially backed and able to effectively track their leverage ratios, ensuring they deliver the expected returns to investors?

(RQ5.3) To what extent are LVTs tied to specific exchanges, and what risks, such as front-running, arise from this dependency?

(RQ5.4) How do LVT fees and tracking errors compare to those of traditional LETFs, which are commonly used by investors as a baseline for comparison?

Contributions. We analyze more than 1,600 LVTs from 10 issuers, and show that 99.9% of LVTs are centralized, which implies they are only accessible internally within the ecosystem of the exchange itself. 80% of them do not interact with the blockchain, leading to the lack of transparency in transactions, holders, custody, and auditing. Moreover, 53% of the issuers do not disclose the total supply, making challenging for investors to trade LVTs by their fair market price. The absence of uniform standards in LVT implementation has led to unpredictable technical and financial performance.

Additionally, 41% of LVTs may have been issued without sufficient financial support upon their launch where required future products were launched with delay. About 97% of them are vulnerable to front-running during well-known events. LVTs have higher leverage deviation from the advertised ratio compared to LETFs because of inconsistencies in the management of funds or inefficiencies in rebalancing algorithms. Holding both LETFs and LVTs for extended periods of time impacts the performance, referred to as volatility decay. LVTs also normally have higher management fees than LETFs, which negatively affects the performance of the fund relative to the expected return. Our analysis provides valuable insights for crypto investors, developers, and auditors, offering a framework for understanding LVT mechanics, risks, and market impact.

1.2.5 Chapter 6: LeverEdge: On-Chain Leveraged Tokens

The research literature [142, 104, 166, 164] has shown that LVTs in the crypto market diverge from the expected returns of LETFs, leading to notable deficiencies. Ten deficiencies are identified, largely due to the absence of a standardized implementation framework. This is a motivation for academia to explore solutions aimed at overcoming the limitations of LVTs. This is critical because users purchase LVTs under the assumption that these tokens will merely amplify the return of the underlying asset (ETH, BTC, *etc.*). However, they remain unaware of the internal mechanics of these tokens, such as daily rebalancing and the volatility decay, which can result in significant losses when the token is held for an extended period. Functional aspects of LVTs can be clarified by addressing the primary research question RQ6 followed by 4 guiding questions. It can assist users in better understanding LVTs before making investment decisions.

(RQ6) What solutions have been proposed to improve LVT functionality, and can a decentralized design eliminate existing shortcomings?

(RQ6.1) What is the impact of current deficiencies on the performance of LVTs?

(RQ6.2) What solutions have been proposed or implemented to address these issues, and how successful have they been?

(RQ6.3) Can a new decentralized design like LeverEdge eliminate existing shortcomings, and how does its efficiency compare to current decentralized LVTs on Ethereum?

(RQ6.4) Are there any inherent flaws in LeverEdge that need to be addressed in future work?

Contributions. We review six decentralized LVTs and evaluate the extent to which they address the identified deficiencies. Despite notable attempts, a closer examination of their functionality reveals ongoing shortcomings that still need to be resolved. In response, we propose a fully decentralized design model, LeverEdge, deployed on the Ethereum blockchain. Unlike existing centralized implementations, LeverEdge is entirely on-chain, addressing most of the identified deficiencies. However, this approach introduces challenges related to blockchain limitations, such as latency, scalability, and gas fees. To mitigate these, we developed a new L1-L2 hybrid model, which has passed security checks and can serve as a reference for future decentralized LVT deployments on EVM-compatible chains.

LeverEdge has been evaluated under similar conditions as other LVTs, demonstrating its capability to address the recognized flaws. It employs perpetual futures to generate leveraged exposure and incorporates a cross-chain mechanism for compatibility with various L2 ecosystems. Designed with a focus on composability, it is deployed on Ethereum, and its open-source code has successfully passed security audits, making it a blueprint for developing new decentralized LVTs or transitioning existing centralized versions to decentralized solutions. The ultimate goal is to enhance investor protection by reducing risks associated with centralized control and improving the transparency of LVT operations.

1.2.6 Chapter 7: Concluding Remarks

We summarize the key findings and contributions of our research in this chapter. It highlights the significance of the results, how they were achieved, and offering suggestions for potential directions in future work.

Chapter 2

Background

This chapter covers foundational concepts that will be discussed in the other chapters.

2.1 Blockchain's Expanding Influence

Blockchain is a decentralized, untrusted network wherein participants (nodes) can freely join or leave. There is no trusted central authority as all nodes work together with various consensus protocols, such as Proof of Work or Proof of Stake, to keep the integrity of data. In the case of hardware or communication failure of one node, other nodes continue to process requests seamlessly since each has a complete and consistent copy of data. The main features of blockchain are not restricted to technical resilience only. According to various reputable advisories [162, 146], blockchain has reached an average annual growth of 51% from 2016 to 2022 and surpassing to more than $2 billion USD in revenue by the year 2022. With the scope of blockchain applications to be further developed combined with the projected growth, blockchain will be one of the key technologies in less conservative industries.

The global blockchain market is expected to reach $39.7 billion by 2025 with a annual growth rate of 67.3% [87]. Financial services are leading in adopting blockchain and all banks will be likely considering blockchain solutions in the near future for payment and settlement systems. Meanwhile, DeFi has been developing explosively by breaking the total value locked (TVL) of $200 billion in 2021 [118]. This demonstrates the ever-improving usability of blockchain for lending, borrowing, and trading applications without depending on any intermediate parties.

Moreover, the utility of blockchain is also growing out of finance sector. By 2025, at about 55% of all healthcare applications are expected to have blockchain integrated in securing patient records [108]. Another fastest-adopting industry for blockchain is supply chain management to introduce more transparency and traceability. Large companies such as Walmart and IBM have already implemented blockchain mechanism in tracking products from their origins to retail [161].

2.2 Ethereum's Smart Contract Ecosystem

Out of 22 various implementations of the blockchain [8], Ethereum [186] is more widely accepted by the industry. It is a public blockchain proposed in 2013, deployed in 2015, and has the second largest market cap at the time of writing. It has a large development community which track enhancements and propose new ideas [24].

Ethereum enables decentralized applications to be deployed and executed on top of the blockchain. Smart contracts are essential component of the Ethereum and has been adopted widely by holding millions dollars worth of digital coins in form of ETH, ERC-20 tokens, Digital wallets and DeFi protocols.

Smart contracts are programs which get executed on the blockchain in a decentralized manner. Their conditions of execution lie inherently inside the code, which in return maintains some predefined rules. Smart contracts perform tasks on their own without the interference of any third-party intermediary. This gives more transparency in the contract operations with little risk of manipulation. Within the Ethereum ecosystem, smart contracts are deployed on-chain and, once published, become immutable and tamper-proof. They can be written by developers in any of Ethereum's high-level programming languages, such as Solidity and Vyper. Smart contracts can also handle simple token transfers to most complex dApps. Their programmability enables developers to define various kinds of interactions like token exchange, voting mechanism, or multi-signature wallets. Smart contracts are in fact the backbone of Ethereum functionality.

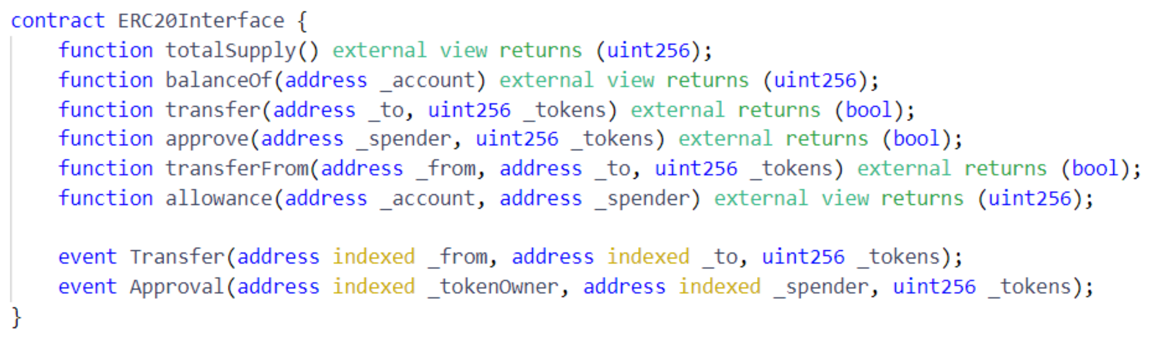
Like other emerging technologies, security is an important aspect of smart contracts. Previous research discovered that at about 45% of existing smart contracts on the Ethereum are vulnerable [115]. The development of smart contracts has been proven to be error-prone, and as a result, smart contracts are often riddled with security vulnerabilities. One of the major smart contract hacks was due to TheDAO bug, which caused a loss of 60 million US dollars in June 2016 [157]. Similar to other programming languages, smart contracts' codes can be exploited by an adversary (*i.e.,* Miner, external users and other contracts) to manipulate executions and gain profit. Some smart contracts have kept millions of dollars, which could be enough to incentivize adversaries to exploit vulnerabilities.

Example 1. Aave is a decentralized lending protocol, with a total value locked (TVL) consistently exceeding \$6 billion. Users can earn interest by supplying assets or take out loans by borrowing crypto with their deposits as collateral. As another example, Uniswap, one of the largest decentralized exchanges (DEX), has a TVL of approximately \$3.7 billion. It enables users to trade directly from their wallets [10].

2.3 Decentralized Apps and ERC-20 Tokens

Migrating applications from centralized to decentralized architecture can solve many issues such as single point of failure, hardware and maintenance costs, and data security. This type of application is distributed over an untrusted network and leverage smart contracts to run codes on the blockchain. Tokens are subset of smart contracts and security is particularly important given that many tokens have considerable market capitalization. As tokens can be held by commercial firms, in addition to individuals, and firms need audited financial statements in certain circumstances, the correctness of the smart contract issuing the tokens is now in the purview of professional auditors.

Ethereum allows dApps to accept and use ETH as its protocol-level cryptocur-



```
contract ERC20Interface {
    function totalSupply() external view returns (uint256);
    function balanceOf(address _account) external view returns (uint256);
    function transfer(address _to, uint256 _tokens) external returns (bool);
    function approve(address _spender, uint256 _tokens) external returns (bool);
    function transferFrom(address _from, address _to, uint256 _tokens) external returns (bool);
    function allowance(address _account, address _spender) external view returns (uint256);

    event Transfer(address indexed _from, address indexed _to, uint256 _tokens);
    event Approval(address indexed _tokenOwner, address indexed _spender, uint256 _tokens);
}
```

Figure 2.1: The ERC-20 interface in Solidity defines a set of functions that any contract implementing the ERC-20 standard must include. These functions ensure interoperability and compatibility with other ERC-20 compliant contracts, wallets, exchanges, and decentralized applications.

rency or issue their own custom tokens with a variety of intents (*e.g.,* In-app purchase, Interoperability with other dApps, Representing digital assets, *etc.*). Tokens might be currencies with different properties than ETH, they may be required for access to a dApp's functionality or they might represent ownership of some off-blockchain asset. It is beneficial to have interoperable tokens with other dApps and off-blockchain webapps, such as exchange services that allow tokens to be traded. In this regard, the Ethereum community accepted a popular token standard called ERC-20 [70]. As shown in Figure 2.1, ERC-20 is an interface that defines abstract methods (name, parameters, return types) and provides guidelines on how the methods should be implemented, however it does not provide an actual concrete implementation. Developers have the flexibility of implementing ERC-20 methods according to the needs of their dApps, or even expand it to offer new functionalities (*e.g.,* leveraged tokens).

While numerous ERC-20 extensions or replacements have been proposed (*e.g.,* ERC-721, ERC-777, ERC-1155, *etc.*), ERC-20 remains prominent. Of the 2.5M smart

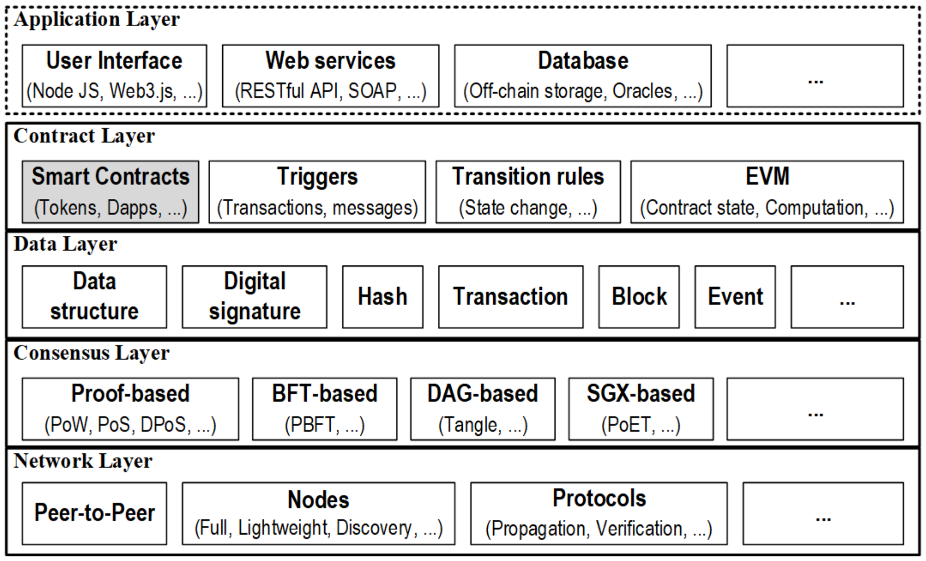


Figure 2.2: Smart contracts generally exist at the contract layer, leveraging the underlying consensus mechanisms for secure execution and state changes. The *Contract Layer* can be considered a sub-layer of the *Application Layer*, focusing on how smart contracts execute and interact with blockchain transactions.

contracts on the Ethereum network, 260K are tokens. 98% of these tokens are ERC-20 tokens [168], demonstrating their widespread acceptance by the industry, smart contract developers and the Ethereum community. As shown in Figure 2.2, among the layers of the Ethereum blockchain, ERC-20 tokens fall under the *Contract layer* in which back-end of dApps are executed.

2.4 ERC-20 Tokens vs. Leveraged Tokens vs. Leveraged ETF

A typical Exchange-Traded Fund (ETF) is a weighted basket of stocks from firms with a common characteristic (*e.g.*, they all operate in a specific sector or have a high market capitalization). The issuer splits the basket into shares, which are bought and sold on exchanges just like individual stocks [112].

Example 2. One of the most traded ETFs is the *SPDR S&P500 ETF* with ticker symbol *SPY*. It is issued by SSGA[1] and holds a basket of stocks from nearly 500 publicly traded companies that are part of the S&P500[2] index. The S&P500 index has globally served as a gauge for the performance of the U.S. stock market as a whole, due to its depth and diversity. Since SPY tracks the S&P500 index, investors can gain broad exposure and diversify their investment risk across the stock performance of 500 companies in 11 sectors without the logistics or starting capital required to buy shares in all these companies.

Leveraged ETFs (LETFs) were introduced in 2006 and are ETFs designed to amplify the daily performance of the underlying basket.[3] Inverse LETFs aim to achieve a return that is a multiple of the inverse of the underlying asset's daily performance [94, 20, 150]. Many investors alternatively refer to LETFs and inverse LETFs as "Bullish" and "Bearish" LETFs, respectively, reflecting their short-term sentiment on future price movements.

Example 3. *Direxion Daily S&P500 Bull 3x ETF (SPXL)* is a 3x (three times) LETF that seeks to deliver triple the daily performance of the S&P500. It magnifies each 1% gain in the S&P500 index into a 3% gain and loses 3% for every 1% drop in the

[1]State Street Bank and Trust Company (SSGA) is one of the three dominant companies in the ETF market, with a 14.01% market share, following BlackRock and Vanguard, which have 33.64% and 29.16%, respectively [163].

[2]The S&P 500 index comprises 500 of the top publicly traded companies in the U.S. It was launched in 1957 by the credit rating agency Standard and Poor's [103].

[3]The underlying asset can be stocks, market indexes (e.g., S&P 500, NASDAQ-100, etc.), commodities (e.g., gold, oil, corn, etc.), or any asset with a price.

index. *Direxion Daily S&P500 Bear 3x ETF (SPXS)* delivers triple the opposite daily performance of the S&P500 index. If the S&P500 index depreciates by 1%, SPXS gains 3%, and vice versa [180, 110].

The equivalent of an LETF in the cryptocurrency and crypto-asset ("crypto") market could be thought of as Leveraged Token (LVT). Similar to LETFs, LVTs use leveraged products offered in the crypto market to outperform the underlying asset's return on a daily basis. While the majority of LETFs are actively managed funds[4], LVTs employ one of three management models: (i) centralized, (ii) decentralized, and (iii) hybrid.

Centralized LVTs are mainly managed by crypto exchanges. They can be purchased on the spot market (similar to cryptocurrency) or directly from the issuer. Decentralized LVTs operate on the blockchain through ERC-20 contracts and can be traded without relying on a third-party. Hybrid LVTs are basically decentralized LVTs that are traded on centralized crypto exchanges. Users prefer centralized exchanges for their user-friendly interfaces, continuous-time order books (rather than automated market makers, which are the only trading mechanism efficient enough to run on-chain), and increased liquidity due to aggregated buy and sell orders.[5] However, this model introduces certain disadvantages resulting from the combination of centralized and decentralized systems (*e.g.,* functional complexities, security concerns, custodial risks, *etc.*).

[4]In actively managed funds, investment managers actively buy and sell assets with the goal of outperforming a specified benchmark index, resulting in higher management fees.

[5]The more liquid an asset is, the easier and more efficient it is to convert back into cash. Less liquid assets take more time and may incur higher costs [92].

Example 4. An issuer may offer BTC3L/BTC3S as a pair of LVTs tracking Bitcoin (BTC) as the underlying asset. A Bitcoin futures contract (BTC-Perp[6]) can be used as the leveraged product to outperform Bitcoin in the short term. The number three in the LVT name represents the multiplier (triple-leveraged), while L/S stands for going long/short on the market.[7] BTC3L gains 3% when the price of Bitcoin rises by 1%, and loses 3% for every 1% price drop. Conversely, when Bitcoin drops by 1%, BTC3S gains 3%, and loses 3% for every 1% price rise.

2.5 Compounding Effect in LVTs

Volatility can be defined as the rate of variation in values of a particular asset. A high volatility asset's value can be spread out over a bigger range of value and may change dramatically either way in a very short period of time. In contrast, the price of a low-volatility asset does not vary much and mostly remains constant. An example could be that, in the equity market, whenever the price of any particular stock has been continuously moved up or down by less than 1%, for some considerable period of time. It is thought to be a volatile stock, especially when it passes this 1% range compared to historical price [93].

Compounding Effect represents the rate at which the value of investments dissipates[8]. As volatility increases and positions are held for a longer period, the drag

[6]A type of Bitcoin futures contract without a defined expiration date (known as a "perpetual").

[7]Going long refers to buying an asset with the expectation that its value will increase, allowing it to be sold for a profit later. Conversely, going short refers to profiting from a decline in the price [105].

[8]Also referred to as *volatility decay* or *volatility erosion*.

Initial Investment	Asset	Leverage		Day 1 5.0%	Day 2 5.0%	Day 3 -10.0%	Day 4 -4.5%	Day 5 7.0%	Day 6 4.0%	Day 7 -5.0%
$200.00	BTC	1	**Value**	$210.00	$220.50	$198.45	$189.52	$202.79	$210.90	$200.35
			PNL ($)	$10.00	$20.50	-$1.55	-$10.48	$2.79	$10.90	$0.35
			PNL (%)	5.00%	10.25%	-0.78%	-5.24%	1.39%	5.45%	0.18%
	BTC3L	3	**Value**	$230.00	$264.50	$185.15	$160.15	$193.79	$217.04	$184.49
			PNL ($)	$30.00	$64.50	-$14.85	-$39.85	-$6.21	$17.04	-$15.51
			PNL (%)	15.00%	32.25%	-7.43%	-19.92%	-3.11%	8.52%	-7.76%
	BTC3S	-3	**Value**	$170.00	$144.50	$187.85	$213.21	$168.44	$148.22	$170.46
			PNL ($)	-$30.00	-$55.50	-$12.15	$13.21	-$31.56	-$51.78	-$29.54
			PNL (%)	-15.00%	-27.75%	-6.07%	6.60%	-15.78%	-25.89%	-14.77%
	BTC5L	5	**Value**	$250.00	$312.50	$156.25	$121.09	$163.48	$196.17	$147.13
			PNL ($)	$50.00	$112.50	-$43.75	-$78.91	-$36.52	-$3.83	-$52.87
			PNL (%)	25.00%	56.25%	-21.88%	-39.45%	-18.26%	-1.91%	-26.44%
	BTC5S	-5	**Value**	$150.00	$112.50	$168.75	$206.72	$134.37	$107.49	$134.37
			PNL ($)	-$50.00	-$87.50	-$31.25	$6.72	-$65.63	-$92.51	-$65.63
			PNL (%)	-25.00%	-43.75%	-15.63%	3.36%	-32.82%	-46.25%	-32.82%

Table 2.1: Impact of the Compounding Effect on the performance of 3x/5x Bitcoin leveraged tokens during a volatile week. It demonstrates why LVTs are not suitable for long-term investments.

imposed by volatility accelerates, making it more destructive to the value of the investment [171, 134]. A similar effect occurs in LVTs, which impacts the return.

Example 5. The performance of 3x and 5x Long/Short BTC tokens with a $200 initial investment is compared in a volatile market. The assumption in Table 2.1 is a 5% BTC price increase on day 1, followed by another 5% increase on day 2, and then a 10% drop on day 3, continuing in this pattern. Mathematically, two 5% increases on days 1 and 2 should be offset by a 10% drop on day 3. However, the math does not align as expected due to accumulated profit or loss. The price of a non-leveraged BTC position increases by 5% from $200 to $210 on day 1 and by another 5% from $210 to $220.5 on day 2 (a 10.25% gain after 2 days). A 10% drop on day 3 from $220.5 to $198.45 results in a position that is -0.78% lower than the initial investment, making the overall return negative.

This loss is -7.43% and -21.88% of the initial investment for BTC3L and BTC5L,

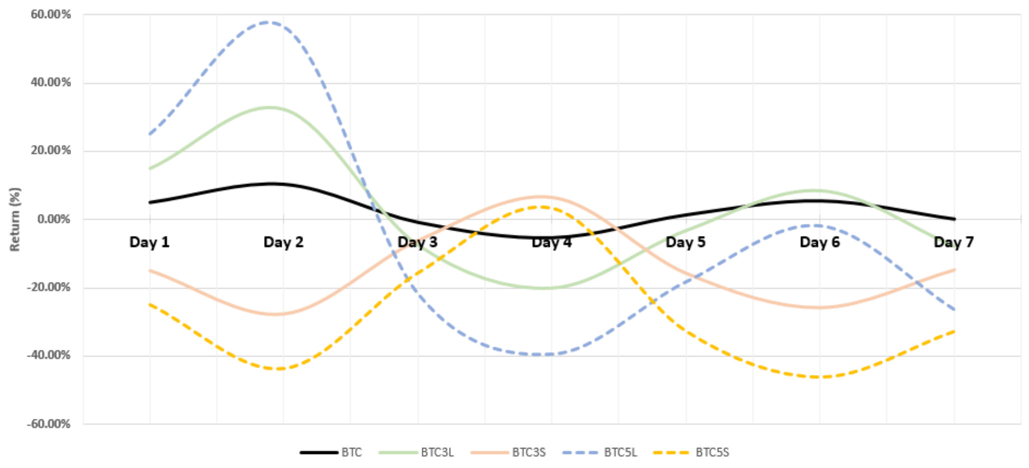


Figure 2.3: LVT returns during a volatile week: the value of the initial investment (BTC in black) reached the breakeven point on day 7, but the returns of all LVTs remained negative due to the impact of Compounding Effect.

respectively. Investors would expect better performance from BTC3S and BTC5S on the short side. However, due to accumulated losses in the first two days, they close at -6.08% and -15.63% lower than the starting value. Even after a week, when the initial investment has nearly reached the break-even point, all LVTs remain negative regardless of their leverage and direction (see Figure 2.3).

As the above example suggests, volatility negatively impacts the performance of LVT investments over time, as previously discussed in the case of LETFs [85, 170]. While volatility can be beneficial in the short term, compounded daily returns produce unexpected mathematical outcomes. Investors would have made some profits if they had closed their positions on the first or second day, but holding the position through the third day would have resulted in a loss. To minimize the impact of Compounding Effect on LVTs, it is better to use them as short-term investments in markets with strong trends and momentum.

It is worth mentioning that the Compounding Effect and Volatility Drag are related but not the same. Volatility drag refers to the reduction in overall returns of a LVT due to the inherent volatility of the underlying asset. Because these tokens reset daily, fluctuations in the asset's price can erode gains over time, especially in sideways markets. Compounding effect refers to how the daily resetting of LVT impacts their long-term returns. Compounding can either magnify gains or amplify losses over time, depending on the sequence of daily returns. This effect can be positive or negative, unlike volatility drag, which is generally detrimental. In summary, while compounding can lead to either higher or lower returns due to the daily reset, volatility drag specifically highlights the adverse impact of market volatility and management fee deductions on LVT returns over time.

2.6 Leverage Products

LVTs derive their value from a leveraged fund, which in turn is based on a leveraged product. This leveraged product usually originates from either the *Crypto derivatives market* or the *DeFi lending market* as highlighted in Figure 2.4. In the *derivatives market*, Perpetual futures (Perps)[9] have become rather popular due to their flexibility. Because Perps do not expire, they can be held forever, with the additional advantage of being able to make leveraged positions in LVTs without any risk of contract expiration or rollover.

[9]Perpetual Futures are also called Perpetual Swaps due to their structural similarity to traditional swap contracts (*i.e.,* Continuous Funding, No Expiry Date, *etc.*).

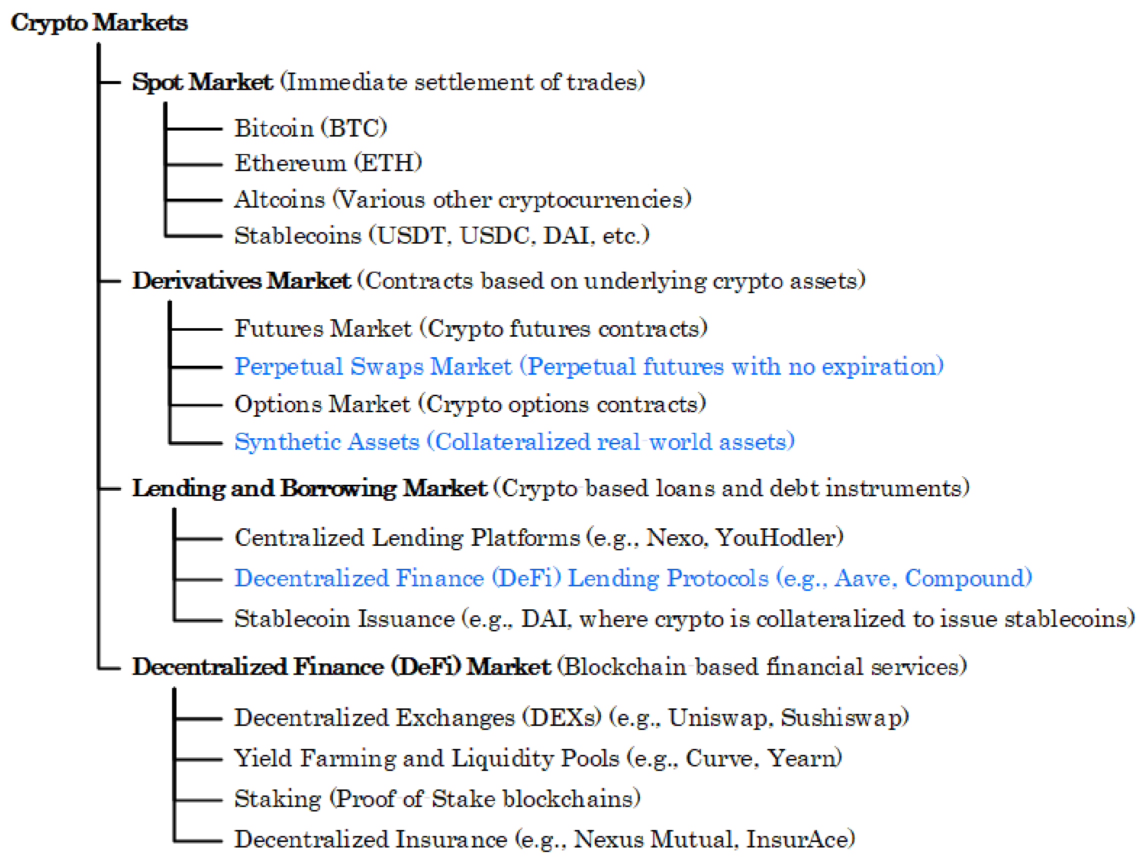

Crypto Markets
Spot Market (Immediate settlement of trades)
Bitcoin (BTC)
Ethereum (ETH)
Altcoins (Various other cryptocurrencies)
Stablecoins (USDT, USDC, DAI, etc.)
Derivatives Market (Contracts based on underlying crypto assets)
Futures Market (Crypto futures contracts)
Perpetual Swaps Market (Perpetual futures with no expiration)
Options Market (Crypto options contracts)
Synthetic Assets (Collateralized real world assets)
Lending and Borrowing Market (Crypto based loans and debt instruments)
Centralized Lending Platforms (e.g., Nexo, YouHodler)
Decentralized Finance (DeFi) Lending Protocols (e.g., Aave, Compound)
Stablecoin Issuance (e.g., DAI, where crypto is collateralized to issue stablecoins)
Decentralized Finance (DeFi) Market (Blockchain based financial services)
Decentralized Exchanges (DEXs) (e.g., Uniswap, Sushiswap)
Yield Farming and Liquidity Pools (e.g., Curve, Yearn)
Staking (Proof of Stake blockchains)
Decentralized Insurance (e.g., Nexus Mutual, InsurAce)

Figure 2.4: A hierarchy of various types of crypto markets and instruments, delineates the positions of the Futures and Lending markets (in blue).

On the other hand, *DeFi lending market* represents a more conservative alternative. It proposes the ability for LVTs to borrow assets through decentralized platforms like Aave and Compound.[10] Perpetual futures allow for higher leverage and are more riskier, while lending protocols provide lower leverage with reduced risk. In general, a choice between perpetual futures and lending protocols in the operation of LVTs depends on (i) specific design and purpose of the token, (ii) token's risk factor, (iii) the depth of liquidity provided by the underlying platform and (iv) the investment time horizon (short-term or long-term).

[10]Aave App: `https://app.aave.com/`, Compound App: `https://app.compound.finance/`

LVTs essentially abstract away the hassle of maintaining a leveraged position for their users. They simplify management of such a position without requiring users to constantly monitor and manage margin requirements. But how do these LVTs actually generate the leverage on behalf of users? Both crypto derivatives and DeFi lending markets provide efficient ways to obtain desired leverage factors for LVTs. With relatively modest leverage ratio in LVTs-commonly up to 5x-one could be used to obtain leverage in either market. Leverage is typically generated by using dated futures, perpetual contracts, or synthetic assets in derivatives markets. In the DeFi lending market, leverage comes from the ability of borrowing funds and posting collateral. Reinvesting borrowed amount can increase exposure and generate leverage for LVTs (see the highlighted options in blue in Figure 2.4).

2.6.1 Dated Futures

They can be used in LVTs to provide leverage till specific expiration date.[11] Unlike perpetual futures, which have no expiration date, dated futures contracts require the position to be settled or rolled over by the expiration date. LVTs using dated futures must roll over contracts before their expiration date to maintain the leveraged exposure. The smart contract managing the LVT sells the expiring contracts and simultaneously buys new contracts with a later expiration date. This process ensures that the fund continues to maintain its leveraged exposure to the underlying asset without interruption.

[11] Also known as fixed-expiry futures contracts.

The rollover process can sometimes result in minor expenses or discrepancies in the leverage ratio, depending on the difference between the price of the expiring and new contract, referred to as the "roll yield". When the futures market is in Contango[12], the roll yield is negative, and the LVT pays a premium to maintain its leveraged positions. In Backwardation[13], the roll yield is positive, and the LVT benefits from the lower price of the new contract. Another risk is that the smart contract may fail to call the renewal or roll over process of the expired futures. This leads to a few consequences, including loss of leverage exposure, token devaluation, and possible liquidation in DEXs if holders rush to sell off the under-collateralized tokens.

2.6.2 Perpetual Futures

They can be utilized as leveraged products in LVTs without the need to roll over an expiring contract on a particular date in the future. This reduces slippage and costs associated with dated futures. However, perpetual futures impose funding fees as opposed to dated futures (see section 2.7.1 for more information). Another disadvantage of them is that they are highly sensitive to volatility in the market. During big market swings, maintenance margin can rapidly change which increases the risk of forced liquidation. This makes perpetual futures more appropriate for short-term investment compared to buy-and-hold use cases.

[12]When the future prices are higher than the spot price
[13]When future prices are lower than the spot price

2.6.3 Automated Stacking

It is a technique for generating leverage through the lending market. Looped positions (*i.e.,* borrowing, re-depositing, and borrowing again) on fixed-rate[14] lending markets can be seen as somewhat analogous to dated futures contracts. Comparatively, variable-rate[15] positions on lending markets can be interpreted as the funding rate of the position, which in turn can be seen as a perpetual futures.

2.6.4 Synthetic Assets

These assets allow users to get exposure to a variety of asset price movements without necessarily owning them. Synthetic assets that track the value of fiat currencies, stocks, commodities, cryptocurrencies, and other financial derivatives already exist. They are usually backed by collateral (often in crypto) to ensure they maintain their value. For example, Synthetix protocol use collateralized debt to mint synthetic assets. They enable users to create tokens that track the value of real-world assets.

2.6.5 Summary of Key Differences

An LVT is primarily utilized by investors who need a tokenized fund. It simplifies the management of leveraged positions with liquidation protection. Hence, identification of the target user becomes critical when choosing the right leveraged product for issuing LVTs. For long-term investors, normally the debt market is cost-effective due

[14]A fixed rate for a specific period, with the possibility of adjustment over time.
[15]Rates that fluctuate in response to market conditions.

to less daily depreciation of tokens. As described below, the futures are suitable for short-term investors.

- *Perpetual futures:* Best used for short-term leveraged tokens, which need rebalancing quite often. The positive feature of these tokens is their flexibility and the possibility of high leverage in both the short and long directions.
- *Debt-based leveraged positions:* This strategy is more suitable for conservative LVTs. However, it comes with the risks of liquidation and interest rate fluctuation. The maximum leverage is constrained by the load-to-debt (LTV) ratio set by lenders and differs among blockchains. Creating LVTs with short position is more also challenging compared with long positions since overcollateralization is necessary.
- *Synthetic assets:* Synthetic assets give leveraged exposure completely decentralized, but at the risk of overcollateralization and concerns of oracle accuracy. These characteristics make them less attractive to be used in LVTs.

Advantages and disadvantages of each leveraged product over key factors vary based on different investment horizons and issuer's preference. The main purpose of using LVTs is by the short-term traders who try to realize a profit in the same trading day. Therefore, perpetual futures shall be the most suitable to be utilized in LVTs. However, a debt-based LVT with different characteristics, may be suitable for issuers with long-term investment horizons.

2.7 Key Factors in Crypto Futures Market

A futures contract, as explained, is an agreement to buy or sell a cryptocurrency at a given price at some future date. The Crypto Futures Market is the place where these contracts are traded, and such trading allows profiting on crypto price movements. Futures often refers to a class of derivatives that are traded in a market with much use of leverage, enabling traders to increase their exposure with less initial capital. It also allows hedging and speculation on future price movements. Perpetual futures are one of the dominant kinds of contracts in this market as users can keep a leveraged position open with no expiration date. The risks associated with these group of derivatives are mostly confined to the *Funding Fee* and the event of *Liquidation*.

2.7.1 Funding Fee

Traditional futures contracts have an expiration date known as the delivery date. Futures prices converge with the spot price on this date. Before that, the market can be in one of two conditions: Contango or Backwardation [88]. When the market is in Contango, futures are traded at a premium to the spot price (*i.e.,* they are more expensive). When the market is in Backwardation, futures are traded at a discount to the spot price (*i.e.,* they are cheaper than the spot price) [1]. Contango or Backwardation may occur due to shocks in supply or demand, carrying costs, geopolitical situations, pandemics, etc. Ultimately, traditional futures prices converge with the spot price by the delivery date (see Figure 2.5).

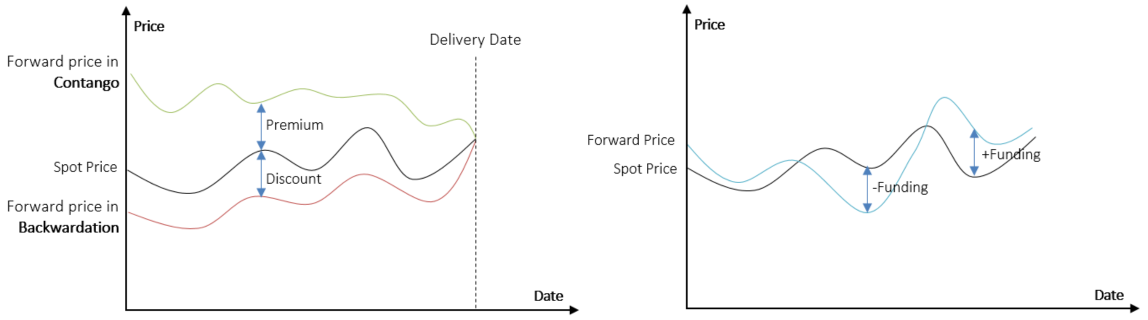


Figure 2.5: Contango or Backwardation conditions in the traditional futures market with a specific delivery date (left image). Price convergence over time in perpetual futures contracts, influenced by the funding rate (right image).

In crypto perpetual contracts, there is no delivery date, but the futures price still needs to settle against the spot price. At times, one side of the market becomes more aggressive, causing a disparity between spot and futures prices. To keep these prices aligned, a funding rate component is added to crypto futures. Only open positions are subject to funding payments or receipts at specific times (usually every 8 hours). If a position is closed before the funding exchange, traders do not pay or receive the funding fee. The funding fee can significantly impact high-leverage positions, potentially even leading to liquidation when paying for funding [135].

Funding fee represents a periodic payment exchanged between traders holding long and short positions, intended to keep the contract's price (P_t) aligned with the underlying asset's spot price (S_t). When $P_t > S_t$, the perpetual contract is trading at a premium, typically indicating higher demand for long positions. Conversely, when $P_t < S_t$, the contract is trading at a discount, suggesting higher demand for short positions. In fact, the funding fee is a mechanism designed to encourage traders to take the opposite position and correct the imbalance. Let F_t represent the funding rate at time t expressed by:

$$F_t = \alpha\left(\frac{P_t - S_t}{S_t}\right) + \beta\,(r_f - r_c) \tag{2.1}$$

where r_f denotes the interest rate of the fiat currency and r_c denotes the interest rate of the cryptocurrency. Subsequently, $(\frac{P_t-S_t}{S_t})$ and $(r_f - r_c)$ are the "premium or discount" and "interest rate" components, with weighted coefficients α and β, respectively. Both α and β are adjustable parameters that exchanges or DeFi platforms might calibrate based on market conditions, volatility, and the desired level of price convergence between the perpetual contract and the spot market. For instance, a high α might be used in a highly volatile market to quickly correct large premiums or discounts, while a high β might be more relevant in a market where the cost of capital (borrowing rates, staking yields, etc.) is highly volatile [49, 50].

2.7.2 Liquidation

It happens when the futures positions are automatically closed because of an inability to maintain a margin. The main drivers for liquidation are:

- *Adverse price movements:* The market moves against the position, which may lead to a decline in the notional value and reduced account equity.
- *Unfavorable funding rates:* Whenever the funding rate is negative, it causes a constant drain from the initial margin. Over time, the funding payments erode the initial margin, dropping it below the maintenance margin level-usually 5-

10% of the initial margin. The exchange will then liquidate the position.

The liquidation process is triggered when the price of the underlying asset falls below the *liquidation price*, at which the value of the initial margin is no longer adequate. The liquidation price (P_{Liq}) can be calculated as $P_{Liq} = P_{Ent} \times \left(1 - \frac{1-\mu}{\lambda}\right)$, where λ represents the leverage of the position (negative for shorts), P_{Ent} is the entry price and μ is maintenance margin in percentage.

Example 6. Bob opens a 10x long BTC/USDT Perps at $P_{Ent} = \$30K$ with an initial margin of \$3K and a maintenance margin of $\mu = 5\%$. Since this is a 10x position ($\lambda = 10$), it requires 10 times less capital than the full position value. The liquidation price is calculated as $P_{Liq} = \$30K \times (1 - \frac{1-5\%}{10}) = \$30K \times 0.905 = \$27,150$, at which point Bob's position would automatically be closed by the exchange, and his initial margin of \$3K would be used to cover the losses. In other words, Bob loses his entire initial investment as a result of liquidation.

Chapter 3

Resolving the Multiple Withdrawal Attack on ERC20 Tokens

This chapter is based on the paper "Resolving the Multiple Withdrawal Attack on ERC-20 Tokens" [143] supervised by Dr. Jeremy Clark. The paper is published in the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), at KTH Royal Institute of Technology in Stockholm, Sweden.

3.1 Introduction

In this chapter, we present a summary of our research on smart contracts and their subset, ERC-20 tokens. First, we examine the *"Multiple Withdrawal Attack"* in ERC-20 tokens, which was identified in November 2016 and has remained unresolved without a reliable solution. We then review ten proposed solutions and present two pro-

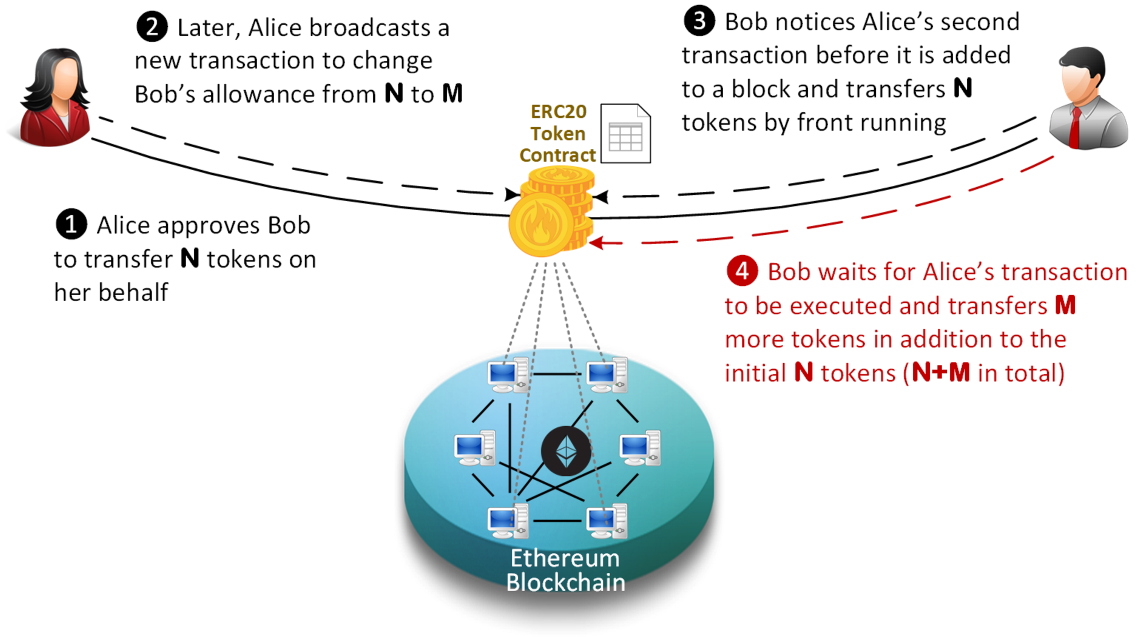

❷ Later, Alice broadcasts a new transaction to change Bob's allowance from N to M
❸ Bob notices Alice's second transaction before it is added to a block and transfers N tokens by front running
ERC20 Token Contract
❶ Alice approves Bob to transfer N tokens on her behalf
❹ Bob waits for Alice's transaction to be executed and transfers M more tokens in addition to the initial N tokens (N+M in total)
Ethereum Blockchain

Figure 3.1: Steps to perform the *"Multiple Withdrawal Attack"* in ERC-20 tokens.

posals to address this issue. By implementing one of the proposals, the security of future ERC-20 smart contracts will be significantly improved.

Since the introduction of ERC-20 in November 2015, several vulnerabilities have been discovered. In November 2016, a security issue called *"Multiple Withdrawal Attack"* was opened on GitHub [176, 91]. The attack originates from two methods in the ERC-20 standard for approving and transferring tokens. The use of these functions in an adverse environment (*e.g.,* front-running [64]) could result in more tokens being spent than what was intended. This issue is still open and several solutions have been made to mitigate it.

According to the ERC-20 API definition, the `approve` function allows a spender (*e.g.,* user, wallet or other smart contracts) to withdraw up to an allowed amount of tokens from token pool of the approver. If this function is called again, it overwrites the current allowance with the new input value. On the other hand, the `transferFrom` function allows the spender to actually transfer tokens from the approver to anyone they choose (importantly: not necessarily themselves). The contract updates balance of transaction parties accordingly. An adversary can exploit the gap between the confirmation of the `approve` and `transferFrom` functions since the `approve` method replaces the current spender allowance with the new amount, regardless of whether the spender already transferred any tokens or not. This functionality of the `approve` method is shaped by the language of the standard and cannot be changed. Furthermore, while variables change and events are logged, this information is ambiguous and cannot fully distinguish between possible traces. Consider steps in Figure 3.1:

1. Alice allows Bob to transfer N tokens on her behalf by `approve(_Bob, N)`.
2. Later, Alice changes Bob's approval from N to M by `approve(_Bob, M)`.
3. Bob notices Alice's second transaction after its broadcast to the Ethereum network but before adding to a block.
4. Bob front-runs (using an asymmetric insertion attack [64]) the original transaction with a call to `transferFrom(_Alice, _Bob, N)`. If a miner is incentivized (*e.g.,* by Bob offering high gas) to add this transaction before Alice's, it will transfer N of Alice's tokens to Bob.
5. Alice's transaction will then be executed which changes Bob's approval to M.
6. Bob can call `transferFrom` method again and transfer M additional tokens.

In summary, in attempting to change Bob's allowance from N to M, Alice makes it possible for Bob to transfer N+M of her tokens. We operate on the assumption that a secure implementation would prevent Bob from withdrawing Alice's tokens multiple times when the allowance changes from N to M. A proposed solution should prevent this from happening by allowing only M approved tokens to be transferred after changing from the initial N tokens.

3.2 Motivation of the Approval Process

In the ERC-20 token standard, the *allowance mechanism* via `approve` and `transferFrom`, allows a token owner to grant permission to another address (typically a smart contract or third-party account) to spend tokens on their behalf. This mechanism exists instead of directly sending tokens to a contract for security, flexibility, and usability reasons.

3.2.1 Security: Prevents Loss of Funds

If a user sends tokens *directly* to a smart contract that is not programmed to handle ERC-20 tokens, the funds could be permanently lost. Many smart contracts do not have a built-in way to recover such mistakenly sent tokens. For example, suppose Alice wants to deposit tokens into a DeFi lending contract. If she sends the tokens directly (via `transfer` method), but the contract is not programmed to handle deposits this way, the tokens might be stuck or lost. Instead, using `approve` + `transferFrom`, the contract explicitly pulls tokens when it is ready to process them correctly.

3.2.2 Flexibility: Enables Conditional Spending

The `approve` + `transferFrom` pattern allows third-party contracts (or users) to spend tokens on behalf of the owner under controlled conditions. This is critical for many DeFi applications. For example, Alice wants to deposit tokens into a *staking contract*, but only if certain conditions (*e.g.,* token price, gas fees) are met. The staking contract can wait until the conditions are met and then call `transferFrom` to pull the tokens from the Alice's balance. This approach gives contracts the ability to execute transfers only when needed rather than requiring Alice to manually send tokens every time.

3.2.3 Decentralized Authorization: Users Keep Control

Instead of handing over their tokens immediately, users can specify exactly how many tokens a third party is allowed to withdraw. This is important for security as users don't have to trust a contract with an unlimited amount of tokens. It also provides granular control as users can set limits on how many tokens an exchange, lending protocol, or escrow contract can spend. For example, Alice trades tokens on a decentralized exchange (DEX). Instead of depositing all her tokens into the exchange (which could be risky), she approves the exchange to withdraw only the amount she wants to trade.

3.2.4 Gas Efficiency: Reduces Unnecessary Transfers

Using `approve` + `transferFrom` can sometimes save gas compared to transferring tokens multiple times manually. for example, Alice joins a yield farming contract.

Instead of transferring tokens in multiple steps, she approves the contract, which then automatically pulls the exact required amount in a single transaction. This reduces unnecessary transfers and minimizes transaction costs.

3.2.5 Required for Automated Interactions

Many smart contract applications, such as DEXs, lending platforms, subscriptions, and staking, rely on `approve` + `transferFrom` because they need to move funds on behalf of users. For example, a subscription service automatically withdraws a fixed number of tokens every month from a Alice's wallet. She grants an allowance once, and the service uses `transferFrom` to collect payments without requiring manual approval each time.

In conclusion, the `approve` + `transferFrom` mechanism in ERC-20 tokens provides safety, flexibility, and efficiency compared to direct token transfers. It allows users to retain control over their tokens while enabling smart contracts, exchanges, and other third-party applications to securely interact with token balances in a decentralized and automated way [189].

3.3 Significance of Mitigation

ERC-20 tokens are important component of Ethereum's supplementary financial system that have many financial (as well as non-financial) uses and could hold considerable value (potentially exceeding the value of Ether itself). There has been more than

64,000 functional ERC-20 tokens as of early 2019 [175] that might be vulnerable to this attack. Furthermore, ERC-20 tokens that have already been issued cannot easily migrate to a new secure implementation and should these tokens appreciate in value in the future. Resolving the attack also serves as basis for other extended standards, such as ERC-777 [40] to be backward compatible with ERC-20 interface [80]. Finally, firms that hold ERC-20 tokens require assurance of their security, particularly in the case that they require their financial statements to be audited—an issue like this could lead to further hesitation by auditors.

3.4 Related Work

This work builds upon prior research in Ethereum smart contract security, token standards, and front-running vulnerabilities. The ERC-20 standard, introduced in 2015 [70], has been widely adopted but has also faced security concerns, particularly regarding the `approve` function and its susceptibility to race conditions. Previous mitigation efforts include modifications like `increaseApproval` and `decreaseApproval` functions (as seen in OpenZeppelin [123] and MonolithDAO [174]) and UI-level enforcement strategies [27]. However, these approaches fail to fully eliminate *"Multiple Withdrawal Attack"* while maintaining ERC-20 compatibility. This work advances the field by systematically evaluating ten existing mitigations, identifying their limitations, and proposing two novel solutions based on the Compare-and-Set (CAS) pattern [32]. Our proposals — one modifying the `approve` method and another securing `transferFrom`— offer stronger security guarantees while preserving backward compatibility with ERC-20, making them valuable contributions to Ethereum's token ecosystem.

3.5 Transaction-Ordering Dependence

The *"Multiple Withdrawal Attack"* could happen when multiple transactions invoke the same contract at roughly the same time. The next block will most likely execute and include these transactions. For example, considering contract c at state s_0, a miner may receive transactions T_i and T_j from two different users ($|T_i - T_j| \leq 12$ *seconds*). Depending on the miner decision on the order of T_i, T_j, there would be uncertainty about the invocation results (*i.e.,* $s_0 \xrightarrow{T_i} s_1 \xrightarrow{T_j} s_2$ or $s_0 \xrightarrow{T_j} s_3 \xrightarrow{T_i} s_4$). Thus, there might be a discrepancy between the state of the contract when users invoke the transaction and the actual state when the execution happens [115]. This issue might happen in one of the following scenarios:

- **Race-condition:** A non-malicious scenario wherein two users (or contracts) attempt to perform two invocations at the same time. Concurrent transactions may lead to an unexpected results due to uncertainty in the execution order. For example, in a decentralized exchange, sellers might update ask prices while buyers submit the bid prices based on the observed ask prices. Depending on the transaction ordering, buy requests may or may not go through, or buyers may pay more for a dropped price.

- **Front-running:** A malicious scenario wherein miners can change sequence of new transactions to gain profit. This is the required condition to perform *"Multiple Withdrawal Attack"* in ERC-20 tokens where Alice attempts to change Bob's allowance from N to M, but she made it possible for Bob (who is running a mining node) to transfer N+M due to change in the execution order of `approve()` and `transferFrom()` transactions.

Mitigating this vulnerability is not easy and depends on the logic of the smart contract. It can happen since there is a time gap around 12 seconds between (1) broadcasting the transaction and (2) including the transaction in a new block. Thus, a malicious miner can listen to the network and change transaction orders for financial advantages. This is part of Ethereum's consensus protocol that allows arbitrary miners to participate in the network. The consensus protocol elects a leader among all miners and the leader then broadcasts its proposed block to all nodes. After block validation, all nodes update their local copy to include the new block.

There are a few logical places to address this attack. Ideally the token author (instead of the token holders) would mitigate the attack within the ERC-20 smart contract. Since two methods are involved in the attack, it could be addressed within the `approve` and/or `transferFrom` method. By contrast, token owners have no control over the implementation of the contract and are relegated to mitigate the attack by monitoring the contract around the time allowance changes are made. Since this is very difficult, we concentrate on mitigating the attack in the contract itself.

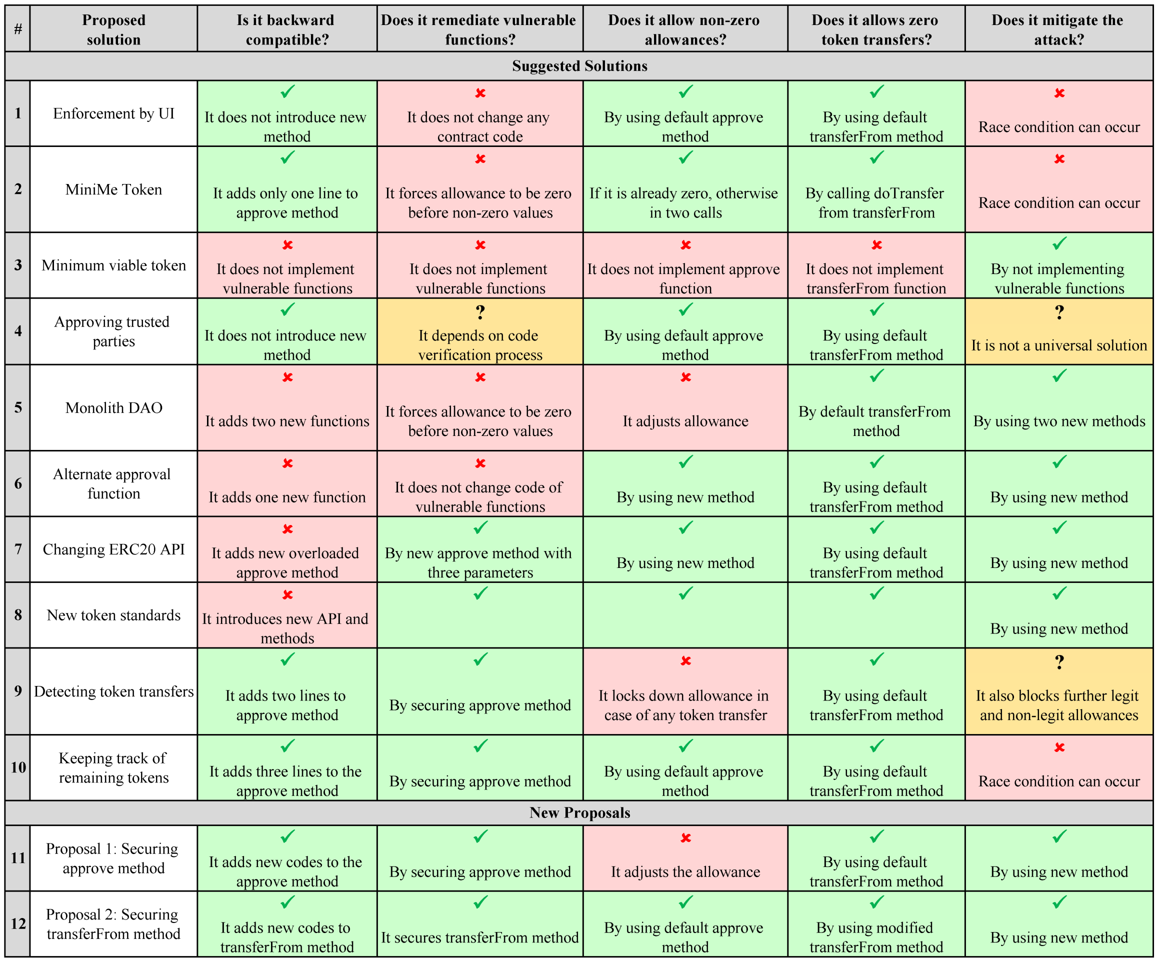
3.6 Evaluating Proposed Mitigations

The authors of the ERC-20 [70] reference two sample implementations from OpenZeppelin [123] and ConsenSys [28]. OpenZeppelin implementation mitigates the attack by introducing two additional methods to increase or decrease approved tokens, and the ConsenSys code does not attempt to resolve it. Additional implementations have a variety of different trade-offs in mitigating the issue. We evaluate ten proposed solutions and develop a set of criteria that encompass backwards compatibility, interoperability, adherence to the ERC-20 standard, and attack mitigation.

The summary of evaluations along with our two proposals is provided in Figure 3.2. Proposal 1 mitigates the attack by comparing transferred tokens with new allowance. It is not fully compliant with ERC-20 specifications since the allowance result does not always match what is requested. In proposal 2, a new variable is defined to keep track of transferred tokens and prevents transfers in the case of already transferred tokens. We examine the details of the two proposals in Sections 3.7.1 and 3.7.2.

3.7 New Mitigations

Since no mitigation is fully satisfactory, we develop two additional solutions based on the *Compare and Set (CAS)* pattern [183]. We study in detail possible implementations of the `approve` and `transferFrom` methods. We argue that a CAS-based approach can never adequately deploy a secure `approve` method while adhering to the ERC-20 standard. We therefore prioritize adherence to the ERC-20 standard.



#	Proposed solution	Is it backward compatible?	Does it remediate vulnerable functions?	Does it allow non-zero allowances?	Does it allows zero token transfers?	Does it mitigate the attack?
Suggested Solutions						
1	Enforcement by UI	✓ It does not introduce new method	✗ It does not change any contract code	✓ By using default approve method	✓ By using default transferFrom method	✗ Race condition can occur
2	MiniMe Token	✓ It adds only one line to approve method	✗ It forces allowance to be zero before non-zero values	✓ If it is already zero, otherwise in two calls	✓ By calling doTransfer from transferFrom	✗ Race condition can occur
3	Minimum viable token	✗ It does not implement vulnerable functions	✗ It does not implement vulnerable functions	✗ It does not implement approve function	✗ It does not implement transferFrom function	✓ By not implementing vulnerable functions
4	Approving trusted parties	✓ It does not introduce new method	? It depends on code verification process	✓ By using default approve method	✓ By using default transferFrom method	? It is not a universal solution
5	Monolith DAO	✗ It adds two new functions	✗ It forces allowance to be zero before non-zero values	✗ It adjusts allowance	✓ By default transferFrom method	✓ By using two new methods
6	Alternate approval function	✗ It adds one new function	✗ It does not change code of vulnerable functions	✓ By using new method	✓ By using default transferFrom method	✓ By using new method
7	Changing ERC20 API	✗ It adds new overloaded approve method	✓ By new approve method with three parameters	✓ By using new method	✓ By using default transferFrom method	✓ By using new method
8	New token standards	✗ It introduces new API and methods	✓	✓	✓	✓ By using new method
9	Detecting token transfers	✓ It adds two lines to approve method	✓ By securing approve method	✗ It locks down allowance in case of any token transfer	✓ By using default transferFrom method	? It also blocks further legit and non-legit allowances
10	Keeping track of remaining tokens	✓ It adds three lines to the approve method	✓ By securing approve method	✓ By using default approve method	✓ By using default transferFrom method	✗ Race condition can occur
New Proposals						
11	Proposal 1: Securing approve method	✓ It adds new codes to the approve method	✓ By securing approve method	✗ It adjusts the allowance	✓ By using default transferFrom method	✓ By using new method
12	Proposal 2: Securing transferFrom method	✓ It adds new codes to transferFrom method	✓ It secures transferFrom method	✓ By using default approve method	✓ By using modified transferFrom method	✓ By using new method

Figure 3.2: Evaluation of the ten proposed mitigations for the *"Multiple Withdrawal Attack"*.

While deviating from the standard might become acceptable if there is no possible way to conform with it and maintain security, we consider that a last resort. Indeed, as we will show, it is possible to secure an ERC-20 contract within the constraints of the standard [70]. Requirements of an ideal solution are summarized here:

- The input to `approve` method is a new allowance and not a relative adjustment.
- The result of `approve` method will overwrite the current allowance with the new allowance.

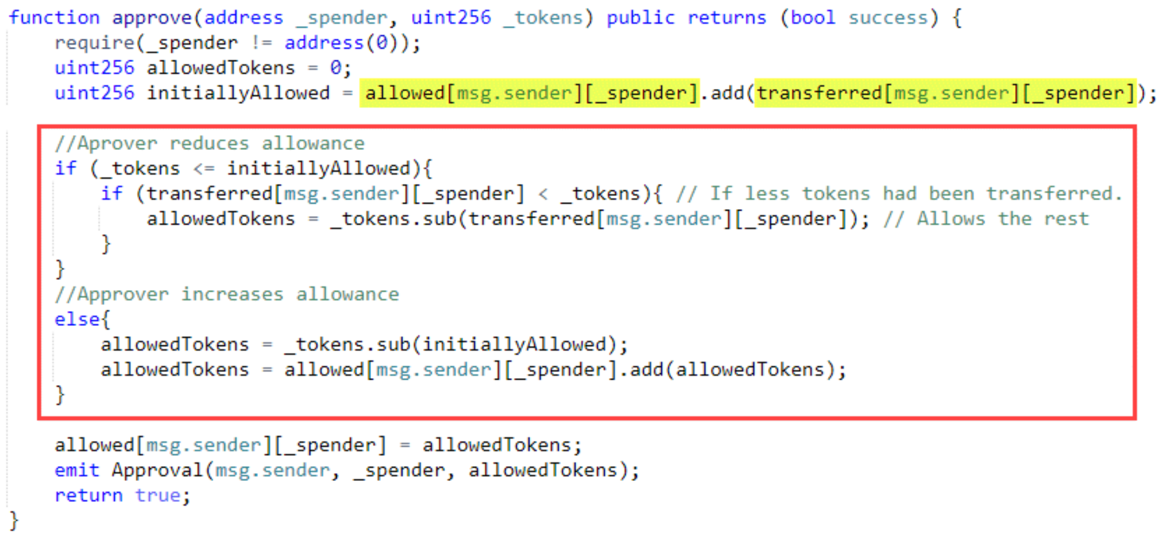
- A call to `transferFrom` on an input of 0 tokens will execute as a normal transfer and emit a `Transfer` event.
- A spender can call `transferFrom` multiple times up to the allowed amount.
- Transferring up to any initial allowance is always a legitimate transfer.
- An ideal solution cannot rely on overloading existing methods or introducing new methods outside of ERC-20, as existing DApps and web apps would have to be modified to interoperate. A solution must eliminate all race conditions.

We now propose two solutions that that mitigates the attack by: (i) securing implementation of the `approve` method, (ii) securing implementation of the `transferFrom`.

3.7.1 Proposal 1: Securing Implementation of `approve`

By implementing the CAS pattern [183] in the `approve` method (highlighted code in the code snippet of Figure 3.3), we set up a small state machine so that new allowances can be set atomically after a comparison with transferred tokens. This tracking also requires adding a new variable to the `transferFrom` method. Since this is an internal variable, it is not visible to already deployed smart and keeps the `transferFrom` function compatible. Similarly, a block of code is added to the `approve` function (Shown with a red box in the code snippet of Figure 3.3) to work in both cases with zero and non-zero allowances. This new logic in the `approve` function compares a new allowance—passed as `_tokens` argument to the function—with the current allowance of the spender and the already transferred tokens.

Allowance are saved in `allowed[msg.sender][_spender]` variable as in typi-



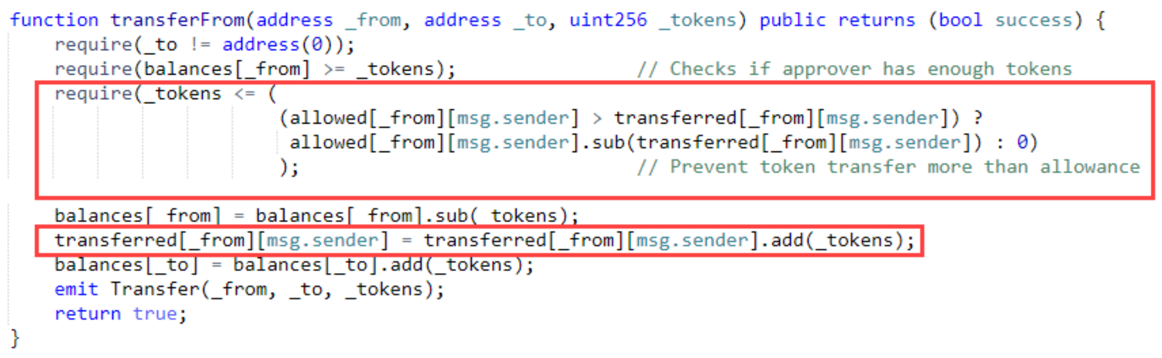
```
function approve(address _spender, uint256 _tokens) public returns (bool success) {
    require(_spender != address(0));
    uint256 allowedTokens = 0;
    uint256 initiallyAllowed = allowed[msg.sender][_spender].add(transferred[msg.sender][_spender]);

    //Aprover reduces allowance
    if (_tokens <= initiallyAllowed){
        if (transferred[msg.sender][_spender] < _tokens){ // If less tokens had been transferred.
            allowedTokens = _tokens.sub(transferred[msg.sender][_spender]); // Allows the rest
        }
    }
    //Approver increases allowance
    else{
        allowedTokens = _tokens.sub(initiallyAllowed);
        allowedTokens = allowed[msg.sender][_spender].add(allowedTokens);
    }

    allowed[msg.sender][_spender] = allowedTokens;
    emit Approval(msg.sender, _spender, allowedTokens);
    return true;
}
```

Figure 3.3: Resolving *"Multiple Withdrawal Attack"* by implementing the CAS pattern in the `approve` method.

cal ERC-20 implementation, and `transferred[msg.sender][_spender]` is the new state. The method decides to increase or decrease the current allowance based on this comparison. If the new allowance is less than initial allowance—sum of `allowance` and `transferred` variables—it denotes decreasing of allowance, otherwise increasing of allowance is intended. Such a modified `approve` function prevents the attack by either increasing or decreasing the allowance instead of setting it to an explicit value.

In summary, we can use the CAS[1] pattern to implement a secure `approve` method that can mitigate the attack effectively. However, it violates one of the ERC-20 specifications that says: "If `approve` function is called again, it overwrites the current allowance with `_value`". Our solution does not comply with this as the resulting allowance can be different than what is passed by the approver. Furthermore we argue that is in fact impossible to secure the `approve` method without adjusting

[1]A lock-free synchronization strategy that allows comparing and setting values atomically.



```
function transferFrom(address _from, address _to, uint256 _tokens) public returns (bool success) {
    require(_to != address(0));
    require(balances[_from] >= _tokens);                // Checks if approver has enough tokens
    require(_tokens <= (
                    (allowed[_from][msg.sender] > transferred[_from][msg.sender]) ?
                     allowed[_from][msg.sender].sub(transferred[_from][msg.sender]) : 0)
                    );                                  // Prevent token transfer more than allowance

    balances[_from] = balances[_from].sub(_tokens);
    transferred[_from][msg.sender] = transferred[_from][msg.sender].add(_tokens);
    balances[_to] = balances[_to].add(_tokens);
    emit Transfer(_from, _to, _tokens);
    return true;
}
```

Figure 3.4: Resolving *"Multiple Withdrawal Attack"* by securing the `transferFrom`.

the allowance. Therefore the `approve` method has to adjust the allowance according to transferred tokens, not based on passed input values to the `approve` method. Overall, there seems to be no solution to secure the `approve` method while adhering specification of ERC-20 standard. In proposal 2, the focus is to take this requirement into consideration by securing the `transferFrom` method.

3.7.2 Proposal 2: Securing Implementation of `transferFrom`

As an alternative to Proposal 1, we can also consider securing the `transferFrom` method. As specified by the ERC-20 standard, the goal here is to prevent the spender from transferring more tokens than allowed. Based on this assumption, we should not rely solely on the allowance value in deciding whether to allow or prevent an approve and should also consider the number of transferred tokens, which requires new state as in Proposal 1. Our solution, which is compliant with a careful reading of ERC-20 specifications, is to interpret allowance as a 'global' or 'lifetime' allowance value, instead of the amount allowed at the specific time of invocation (see red boxes in the code snippet of Figure 3.4).

Example 7. Alice approves Bob for 50 tokens, Bob transfers 50 tokens, Alice approves Bob for 30 (more) tokens, and Bob transfers 30 tokens. In our implementation, Alice would approve Bob for 50 tokens and he transfers 50 tokens. To approve Bob for 30 more tokens, she approves Bob for 80 tokens. He has already spent 50 of these 80 tokens so he will only be allowed to transfer an addition 30. Thus 80 is his lifetime allowance and 50 (kept internally) is the amount he has transferred.

In a bit more detail, consider the following, which prevents multiple withdrawals by modifying the implementation of `transferFrom` but keeping `approve` untouched:

1. Alice approves Bob to transfer 100 tokens
2. Alice broadcasts an approval of 70, decreasing Bob's allowance.
3. Bob front-runs Alice's transaction and transfers 100 tokens (remark: a legitimate transfer).
4. Alice's transaction is confirmed and sets Bob allowance to 70 by the default `approve` method.
5. Bob's noticed the new allowance and tries to move 70 additional tokens by broadcasting `transferFrom(_Bob,70)`.
6. Since Bob has already transferred more than 70 tokens, his transaction fails and prevents multiple withdrawal.
7. After all, Bob's allowance is set at 70 and his transferred tokens are set at 100.

Our advice to developers is to use proposal 2 which is compatible with the ERC-20 token standard. Additionally, while this is a deep dive into a specific issue with ERC-20, it also illustrates a number of higher level lessons for blockchain developers. When

ERC-20 standard was first implemented, it changed how people used Ethereum, giving rise to an ICO craze with its ease of use [71]. This led to the deployment of thousands of early implementation of ERC-20 tokens which has resulted in numerous attacks on different implementations. Now we see decentralized exchanges relying on existing ERC-20 tokens and the *"Multiple Withdrawal Attack"* seems too important to ignore. Fixing existing ERC-20 code will help future deployments but cannot fix the already deployed tokens. In addition to deploying secure contracts, we suggest blockchain developers conduct external audits and consider security-by-design practices when dealing with other smart contract implementations.

Performance

In order to evaluate functionality of the new `approve` and `transferFrom` functions, we have implemented a standard ERC-20 token (TKNv1[2]) along side the proposed ERC-20 token (TKNv2[3]). Our testing for different input values shows that TKNv2 can address the *"Multiple Withdrawal Attack"* by making front-running gains ineffective. Moreover, we compared these two tokens in term of gas consumption. The `approve` function in TKNv2 uses almost the same amount of gas as TKNv1, however, gas consumption of `transferFrom` in TKNv2 is around 37% more than TKNv1. This difference in TKNv2 is because of maintaining a new mapping variable for tracking transferred tokens. In term of compatibility, both are equivalent interoperable with standard wallets (*e.g.,* MetaMask and MEV) and have not raised any transfer issues.

[2]`https://rinkeby.etherscan.io/address/0x8825bac68a3f6939c296a40fc8078d18c2f66ac7`
[3]`https://rinkeby.etherscan.io/address/0x5d148c948c01e1a61e280c8b2ac39fd49ee6d9c6`

We believe this is acceptable for having a secure ERC20 token.

3.8 Contributions

We evaluate ten proposed mitigations for the *"Multiple Withdrawal Attack"* and develop a set of criteria that encompass backwards compatibility, interoperability, adherence to the ERC-20 standard, and attack mitigation. Since no mitigation is fully satisfactory, we develop two additional solutions based on the *Compare and Set (CAS)* pattern. We study in detail possible implementations of ERC-20's `approve` and `transferFrom` methods. We argue that a CAS-based approach can never adequately deploy a secure `approve` method while adhering to the ERC-20 standard. We then propose a secure implementation of the `transferFrom` method that mitigates the attack and fully satisfies the ERC-20 standard. It consumes 37% more gas compared with the non-secure `transferFrom` method, which appears to be acceptable for achieving a secure ERC-20 implementation that protects user investments and provide a secure token transfer mechanism.

3.9 Discussion

This chapter was a deep dive into a specific issue with ERC-20, illustrating a number of higher level lessons for blockchain developers. When ERC-20 standard was first implemented, it changed how people used Ethereum, giving rise to an ICO craze with its ease of use [71]. This led to the deployment of thousands of early implementation of ERC-20 tokens which has resulted in numerous attacks on different implementations. Now we see decentralized exchanges relying on existing ERC-20 tokens and the *"Multiple Withdrawal Attack"* seems too important to ignore. Fixing existing ERC-20 code will help future deployments but cannot fix the already deployed tokens. In addition to deploying secure contracts, we suggest blockchain developers conduct external audits and consider security-by-design practices when dealing with other smart contract implementations.

Chapter 4

TokenHook: Secure ERC-20 Smart Contract

This chapter is based on the paper "TokenHook: Secure ERC-20 Smart Contract" [141] supervised by Dr. Jeremy Clark.

4.1 Introduction

In this chapter, we present a summary of our research on ERC-20 vulnerabilities and a new proposal, TokenHook, as a secure ERC-20 token contract. Of the 2.5M smart contracts on Ethereum [145], 260K are tokens [168] and 98% of these tokens are ERC-20. Although various token standards (*e.g.,* ERC-721, ERC-777, ERC-1155) have been introduced by the Ethereum community, ERC-20 is widely used for industrial implementations. Therefore, our research focus is on this type of tokens.

The authors of the ERC-20 standard reference two sample implementations: one that is actively maintained by OpenZeppelin [123] and one that has been deprecated by ConsenSys [28] (and now refers to the OpenZeppelin implementation). As expected, the OpenZeppelin template is very popular within the Ethereum developers [148, 188, 139]. However, diversity in software is important for robustness and security [76, 75]. For ERC-20, a variety of implementations will reduce the impact of a single bug in a single implementation. For example, between 17 March 2017 and 13 July 2017, OpenZeppelin's implementation used the wrong interface and affected 130 tokens [38]. This is also one the cases that motivated us for using our experience to provide a new secure implementation of the ERC-20 interface called TokenHook. It is freely available and can be used by developers as a reference implementation.

TokenHook is our compliant ERC-20 implementation written in Vyper (v. 0.2.8) and Solidity (v. 0.8.4).[1] It can be customized by developers, who can refer to each mitigation technique separately and address specific attacks. Required comments are added to clarify the usage of each function. Standard functionalities of the token (*i.e.,* `approve()`, `transfer()`, `transferFrom()`, *etc.*) have been unit tested. A demonstration of token interactions and event triggering can also be seen on Etherscan[2].

Among the layers of the Ethereum blockchain, ERC-20 tokens fall under the *Contract layer* in which dApps are executed. The presence of a security vulnerability in supplementary layers affect the entire Ethereum blockchain, not necessarily ERC-20

[1]TokenHook deployed on Rinkeby at `https://bit.ly/33wDENx` (Solidity) and `https://bit.ly/3dXaaPc` (Vyper). Mainnet at `https://bit.ly/35FMbAf` (Solidity 0.5.11)

[2]Etherscan: `https://bit.ly/33xHfL2`, `https://bit.ly/35TimMW` and `https://bit.ly/3eFAnAZ`.

tokens. Therefore, vulnerabilities in other layers are assumed to be out of the scope TokenHook implementation. (*e.g., Indistinguishable chains* at the data layer, the *51% attack* at the consensus layer, *Unlimited nodes creation* at network layer, and *Web3.js Arbitrary File Write* at application layer). Moreover, we exclude vulnerabilities identified in now outdated compiler versions. Examples: *Constructor name ambiguity* in versions before 0.4.22, *Uninitialized storage pointer* in versions before 0.5.0, *Function default visibility* in versions before 0.5.0, *Typographical error* in versions before 0.5.8, *Deprecated solidity functions* in versions before 0.4.25, *Assert Violation* in versions before 0.4.10, *Under-priced DoS attack* before EIP-150 & EIP-1884).

4.2 Related Work

This work builds upon previous research in Ethereum security, token standards, and smart contract auditing. The ERC-20 standard, first introduced in 2015 [70], has become the dominant framework for fungible tokens, but its security vulnerabilities have been widely studied [17]. Prior research has highlighted critical weaknesses such as re-entrancy [73], integer overflows [129], and the multiple withdrawal attack [144], leading to various mitigation techniques like OpenZeppelin's SafeMath [125] and alternative approval mechanisms [124]. Despite these efforts, security challenges persist, as demonstrated by past high-profile exploits. This work extends these studies by systematizing known ERC-20 vulnerabilities, evaluating the effectiveness of seven auditing tools, and proposing TokenHook as a secure alternative ERC-20 implementation.

By offering an open-source, security-enhanced reference contract in both Solidity and Vyper, TokenHook contributes to software diversity and provides a practical benchmark for assessing smart contract security tools.

4.3 Security Features

In our research, we identified 53 security vulnerabilities and 29 best practices for ERC-20 tokens (82 guidelines in total). These vulnerabilities pose risks to the integrity and safety of ERC-20 tokens, while the best practices are meant to mitigate these risks. Our goal is to proactively prevent known issues by integrating these best practices into the development process, helping to safeguard against vulnerabilities before they can be exploited.

We focus here on how TokenHook mitigates these attacks. While many of these attacks are no doubt very familiar to the reader, our emphasis is on their relevance to ERC-20 smart contract. We sample some high profile vulnerabilities, typically ones that have been exploited in real world ERC-20 tokens [117, 97, 57, 47, 109]. For each, we (i) briefly explain technical details, (ii) the ability to affect ERC-20 tokens, and (iii) discuss mitigation techniques. Some of these vulnerabilities are:

Multiple Withdrawal Attack

An attacker can use a front-running attack [37, 64] to transfer more tokens than what is intended (approved) by the token holder. We secure the `transferFrom()` function by tracking transferred tokens to mitigate the *"Multiple Withdrawal Attack"* [143]. Securing the `transferFrom()` function is fully compliant with the ERC-20 standard without the need of introducing new functions such as `decreaseApproval()` and `increaseApproval()`.

Arithmetic Over/Under Flows

In Solidity implementation, we use the `SafeMath` library in all arithmetic operations to catch over/under flows. Using it in Vyper is not required due to built-in checks.

Re-entrancy

At first glance, re-entrancy might seem inapplicable to ERC-20. However any function that changes internal state, such as balances, need to be checked. We use Checks-Effects-Interactions pattern (CEI) [56] in both Vyper and Solidity implementations to mitigate *same-function re-entrancy* attack. Mutual exclusion (Mutex) [184] is also used to address *cross-function re-entrancy* attack. Vyper supports Mutex decorator on a function and we use `noReentrancy` modifier in Solidity to apply Mutex. Therefore, both re-entrancy variants are addressed in TokenHook.

Unchecked Return Values

Unlike built-in support in Vyper, we must check the return value of `call.value()` in Solidity to revert failed fund transfers. It mitigates the *unchecked return values* attack while making the token contract compatible with EIP-1884 [96].

Frozen Ether

We mitigate it by defining a `withdraw()` function that allows the owner to transfer all Ether out of the token contract. Otherwise, unexpected Ether forced onto the token contract (*e.g.,* from another contract running `selfdestruct`) will be stuck forever.

Unprotected Ether Withdrawal

We enforce authentication before transferring any funds out of the contract to mitigate *unprotected Ether withdrawal*. Explicit check is added to the Vyper code and `onlyOwner` modifier is used in Solidity implementation. It allows only owner to call `withdraw()` function and protects unauthorized Ether withdrawals.

State Variable Manipulation

In the Solidity implementation, we use embedded `Library` code (for `SafeMath`) to avoid external calls and mitigate the *state variable manipulation* attack. It also reduces gas costs since calling functions in embedded libraries requires less gas than external calls.

Function Visibility

We carefully define the visibility of each function. Most of the functions are declared as `External` (*e.g.,* `Approve()`, `Transfer()`, *etc.*) per specifications of ERC-20 standard.

4.4 Best Practices and Enhancements

We also highlight a few best practices that have been accepted by the Ethereum community to proactively prevent known vulnerabilities [48]. Some best practices are specific to ERC-20, while others are generic for all dApps — in which case, we discuss their relevance to ERC-20 standard. Some of these best practices are:

Compliance with ERC-20

We implement all ERC-20 functions to make it fully compatible with the standard. Compliance is important for ensuring that other DApps and web apps (*i.e.,* crypto-wallets, crypto-exchanges and web services) compose with TokenHook as expected.

External Visibility

To improve performance, we apply an `external` visibility (instead of `public` visibility in the standard) for interactive functions (*e.g.,* `approve()` and `transfer()`, *etc.*). External functions can read arguments directly from non-persistent `calldata` instead of allocating persistent memory by the EVM.

Fail-Safe Mode

We implement a 'cease trade' operation that will freeze the token in the case of new security threats or new legal requirements (*e.g.,* Liberty Reserve [185] or TON cryptocurrency [61]). To freeze all functionality of TokenHook, the owner (or multiple parties) can call the function `pause()` which sets a lock variable. All critical methods are either marked with a `notPaused` modifier (in Solidity) or explicit check (in Vyper), that will throw exceptions until functionality is restored using `unpause()`.

Firing Events

We define nine extra events: `Buy`, `Sell`, `Received`, `Withdrawal`, `Pause`, `Change`, `ChangeOwner`, `Mint` and `Burn`. The name of each event indicates its function except `Change` event which logs any state variable updates. It can be used to watch for token inconsistent behavior (*e.g.,* via TokenScope [19]) and react accordingly.

Global or Miner Controlled Variables

Since malicious miners are able to manipulate global Solidity variables (*e.g.,* `block.timestamp`, `block.number`, `block.difficulty`, *etc.*), it is recommended to avoid these variables in ERC-20 tokens.

DoS with Block Gas Limit

The use of loops in contracts is not efficient and may lead to DoS attack. If execution of a function exceeds the block gas limit, all transactions in that block will fail. Hence, it is recommended to not use loops and rely on `mappings` variables which store data in collection of key value pairs and are more efficient for tracking owned tokens by each holder.

Proxy Contracts

We choose to make TokenHook non-upgradable so it can be audited, and upgrades will not introduce new vulnerabilities that did not exist at the initial audit.

Other Enhancements

We also follow other best practices such as not using batch processing in `sell()` function to avoid *DoS with unexpected revert* issue, not using miner controlled variable in conditional statements, and not using `SELFDESTRUCT`.

4.5 Need for Another Reference Implementation

OpenZeppelin's implementation is actually part of a small portfolio of implementations (ERC-20, ERC-721, ERC-777, and ERC-1155). Code reuse across the four implementations adds complexity for a developer that only wants ERC-20. This might be the reason for not supporting Vyper in OpenZeppelin's implementation. No inheritance in Vyper requires different implementation than the current object-oriented OpenZeppelin contracts. Further, most audit tools are not able to import libraries/interfaces from external files (*e.g.,* SafeMath.sol, IERC20.sol). By contrast, TokenHook uses a flat layout in a single file that is specific to ERC-20. It does not use inheritance in Solidity which allows similar implementation in Vyper.

Although Vyper offers less features than Solidity (*e.g.,* no class inheritance, modifiers, inline assembly, function/operator overloading, *etc.* [66]), the Vyper compiler

Vulnerability (Vul.) or Best Practice (BP.)		**TokenHook Implementation**		**Comment**
		Vyper	**Solidity**	
Arithmetic Over/Under Flows	Vul.	+		- Vyper includes built-in checks for over/under flows. - `SafeMath` library is required in Solidity to mitigate the attack.
Re-Entrancy	Vul.	+		- `@nonreentrant` decorator places a lock on functions to mitigate the attack. - `noReentrancy` modifier is required in Solidity.
Unchecked return values	Vul.	+		- It is already addressed in Vyper. - There is a need in Solidity to check return values explicitly.
Code readability	BP.	+		- No inheritance in Vyper enforces simpler design. - Solidity allows inline assemblies which is riskier and decreases readability.
Contract complexity	BP.	+		- 300 lines in Vyper have the same functionality as the Solidity with 500 lines.
Auditable	BP.		+	- Most of the auditing tools are able to analyze Solidity contracts.
Compatibility	BP.		+	- Majority of the current Ethereum projects are based on Solidity. - Developers are more familiar with Solidity than Vyper.
Production readiness	BP.		+	- Vyper is not as mature as Solidity in terms of stability, documentation, etc. - Solidity is adapted by a larger development community.

Table 4.1: Comparison of TokenHook implementation in Vyper and Solidity. The plus sign can be considered as an advantage. However, both versions of TokenHook offer the same level of security.

includes built-in security checks. Table 6.3 provides a comparison between the two from the perspective of TokenHook (see [102] for a broader comparison on vulnerabilities). Security and performance are advantages of Vyper. However, Vyper may not be a preferred option for production ("Vyper is beta software, use with care" [177]), most of the auditing tools only support Solidity,[3] and Solidity currently enjoys widespread implementation, developer tools, and developer experience.

4.6 Audit Results

To determine the extent to which TokenHook addresses the outlined issues, We conducted an experiment on code auditing tools using the Solidity implementation of TokenHook to understand the current state of automated vulnerability testing. Each tool uses different techniques (*e.g.,* Symbolic execution, Fuzzing, Static analysis, *etc.*) and reveal potential vulnerabilities. Our results illuminate the (in)completeness and

[3]Vyper support is recently added to some tools (*e.g.,* Crytic-compile, Manticore and Echidna). Slither integration is still in progress [121]

error-rate of such tools on one specific use-case (related work studies, in greater width and less depth, a variety of use-cases [45]). We did not adapt older tools that support significantly lower versions of the Solidity compiler (*e.g.,* Oyente). We concentrated on Solidity as Vyper analysis is currently a paid services or penciled in for future support (*e.g.,* Slither). The provided version number is based on the GitHub repository and tools without a version are web-based:

1. EY Smart Contract & Token Review by Ernst & Young Global Limited [69].
2. SmartCheck by SmartDec [158].
3. Securify v2.0 by ChainSecurity [172, 173].
4. ContractGuard by GuardStrike [89].
5. MythX by ConsenSys [29].
6. Slither Analyzer v0.6.12 by Crytic [101].
7. Odin by Sooho [160].

These audit tools are designed to detect vulnerabilities, inefficiencies, and security flaws in blockchain-based smart contracts. However, their guarantees, effectiveness, and accuracy vary based on the methodology they use and the depth of their analysis. Most smart contract audit tools do not offer strict guarantees of security. Instead, they provide risk assessments, vulnerability reports, and recommendations based on automated and, in some cases, manual analysis. A clean audit from a tool does not mean the contract is 100% secure. Undiscovered vulnerabilities may still exist. Some enterprise-level solutions (like EY Smart Contract Review) may offer service-level agreements (SLAs) or warranties for their audits, but most open-source tools do not.

Moreover, some tools may claim compliance with security standards like OWASP, SWC Registry, or ISO security standards, but this is not equivalent to a guarantee. Even if a tool provides a security assessment, developers are responsible for their contracts' security. Each tool has its own methodology, but they generally fall into the following categories:

4.6.1 Static Analysis

These tools analyze the code without executing it, detecting vulnerabilities based on predefined rules and pattern matching. For example, Slither, SmartCheck and Securify are fast, scalable, and useful for catching common vulnerabilities. But they may produce *false positives* (flagging safe code as vulnerable) or *false negatives* (missing actual vulnerabilities).

4.6.2 Symbolic Execution

These tools simulate contract execution for all possible input states, checking for vulnerabilities such as integer overflows, re-entrancy, and access control issues. For example, MythX and Odin are more thorough than static analysis, and can catch deep vulnerabilities. But they are computationally expensive and may miss real-world issues if execution paths are too complex.

4.6.3 Fuzz Testing (Fuzzing)

These tools generate random inputs to execute the contract dynamically, looking for unexpected behavior and vulnerabilities. For example, ContractGuard can find real-world exploitable bugs that static analysis might miss. But it May not cover all execution paths, and results depend on test input quality.

4.6.4 Manual Review Assistance

Some tools provide reports and recommendations that auditors use during manual reviews. For example, EY Smart Contract & Token Review is used by human auditors to catch complex logic errors that automation might miss. But it might be time-consuming and costly.

In summary, assessing the reliability of audit tools requires several considerations:

- *Cross-Checking with Multiple Tools*: Since no tool is perfect, running a contract through multiple tools (*e.g.,* Slither + MythX + manual review) increases confidence in the findings.
- *Analyzing False Positives & Negatives*: Some tools report *false positives* (safe code marked as vulnerable), requiring manual verification. Others may miss *false negatives* (real vulnerabilities left undetected).
- *Using Well-Tested & Reputable Tools*: Industry-trusted tools such as Slither, MythX, and Securify are more reliable due to their track record in audits.
- *Community & Open Source Validation*: Open-source tools (like Slither) undergo public scrutiny, making their methodologies transparent and more reliable.

- *Manual Audit as a Final Check*: Even the best automated tools should be complemented with a human review by security experts to catch logical flaws and vulnerabilities that automation might miss.

Tables 4.2, 4.3 and 4.4 summarize audit results by including best practices and security vulnerabilities.

To compile the list of 82, we referenced the knowledge-base of each tool [172, 158, 29, 89, 101, 130], understood each threat, manually mapped the audit to the corresponding SWC registry [30], and manually determined when different tools were testing for the same vulnerability or best practice (which was not always clear from the tools' own descriptions). Since each tool employs different methodology to analyze codes (*e.g.,* comparing with violation patterns, applying a set of rules, using static analysis, *etc.*), there are false positives to manually check. Many false positives are not simply due to old/unmaintained rules but actually require tool improvement.

After manually overriding the false positives, the average percentage of passed checks for TokenHook reaches to 99.5%. To pass the one missing check and reach a 100% success rate across all tools, we prepared the same code in Solidity version 0.8.4, however it cannot be audited anymore with most of the tools.

It is worth mentioning that the official SWC Registry does not categorize vulnerabilities into high-level groups like Coding, Financial, Fund Transfer, and Time-Based; instead, it lists individual vulnerabilities with unique SWC IDs. Moreover, we mapped 53 security vulnerabilities to the SWC IDs using NVivo [99]. While it is possible that different researchers might categorize or map the vulnerabilities to SWC IDs slightly

ID	SWC	**Vulnerability or best practice** Mitigation or recommendation	Security tools EY Token Review	 Smart Check	 Securify	 MythX (Mythril)	 Contract Guard	 Slither	 Odin
1	100	**Function default visibility** Specifying function visibility, external, public, internal or private		✓		✓	✓		✓
2	101	**Integer Overflow and Underflow** Utilizing the SafeMath library to mitigate over/under value assignments	⊕	!		✓	✓		✓
3	102	**Outdated Compiler Version** Using proper Solidity version to protect against compiler attacks	✓	✓	✓	✓	✓	✓	×
4	103	**Floating Pragma** Locking the pragma to avoid deployments using outdated compiler version		✓	✓	✓		✓	✓
5	104	**Unchecked Call Return Value** Checking call() return value to prevent unexpected behavior in DApps	⊕		✓	✓	✓	⊕	✓
6	105	**Unprotected Ether Withdrawal** Authorizing only trusted parties to trigger ETH withdrawals		!		✓		✓	✓
7	106	**Unprotected SELFDESTRUCT Instruction** Removing self-destruct functionality or approving it by multiple parties			✓	✓		✓	✓
8	107	**Re-entrancy** Using CEI and Mutex to mitigate self-function and cross-function attack		✓	⊕	⊕	⊕	✓	✓
9	108	**State variable default visibility** Specifying visibility of all variables, public, private or internal	✓	✓	✓	✓	✓		✓
10	109	**Uninitialized Storage Pointer** Initializing variables upon declaration to prevent unexpected storage access	✓	✓	✓	✓	✓	✓	✓
11	110	**Assert Violation** Using require() statement to validate inputs, checking efficiency of the code		✓		✓			✓
12	111	**Use of Deprecated Solidity Functions** Using new alternatives functions such as keccak256() instead of sha3()		✓		✓	✓	✓	✓
13	112	**Delegatecall to untrusted callee** Calling into trusted contracts to avoid storage access by malicious contracts		⊕	⊕	✓	✓	✓	✓
14	113	**DoS with Failed Call** Avoid multiple external calls where one error may fail other transactions	✓	✓		✓	✓		✓
15	114	**Transaction Order Dependence** Preventing race conditions by securing approve() or transferFrom()	⊕		✓	✓			✓
16	115	**Authorization through tx.origin** Using msg.sender to authorize transaction initiator instead of originator	✓	✓	✓	✓	✓	✓	✓
17	116	**Block values as a proxy for time** Not using block.timestamp or block.number to perform functionalities	✓	✓	✓	✓	✓		✓
18	117	**Signature Malleability** Not using signed message hash to avoid signatures alteration				✓			✓
19	118	**Incorrect Constructor Name** Using constructor keyword which does not match with contract name		✓		✓			✓
20	119	**Shadowing State Variables** Removing any variable ambiguities when inheriting other contracts			✓	✓	✓	✓	✓
21	120	**Weak Sources of Randomness from Chain Attributes** Using oracles as source of randomness instead of block.timestamp	✓	✓		✓	✓		✓
22	121	**Missing Protection against Signature Replay Attacks** Storing every message hash to perform signature verification				✓			✓
23	122	**Lack of Proper Signature Verification** Using alternate verification schemes if allowing off-chain signing				✓			✓
24	123	**Requirement Violation** Checking the code for allowing only valid external inputs		✓	✓	✓			✓
25	124	**Write to Arbitrary Storage Location** Controlling write to storage to prevent storage corruption by attackers		✓	✓	✓			✓
26	125	**Incorrect Inheritance Order** Inheriting from more general to specific when there are identical functions				✓			✓
27	126	**Insufficient Gas Griefing** Allowing trusted forwarders to relay transactions		✓					✓

Table 4.2: Auditing results of 7 smart contract analysis tools on TokenHook. ✓=Passed audit, ⊕=False positive, ×=Failed audit, Empty=Not supported audit by the tool, !=Informational, ◯=Tool specific audit (No SWC registry), BP=Best practice

ID	SWC	**Vulnerability or best practice** Mitigation or recommendation	Security tools EY Token Review	 Smart Check	 Securify	 MythX (Mythril)	 Contract Guard	 Slither	 Odin
28	127	**Arbitrary Jump with Function Type Variable** Minimizing use of assembly in the code		✓	✓	✓		✓	✓
29	128	**DoS With Block Gas Limit** Avoiding loops across the code that may consume considerable resources	✓	✓	✓	✓	✓	✓	✓
30	129	**Typographical Error** Using SafeMath library or performing checks on any math operation				✓			✓
31	130	**Right-To-Left-Override control character (U+202E)** Avoiding U+202E character which forces RTL text rendering			✓	✓	✓	✓	✓
32	131	**Presence of unused variables** Removing all unused variables to decrease gas consumption		✓	✓		✓	✓	⊕
33	132	**Unexpected Ether balance** Avoiding Ether balance check in the code (*e.g.,* this.balance == 0.24 Ether)		✓	✓		✓	✓	✓
34	133	**Hash Collisions With Variable Length Arguments** Using abi.encode() instead of abi.encodePacked() to prevent hash collision							✓
35	134	**Message call with hardcoded gas amount** Using .call.value()("") which is compatible with EIP1884		⊕	⊕	✓	✓		✓
36	135	**Code With No Effects** Writing unit tests to ensure producing the intended effects by DApps		✓					✓
37	136	**Unencrypted Private Data On-Chain** Storing un-encrypted private data off-chain		!					✓
38	○	**Allowance decreases upon transfer** Decreasing allowance in transferFrom() method	✓						
39	○	**Allowance function returns an accurate value** Returning only value from the mapping instead of internal function logic	✓						
40	○	**It is possible to cancel an existing allowance** Possibility of setting allowance to 0 to revoke previous allowances	✓	✓					
41	○	**A transfer with an insufficient amount is reverted** Checking balances in transfer() method before updating balances	✓					✓	
42	○	**Upon sending funds, the sender's balance is updated** Updating balances in transfer() or transferFrom() methods	✓						
43	○	**The Transfer event correctly logged** Emitting Transfer event in transfer() or transferFrom() functions	✓						
44	○	**Transfer an amount that is greater than the allowance** Checking balances in transferFrom() method before updating balances	✓						
45	○	**Risk of short address attack is minimized** Using recent Solidity version to mitigate the attack	✓				✓		
46	○	**Function names are unique** No function overloading to avoid unexpected behavior	✓					✓	
47	○	**Using miner controlled variables** Avoiding block.number, block.timestamp, block.difficulty, now, etc	✓	✓	✓	✓	✓	✓	
48	○	**Use of return in constructor** Not using return in contract's constructor		✓					
49	○	**Throwing exceptions in transfer() and transferFrom()** Returning true after successful execution or raising exception in failures		✓				✓	
50	○	**State variables that could be declared constant** Adding constant attribute to variables like name, symbol, decimals, etc						✓	
51	○	**Tautology or contradiction** Fixing comparison in the code that are always true or false						✓	
52	○	**Divide before multiply** Ordering multiplication prior division to avoid integer truncation						✓	
53	○	**Unchecked Send** Ensuring that the return value of send() is always checked						✓	
54	BP	**Too many digits** Using scientific notation to make the code readable and simpler to debug						✓	

Table 4.3: Continuation of Table 4.2.

.

ID	SWC	**Vulnerability or best practice** Mitigation or recommendation	Security tools						
			EY Token Review	Smart Check	Securify	MythX (Mythril)	Contract Guard	Slither	Odin
55	BP	**The decreaseAllowance definition follows the standard** Defining decreaseAllowance input and output variables as standard	✓						
56	BP	**The increaseAllowance definition follows the standard** Defining increaseAllowance input and output variables as standard	✓						
57	BP	**Minimize attack surface** Checking whether all the external functions are necessary or not	✓	✓	✓				
58	BP	**Transfer to the burn address is reverted** Reverting transfer to 0x0 due to risk of total supply reduction	✓						
59	BP	**Source code is decentralized** Not using hard-coded addresses in the code	✓	✓					
60	BP	**Funds can be held only by user-controlled wallets** Transferring tokens to users to avoid creating a secondary market	!						
61	BP	**Code logic is simple to understand** Avoiding code nesting which makes the code less intuitive	✓	✓					
62	BP	**All functions are documented** Using NatSpec format to explain expected behavior of functions	✓						
63	BP	**The Approval event is correctly logged** Emitting Approval event in the approve() method	✓						
64	BP	**Acceptable gas cost of the approve() function** Checking for maximum 50000 gas cost when executing the approve()	!						
65	BP	**Acceptable gas cost of the transfer() function** Checking for maximum 60000 gas cost when executing the transfer()	!						
66	BP	**Emitting event when state changes** Emitting Change event when changing state variable values	✓						
67	BP	**Use of unindexed arguments** Using indexed arguments to facilitate external tools log searching		✓			✓	✓	
68	BP	**ERC-20 compliance** Implementing all 6 functions and 2 events as specified in EIP-20	✓	✓	✓		✓	✓	
69	BP	**Conformance to naming conventions** Following the Solidity naming convention to avoid confusion						✓	
70	BP	**Token decimal** Declaring token decimal for external apps when displaying balances	✓						
71	BP	**Locked money (Freezing ETH)** Implementing withdraw/reject functions to avoid ETH lost		✓			✓	✓	
72	BP	**Malicious libraries** Not using modifiable third-party libraries		✓					
73	BP	**Payable fallback function** Adding either fallback() or receive() function to receive ETH		✓			✓		
74	BP	**Prefer external to public visibility level** Improving the performance by replacing public with external		✓				✓	
75	BP	**Token name** Adding a token name variable for external apps	✓						
76	BP	**Error information in revert condition** Adding error description in require()/revert() to clarify the reason					✓		
77	BP	**Complex Fallback** Logging operations in the fallback() to avoid complex operations					✓		
78	BP	**Function Order** Following fallback, external, public, internal and private order					✓		
79	BP	**Visibility Modifier Order** Specifying visibility first and before modifiers in functions						✓	
80	BP	**Non-initialized return value** Not specifying return for functions without output		✓			✓ 6		
81	BP	**Token symbol** Adding token symbol variable for usage of external apps	✓						
82	BP	**Allowance spending is possible** Ability of token transfer by transferFrom() to transfer tokens on behalf of another usercalc	✓						
		99.5% success rate in performed audits by considering 'False Positives' and 'Informational' checks as 'Passed'	100%	100%	100%	100%	100%	100%	97%

Table 4.4: Continuation of Table 4.3.

differently based on interpretation, we adhered to the mappings discussed and agreed upon within our research group and with other team members, ensuring consistency and alignment with established standards in the field. This approach helped ensure that our analysis was reliable and grounded in the common understanding of smart contract vulnerabilities.

4.7 Contributions

We provide a detailed study of ERC-20 token security, collecting and deduplicating applicable vulnerabilities and best practices, examining the ability of seven audit tools. Most importantly, we provide a concrete implementation of ERC-20 called TokenHook.[4] It is designed to be secure against known vulnerabilities, and can serve as a second reference implementation to provide software diversity. We test it at Solidity version 0.5.11 (due to the limitation of the audit tools) and also provide it at version 0.8.4. Vyper implementation is also provided at version 0.2.8 to make ERC-20 smart contracts more secure and easier to audit.

TokenHook can be used as template to deploy new ERC-20 tokens, migrate current vulnerable deployments, and to benchmark the precision of Ethereum audit tools. This chapter has focused more on the technical aspects of smart contracts and securing ERC-20 tokens. The most important contribution can be summarized as follows:

1. Identify security vulnerabilities specific to ERC-20 tokens.

[4]Compatible Solidity version of TokenHook (v. 0.5.11) deployed on Mainnet at `https://bit.ly/35FMbAf` and the latest Solidity (v. 0.8.4) on Rinkeby `https://bit.ly/3tI139S`. Vyper code at `https://bit.ly/3dXaaPc`.

2. Create the first online knowledge-base for ERC-20 tokens that can be referenced by developers and auditors.
3. Propose a secure ERC-20 implementation, TokenHook, that can be used as a template for real-world scenarios (*i.e.,* ICOs, dApps, *etc.*).
4. Consider best practices[5] in addition to the known vulnerabilities to improve performance of the token contract while prevent attacks.
5. Use audit tools to evaluate security of the proposed code.

4.8 Discussion

The main challenge in securing smart contracts is their immutable property. After deploying a buggy smart contract on the blockchain, it would not be possible to patch it as in classical applications. It becomes immutable and irreversible. Therefore, the focus of developers should be on avoiding known security flaws and considering required security measures before deploying smart contracts (and tokens as a subset of them). This motivates us to (i) comprehensively study all known vulnerabilities in smart contracts and particularly in ERC-20 tokens, (ii) systematize them into a set of 82 distinct vulnerabilities and best practices that can be referenced by developers and auditors, (iii) use our experience to provide a new secure implementation, TokenHook, that is freely available, and (iv) confirm the security of TokenHook by using seven different audit tools and comparing the results with the top 10 ERC-20 tokens.

[5]Best practices are techniques or rules that are accepted to develop the most effective smart contract.

In short, TokenHook enhances security of ERC-20 tokens and ensures stronger compliance with best practices compared to the sole surviving reference implementation (from OpenZeppelin). All these contributions are aimed at strengthening user protection by improving the security of the deployed contracts.

Chapter 5

A Shortfall in Investor Expectations of Leveraged Tokens

This chapter is based on the work "A Shortfall in Investor Expectations of Leveraged Tokens" [142] supervised by Dr. Jeremy Clark and published in the sixth international conference on Advances in Financial Technologies (AFT 2024), at the Austrian National Bank (OeNB) in Vienna, Austria.

5.1 Introduction

In this chapter, we present a summary of our research on Leveraged Tokens (LVTs) and discuss shortcomings such as deviation from leverage, higher fee and the risk of front-running. LVTs are emerging crypto-assets primarily issued by centralized exchanges. The concept is borrowed from leveraged ETFs (LETFs) in traditional financial markets, which offer higher gains (and higher losses) relative to price movements in the underlying asset. However, LVTs have been implemented differently from LETFs by exchanges in the crypto market, with variations across platforms.

We examine the mechanics and constituent components of LVTs, demonstrating that the lack of a standard has resulted in deficiencies and unexpected technical and economic outcomes. To identify existing problems, we analyze over 1,600 leveraged tokens from 10 issuers. Our analysis reveals that 99.9% of LVTs are centralized, with 80% lacking blockchain interaction, leading to transparency issues. Total supply information is difficult to access for 53% of them, and 41% appear inadequately backed at launch. Additionally, 97% of LVTs are vulnerable to front-running during well-known events, and they deviate from their stated leverage ratios more than LETFs, partly due to inconsistent re-leveraging processes and higher management fees. This work provides a framework for crypto investors, blockchain developers, and data analysts to gain a deep understanding of leveraged tokens and their impact on market dynamics, liquidity, and price movements. It also offers insights for crypto exchanges and auditors into the internal functionalities and financial performance of LVTs under varying market conditions.

5.2 Related Work

The study of leveraged financial instruments has been extensively explored in traditional markets, particularly with Leveraged ETFs (LETFs). Prior research has shown that LETFs suffer from compounding effects, which distort their intended leverage over time, particularly in volatile markets [169, 90, 116, 22, 16]. Frequent rebalancing has been proposed as a necessary mechanism to maintain tracking accuracy, although it may introduce additional market inefficiencies [21, 95, 111].

Moreover, the impact of LETFs on market liquidity and volatility has been widely studied, with findings indicating that daily rebalancing can create price distortions and increased trading volumes near market close [149, 155, 169, 90, 178, 181]. Despite the extensive analysis of LETFs, there has been limited academic work on Leveraged Tokens (LVTs) in the cryptocurrency market. Some research has highlighted how investors often misunderstand the risks associated with leveraged instruments, emphasizing the need for better regulatory guidance and investor education [111].

Our study builds upon these prior works by conducting an extensive analysis of over 1,600 LVTs, identifying key deficiencies in transparency, leverage consistency, and financial backing. By bridging the gap between traditional leveraged instruments and their crypto counterparts, our work contributes to the understanding of LVTs' structural issues and offers insights into their financial and regulatory implications.

5.3 Motivation for Studying LVTs

Since 2019, more than 1,600 LVTs have been issued by various crypto exchanges. The FTX exchange introduced the original concept by issuing 102 tokens on the blockchain. Trading volumes exceeded $1 million per day [36]. This upward trend

	Number of LVTs per year							Underlying Assets
Issuer	**2019**	**2020**	**2021**	**2022**	**2023**	**Total**	**Average**	
MEXC		162	116	102	76	456	114	217
AscendEX				228	112	340	170	94
Gate.io		116	96	54	8	274	69	123
Pionex		60	102	36	2	200	50	76
FTX	102	27				129	65	43
KuCoin			50	6	38	94	31	45
Binance		38	2			40	20	20
ByDFi				16	24	40	20	20
ByBit				34		34	34	17
Index Coop			2			2	2	2
Total	**102**	**403**	**368**	**476**	**260**	**1609**	**322**	**654**

Characteristics of LVTs			
Fund Source	**Blockchain Rep.**	**Fund Management Algorithm**	**Leveraged Product**
Internal	No	Off-Chain	Futures
Internal / External			
Internal	Yes		
	No		
External	Yes	On-Chain / Off-Chain	Debt

Table 5.1: *Left table:* Number of issued leveraged tokens per year, average per year, and number of unique underlying assets, which we collected manually from different sources. An underlying asset might be used to create multiple tokens with different leverage levels. *Right table:* Characteristics of issued LVTs by different issuers. Only 20% of tokens have been created on the blockchain. 99.9% of LVTs use derivatives as the leveraged product, which is offered by the same issuer (Internal for Pionex as of Jan 2023). Except for *Index Coop*, the rest of the issuers use off-chain fund management systems. Rebalancing triggers for *Index Coop* are still off-chain.

has continued, with other exchanges issuing approximately 32 new LVTs per month on average from January 2020 to November 2023 (see Table 5.1). Considering the growing trend of these tokens, it is important to study them from the following perspectives:

- *LVT attractiveness for investors:* Investment in LETFs nearly doubled in 2022 compared to 2021 [167], demonstrating an appetite for low-risk leverage, which is satisfied in the crypto market by LVTs. LVTs reduce liquidation risks compared to derivatives and margin trading. However, other characteristics (*e.g.,* volatility drag) must be understood to avoid unexpected value destruction. These risks are not unique to LVTs; they also exist in LETFs.[1]
- *LVT distinctive dynamics:* Understanding key aspects of LVTs, such as their underlying dynamics, peculiarities in product design, effects on crypto markets,

[1]In 2018, Credit Suisse had to close an LETF ETN after its price plunged 90% in one day. In another example, WisdomTree had to close its 3x oil products in March 2020 after their value was wiped out [167].

and investor suitability. Leveraged products can impact market dynamics, especially in highly volatile markets [156]. Technical details on LVT mechanics can be also useful for those involved in the design and implementation of LVTs to understand how these tokens affect liquidity and price movements, potentially influencing the robustness and reliability of trading algorithms.

- *Regulatory implications:* LVTs introduce new risks for market regulation, investor protection, and financial stability. Our work contributes to broader discussions on how to effectively regulate emerging financial technologies like LVTs. Additionally, since LVTs are often held by commercial firms requiring audited financial statements [43], auditors should understand how LVTs function, their risks, and how they perform under different market conditions.

It is worth mentioning that a single crypto-asset can be used to create multiple leveraged tokens, with each token representing a different leverage ratio or strategy. In our review of over 1,600 tokens, manual verification was not employed for every token; instead, data was aggregated through APIs, issuer websites, and platforms like cryptodatadownload.com. As a result, the analysis was conducted in clusters, grouping tokens based on common characteristics and leveraging automated tools to handle the large dataset efficiently.

5.4 Leveraged Token Mechanics

LVTs are tokenized representations of a leveraged fund whose value is derived from the value of a leveraged product. Leveraged products are essential components of LVTs, allowing issuers to form a leveraged fund and offer it as centralized or decentralized tokens. 99.9% of LVT issuers use crypto futures as the leveraged product. LVTs are intentionally designed with leverage as a core component of their architecture. They are aimed at outperforming the return of the underlying benchmark on a daily basis.[2] Let P_{t_n} represent the LVT price at calendar time t_n, expressed as:

$$P_{t_n} = P_{t_{n-1}} \left(1 + k\frac{\Delta S_{t_n}}{S_{t_{n-1}}}\right) \qquad n \in [1, 365), t \geq 0, k \in [-5, -0.5] \cup [0.5, 5] \tag{5.1}$$

S_{t_n} is the underlying price at time t_n, indexed by n, where n denotes the days of the year. The frequency of n does not have to be daily; it can be redefined in hours or minutes without any loss of generality. However, since daily returns are embedded in the LVT product design, n is effectively daily. P_{t_n} represents the price of the LVT at the close of trading day n. $S_{t_{n-1}}$ and $P_{t_{n-1}}$ are the initial prices of the underlying asset and LVT, respectively, at the beginning of trading day n (or at the end of trading day $n-1$). ΔS_{t_n} is the amount of change in the underlying price relative to the initial price. The constant variable k is the LVT multiplier (leverage), which can be defined as either a fixed or dynamic value, depending on the issuer. LVTs with fixed leverage can take values from the set $\{-5, -3, -2, -1, -0.5, 0.5, 2, 3, 5\}$, while

[2]Returns will be slightly lower after deducting fund management fees, accounting for market volatility, interest paid on borrowing, and other associated expenses.

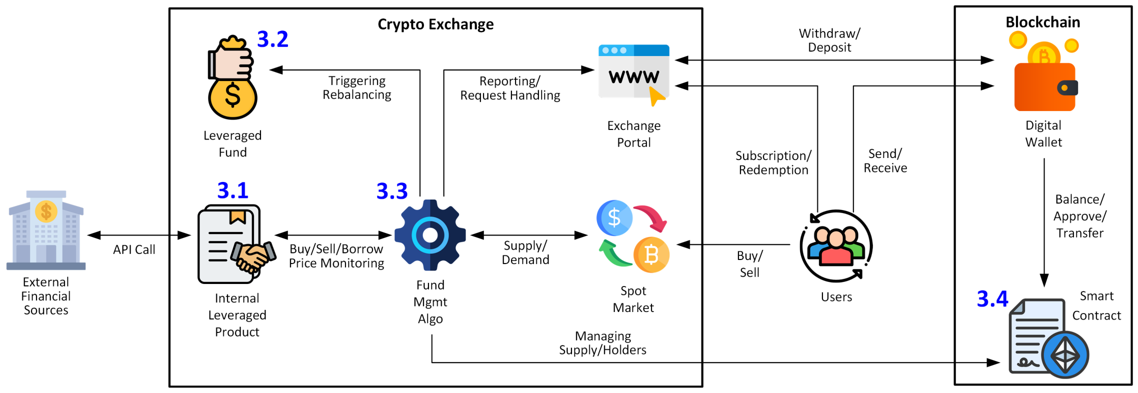


Figure 5.1: The constituent components of LVTs, according to the issuer's documentation. Some issuers have implemented them internally, resulting in missing blockchain components.

dynamic leverage fluctuates within the range $[-4.0, -1.25] \cup [1.25, 4.0]$.

To properly reflect the value of the fund through circulating LVTs, the notional value of all tokens must match the fund's notional value. As the price of the leveraged product changes over the trading day, the leverage of the fund gradually diverges from the stated ratio. The *fund management algorithm* resets this deviation by buying or selling the leveraged product on a daily basis. It also implements the logic of the LVT and defines how it should function in different market conditions. If an LVT is designed to be fully on-chain, the smart contract provides the functionality of this functionality, by extending common features of the ERC-20 standard. The constituent components of LVTs vary depending on the issuer. According to issuer documentation, the general components of a typical LVT are illustrated in Figure 5.1, followed by a brief explanation of the functionality of each component.

5.4.1 Leveraged Product

Financial managers of LETFs use leverage to open positions worth more than the required capital. In LVTs, leveraged exposure can be achieved by (i) opening positions in the crypto derivatives market, which provides up to 200x leverage, and (ii) borrowing capital from external sources, generating up to 10x leverage. LVT issuers typically do not use high multiples, offering tokens with up to 5x leverage. This allows them to choose either derivatives or debt as the leveraged product. 99.9% of issued LVTs use futures (a type of derivative), and only *Index Coop* uses the debt market to finance investments.[3] The desired outcome for *Index Coop*'s tokens is to generate future returns that outweigh the cost of borrowing. Other issuers that use derivatives aim to minimize dependency on other exchanges for buying and selling futures. They often offer the corresponding futures trading in their own portfolio to facilitate LVT management and reduce the cost of transactions between exchanges (compare the *Leveraged Product* and *Fund Source* columns in Table 5.1). For example, every issuer that launched BTC Long/Short tokens offers BTC-Perp futures as the underlying. Internal leveraged products facilitate LVT operations, such as adjusting fund positions, monitoring underlying price fluctuations, and triggering fund rebalancing.

[3]*Index Coop* uses money market protocols on Ethereum (e.g., Compound protocol) that offer permissionless borrowing and lending capabilities [33].

5.4.2 Leveraged Fund

It is a fund that derives its *notional value* from a basket of leveraged products.[4] The leveraged products provide leveraged exposure, upon which the value of the issued tokens is based. Let V_{t_n} represent the price of the k-times leveraged product V on trading day n, tracking the underlying asset S. The price of V_{t_n} differs from S as it carries k-times exposure. A leveraged fund L with a notional value of L_{t_n} can be formed by purchasing a basket of V, given by:

$$L_{t_n} = kV_{t_n}B_{t_n}\big(1 + (\rho_{t_n} + \phi_{t_n})\big) \quad t \geq 0, \rho_{t_0} = 0, \phi_{t_0} = 0 \tag{5.2}$$

Where B_{t_n} is the number of V units forming the fund at the rebalancing time t_n. ρ_{t_n} and ϕ_{t_n} represent management and futures funding fees, respectively. ρ_{t_n} is always negative, as the issuer deducts associated expenses from the fund's value. ϕ_{t_n} can be positive or negative, depending on the *futures funding fee* payments (more in Section 5.5). The sum of ρ_{t_n} and ϕ_{t_n} (*i.e.,* the total daily fee) varies per LVT issuer and ranges from 0.01 to 0.5 percent daily. Note that the change in the price of the leveraged product (V_{t_n}) is proportional to the underlying price (S_{t_n}), but does not vary based on the k multiplier. More precisely, $V_{t_n} = V_{t_{n-1}}(1 + R_{t_{(n-1)\to n}})$. In equation (5.2), L_{t_n} represents the financial value controlled by the leveraged fund L, which originates from the leveraged product V. The change in the price of V is proportional to S, but

[4]The notional value represents the total value of a financial instrument or contract at its full face value (i.e., controlled money by the financial instrument). The notional value is not typically exchanged between counterparties; instead, it serves as a reference point for calculating payments or obligations [83].

the value of L changes with respect to the k factor. In simple terms, V represents the price of the leveraged product, while L represents the amount of money that can be controlled using V.

Example 8. An issuer may arrange an *Ether long double-leveraged fund* by purchasing 4 *Ether-Perp long 2x futures* at \$1.5K (V_{t_0}). The 2x leverage of Ether-Perp allows the issuer to pay half of the Ether price, which is assumed to be \$3K (S_{t_0}). With zero fees in (5.2) at t_0 ($\rho_{t_0} + \phi_{t_0} = 0$), a leveraged fund L worth $2 \times \$1.5K \times 4 = \$12K$ can initially be formed (L_{t_0}). A 10% change in the price of Ether affects the price of futures by the same 10%, bringing it to \$1.65K (V_{t_n}). However, the notional value of the fund (L_{t_n}) changes according to $k = 2$, reaching $2 \times \$1.65K \times 4 = \$13.2K$. This demonstrates the effect of k on L compared to V.

LVTs are issued with a certain initial supply that can be adjusted through the *Subscription* and *Redemption* process. For added or removed tokens, the issuer offsets the notional value of the fund with the notional value of the tokens by buying or selling the corresponding amount of the leveraged product. Let N_{t_n} represent the total supply of a k-leveraged LVT at time t_n. The notional value of the issued LVTs (A_{t_n}) can be expressed as:

$$A_{t_n} = kN_{t_n}P_{t_n} \tag{5.3}$$

Equating (5.2) and (5.3) gives the total number of tokens (N_{t_n}) that should exist at the price of P_{t_n} at time t_n. Mathematically, this can be expressed as: $kN_{t_n}P_{t_n} = kV_{t_n}B_{t_n} \Rightarrow N_{t_n} = \frac{V_{t_n}B_{t_n}}{P_{t_n}}, \quad t \geq 0, P_{t_0} \geq 1$.

Example 9. Assume $P_{t_0} = \$10$ as the initial offering price of ETH2L in the previous example 8. P_{t_0} is typically set by the issuer at either \$1 or \$10 per token.[5] The initial supply of ETH2L at \$10 per token is $N_{t_0} = \frac{V_{t_n} B_{t_n}}{P_{t_n}} = \frac{\$1.5K \times 4}{\$10} = 600$. The notional value of all 600 tokens ($A_{t_0} = 2 \times 600 \times \$10 = \$12K$) is consistent with the notional value of the leveraged fund ($L_{t_0} = 2 \times \$1.5K \times 4 = \$12K$). Investors purchase a portion of this fund in the form of LVT, allowing them to generate twice the profit compared to the underlying Ether-Perp. Essentially, the value of the ETH2L token is derived from the *Ether long 2x leveraged fund*, which in turn is derived from the 4 positions in the *Ether-Perp long 2x futures*.

5.4.3 Fund Management System

As the price of the leveraged product (V_{t_n}) fluctuates over time, the notional value of the leveraged fund (L_{t_n}) changes, causing the leverage ratio of the LVT to deviate from the stated leverage. Let $\tilde{k_{t_n}}$ represent the realized leverage ratio that the notional value of the tokens (A_{t_n}) represents at time t_n, expressed as $\tilde{k_{t_n}} = \frac{L_{t_n}}{N_{t_n} P_{t_n}}$.

Example 10. Referring to the first trading day of ETH2L in examples 8 and 9, in which the price of Ether (S_{t_0}) increases by 10%, the notional value of the fund (L_{t_n}) changes to $2 \times \$1.65K \times 4 = \$13.2K$. The 2x leverage of the LVT increases its price by 20%, rising to \$12 (P_{t_n}) from the initial \$10 (P_{t_0}). Since the LVT supply remains constant at 600 tokens, the leverage ratio of the fund drops from 2x to 1.8x.

[5]Taken from the initial public offering (IPO) price of a SPAC (Special Purpose Acquisition Company), which is typically set at a nominal \$10 per unit. Unlike a traditional IPO, the SPAC IPO price is not based on the valuation of an existing business but rather on future income expectations.

$(\tilde{k_{t_n}} = \frac{L_{t_n}}{N_{t_n} P_{t_n}} = \frac{\$13.2K}{600 \times \$12} = 1.8\overline{3})$.

The analysis above suggests that with the change in the price of the underlying (S_{t_n}) and subsequently the price of the leveraged product (V_{t_n}), the notional value of the leveraged fund (L_{t_n}) changes, and the realized leverage ratio of LVTs ($\tilde{k_{t_n}}$) becomes higher or lower than the stated leverage k. Mathematically, if $\mathbb{E}[\tilde{k_{t_n}}]$ represents the expected change in $\tilde{k_{t_n}}$ in relation to the underlying price change (S_{t_n}), applying equations (5.1) gives us:

$$\mathbb{E}[\tilde{k_{t_n}}] = \frac{kV_{t_n} B_{t_n}(1 + R_{t_{(n-1)\to n}})}{kN_{t_n} P_{t_n}} = \frac{V_{t_n} B_{t_n}(1 + R_{t_{(n-1)\to n}})}{N_{t_n} P_{t_{n-1}}(1 + kR_{t_{(n-1)\to n}})} \tag{5.4}$$

As the number of tokens (N_{t_n}) remains constant while the price of the underlying changes at a rate of $(1 + R_{t_{(n-1)\to n}})$, the denominator of (5.4) changes k-times faster (or slower) than the numerator, resulting in positive or negative leverage skewness. This highlights the need to re-leverage the fund on a daily basis, a process managed by the *fund management*.[6]

Fund management is an off-chain algorithm (or on-chain for decentralized tokens) that dynamically adjusts the fund to maintain the leverage at the expected ratio. When the token's leverage increases, it sells some of the fund's positions to reduce the leverage and return it to the expected level. The majority of algorithms are off-chain with no interaction with the blockchain. The only on-chain instance is implemented by *Index Coop* [33]. In addition to correcting the leverage, the algorithm interacts

[6] Also referred to as *fund management agent*, *fund management party*, or *certified fund manager*.

with other components to adjust supply, update balances, monitor the price of the underlying, and deduct daily fees.

5.4.4 On-chain Contracts

For decentralized LVTs, a smart contract represents the leveraged fund on the blockchain. It is typically implemented as an ERC-20 token [70, 3, 154], allowing users to exchange LVTs on the blockchain without issuer intervention. As indicated in the *Blockchain Representation* column of Table 5.1, 80% of issuers have not created LVTs on the blockchain. As a result, this component is missing from Figure 5.1. The absence of a smart contract leads to several deficiencies, which are discussed in the next section.

5.5 Deficiencies of Existing LVTs

Due to the lack of a common standard in LVTs for defining the rebalancing process, data transparency and interoperability, these tokens are issued with varying features at the discretion of the issuer. We examine the characteristics of issued tokens per issuer and discuss the respective deficiencies as research questions RQ1 to RQ6.

RQ1: What information is visible to traders of an LVT?

Among the 10 LVT issuers, only *FTX* and *Index Coop* have created tokens on the Ethereum blockchain. FTX's management model was hybrid (*i.e.,* tradable decentralized tokens on a centralized exchange), while only *Index Coop*'s tokens are fully decentralized. The remaining 8 exchanges prefer to implement LVTs centrally and entirely internal.

Example 11. Binance leveraged tokens (BLVTs) are one of the centralized LVTs that are entirely accessible within Binance's ecosystem. They can be exclusively traded on Binance's spot market with no possibility of withdrawal. BLVTs are not even published on Binance's own blockchain (BNB Smart Chain) and are created more like a pseudo-crypto.[7]

Transparency in Total Supply

Total supply is used to calculate the Net Asset Value (NAV) of LVTs as a representation of the market's fair value. Due to imbalances in supply and demand, the market price of LVTs may deviate from the NAV, trading at a premium or discount. In the long run, LVT prices converge to the NAV due to a mechanism similar to arbitrage in traditional markets. Orders placed far from the NAV price lose or gain value over time. In the short run, however, investors use the NAV as a reference price when buying or selling, especially in bulk. The NAV of LVTs can be calculated by equating (5.2) and (5.3), with the current token supply N_{t_n}:

$$kN_{t_n}P_{t_n} = kV_{t_n}B_{t_n} \Rightarrow P_{t_n} = \frac{V_{t_n}B_{t_n}}{N_{t_n}} \quad t \geq 0, N_{t_0} \geq 1 \tag{5.5}$$

Example 12. In the previous examples (8) to (10), when the Ether price increases by 10% on day 2, the market price of ETH2L trades at \$12 (after a $2 \times 10\% =$

[7]A cryptocurrency that is not sufficiently decentralized [2].

20% increase), while its NAV price is ($\$1.65K \times 4)/600 = \$11$. ETH2L is, in fact, overvalued, and traders should wait for either (i) the arbitrage mechanism to play out and bring the LVT price down, or (ii) the next rebalancing schedule, which will match the fund's value with the notional value of the tokens.

For LVTs hosted on the blockchain, total supply is public and can be retrieved for NAV calculations. However, for centralized LVTs, investors must refer to the exchange's website. The total supply of tokens on some exchanges, such as AscendEX, Pionex, Gate.io, and ByDFi, does not appear to be public, making it difficult to verify the real value of LVTs (*i.e.,* 53% of all tokens). LVTs are *open-end funds* with a theoretically unlimited token supply.[8] Issuers can increase the supply based on market liquidity and demand for the token. Transparency in the number of issued tokens builds trust and reduces the risk of investment. Moreover, it addresses audit questions such as, Has the fund's value changed proportionately after increasing or decreasing the supply of LVTs? How much were the fund's value deviations in the previous audit period, and were they within the acceptable range?

Transparency in Transactions

Transactions on the blockchain show the flow of tokens and the movement of the fund. This enables investors to analyze transactions and ensure the expected functionality of LVTs.

[8]Open-end funds can issue an unlimited number of shares. The fund sponsor sells shares directly to investors and redeems them as well. The NAV per share of an open-end fund is calculated daily by dividing the total value of the fund (minus liabilities) by the total number of shares outstanding [18].

Example 13. We reviewed all *Mint* and *Burn* transactions of ETCBULL (FTX 3x Long Ethereum Classic) on the blockchain.[9] The analysis suggests that a total of 51,640,895 tokens were issued, and 24,207 were destroyed (*i.e.,* 51,616,688 circulating tokens). The trend of issuing tokens has taken on exponential velocity since April 2022. A total of 783,022 tokens were issued during the 960-day period between October 2019 and May 2022, while 50,857,873 tokens were issued over just 184 days from April to October 2022. In other words, 98.5% of all tokens were issued in just 6 months. Checking the recipient address indicates FTX's possible sub-wallet as the receiver. This sudden change in token supply warrants further investigation, especially given FTX's collapse shortly afterward.

This is just an example indicating the importance of transparency in LVT transactions. Transactions of centralized tokens are not public and only available to the issuer. Statistically, transactions of 80% of LVTs cannot be analyzed as we did in the above case.

Transparency in Token Holders

Holders of tokens created on the blockchain are public, allowing investors to check them as a measure of the token's liquidity. A small number of market participants reduces the token's liquidity and can make it more challenging to execute large orders. It may also lead to a wider bid-ask spread, increasing the cost of executing trades. Investors generally prefer assets with higher liquidity, narrower bid-ask spreads, and

[9]ETCBULL transactions on the Ethereum blockchain filtered for issued and deposited tokens to FTX's own address: `https://bit.ly/3MwHVqv`.

more market participants.

Example 14. 90% of XRPBULL (FTX 3x Long Ripple) tokens are distributed among four holders.[10] In another example, three accounts own 94% of all issued FTX 3x Long Cardano (ADABULL) tokens.[11]

Holding a large number of tokens by a limited number of accounts can noticeably elevate investment risk. One holder may decide to sell a significant number of tokens at any moment, potentially resulting in a notable price drop within a short period of time, leading to significant losses for smaller holders. Since most issuers do not publish the list and respective ownership percentages of centralized LVTs, the participants of 80% of LVTs remain uncertain.

Inability to Audit

Conducting audits ensures the security, functionality, and compliance of LVTs as claimed by the issuer. Unlike centralized LVTs, the code of tokens created on the blockchain is public, allowing auditors to identify vulnerabilities and associated risks. The security of these 20% decentralized LVTs can be evaluated by reviewing the code against industry best practices such as SWC [30]. Moreover, external audits are essential for LVTs to ensure they function as intended, such as verifying the output of methods when transferring tokens or updating balances. Auditors may also provide recommendations to improve the security, functionality, and compliance of LVTs. For centralized LVTs, the code is not public, requiring cooperation and willingness of the

[10]List of XRPBULL holders: `https://bit.ly/3MQurrc`.
[11]List of ADABULL holders: `https://bit.ly/3MUxPRZ`.

issuer to conduct a thorough review and quality assessment. Sharing the code and the results of an independent audit would improve transparency and help build trust between token holders and issuers.

RQ2: To what extent are LVTs locked to the exchange?

Interoperability with dApps and DeFi

In 2019, the total value locked in Decentralized Finance (DeFi) was approximately 700 million USD. As of April 2022, it stands at around 150 billion USD, representing more than 200% growth in less than three years [179]. Hosted LVTs on the blockchain (which usually comply with one of the fungible[12] token standards) facilitate interaction with DeFi systems, unlocking potential interoperability opportunities.

Example 15. FTX was able to employ blockchain interoperability to share its ETHBULL (3x Long Ether) with other exchanges such as Poloniex, Indodax, Bittrex, and Gate.io.[13] These exchanges owned 20%, 4%, 3%, and 2% of ETHBULL, respectively, and offered it on their platforms due to the possibility of interaction with DeFi.

In contrast, centrally issued LVTs cannot interact with other platforms and operate in isolation, preventing LVTs from moving across different platforms. Decentralized LVTs, on the other hand, foster connectivity and enable users to access a wide range of services and functionalities without being confined to a single exchange.

[12]Fungible (interchangeable) token standards are widely used by decentralized applications (dApps) to interact with other applications. ERC-20 is the dominant standard, followed by ERC-777 and ERC-1155.

[13]List of ETHBULL holders: `https://bit.ly/3MSZX7P`.

Inability to Custody

At first glance, the custody issue seems common to all assets on centralized exchanges. However, BTC buyers can transfer it to their personal wallets, while centralized LVTs remain locked within the exchange. Holders do not own the actual tokens but are simply betting on price movements. Some explain this custodial issue by viewing LVTs as "token contracts", though this term is not widely recognized nor aligns with the functionality of crypto derivatives. LVTs are essentially tokenized forms of derivative exposures.

Example 16. BTCUP and BTCDOWN are issued by Binance and track Bitcoin as the underlying asset. Unlike Bitcoin holders, owners of these tokens cannot withdraw or transfer them to their own digital wallets. In contrast, similar Bitcoin leveraged tokens were created by FTX on the Ethereum blockchain (known as BULL and BEAR tokens). Holders of these tokens still had the opportunity to exchange them on decentralized exchanges, such as Uniswap[14], shortly after FTX's bankruptcy. Holders could recover 80% of the token value on the first day of the bankruptcy, 50% on the second day, and up to 20% on the third day.

RQ3: Are the LVTs offered today adequately backed?

The simplest definition of an LVT is a tokenized leveraged fund. According to the documentation of LVT issuers, 99.9% of leveraged funds derive their value from a

[14]DEX transactions of FTX 3X Long Bitcoin Token (BULL) and FTX 3X Short Bitcoin Token (BEAR) on the Ethereum blockchain: `https://bit.ly/3pOh3uz`, `https://bit.ly/41Dal81`.

basket of positions in the futures market [6, 86, 9, 13, 14, 136, 39, 107, 133]. The issuer must either (i) offer futures trading in their portfolio, or (ii) open futures positions on other crypto exchanges and manage them systematically through APIs as the underlying asset fluctuates.[15] The question we raise, due to the lack of external audits, is to what extent LVT issuers have properly prepared futures contracts before launching LVTs. Have users invested in tokens that are properly backed, or are they simply trusting the issuer and potentially investing in tokens with no real value?

Missing Futures Product

Some issuers have launched LVTs without offering the corresponding futures products. While we cannot rule out the possibility that they hold the necessary futures positions on other exchanges, this raises concerns that these LVTs might not be adequately backed financially.

Example 17. AscendEX uses its own futures products and does not rely on futures products from other exchanges. However, they issued 3x/5x Long/Short Monero (XMR3L/S and XMR5L/S) without offering XMR perpetual contracts initially. To our knowledge, no XMR futures products were available on the market from any exchange to be used as leveraged products at the time of the launch of these XMR tokens.

[15]Among 10 LVT issuers, Pionex used the *Binance Broker API* for its *Futures Arbitrage Bot*, but it has been terminated since June 2021 [132]. After reviewing Pionex's documentation [133], it remains unclear whether *Binance Futures* is still used as the leveraged product for LVTs. However, Pionex launched its own futures product in January 2023. If they no longer use *Binance Futures* and rely solely on their own futures product, it appears that 148 LVTs did not have corresponding futures contracts at the time of launch (e.g., ETC3L/S, ZRX2L/S, XLM3L/S).

Issuer	Delayed Futures Launch	Missing Futures Product	Total Delayed or Missing	Total Launched LVTs	% of unbacked LVTs
	(A)	(B)	(C)	(D)	(E)
AscendEX	36	214	250	340	74%
Pionex	0	148	148	200	74%
MEXC	176	12	188	456	71%
ByDFi	16	0	16	40	40%
FTX	18	0	18	129	14%
Gate.io	18	16	34	274	12%
Binance	0	0	0	40	0%
ByBit	0	0	0	34	0%
KuCoin	0	0	0	94	0%
Index Coop	0	0	0	2	0%
Total	**264**	**390**	**654**	**1609**	**41%**

Leverage	Rebalancing Schedule		Fee deduction	
	Regular Daily	Interim	Daily Schedule	Expense Ration
(a)	(b)	(c)	(d)	(e)
Fixed	02:30 UTC	10% TBR / OOR	00:00 UTC	0.500%
Fixed / Variable	00:00 UTC+8	10% TBR	00:00 UTC+8	0.030%
Fixed	00:00 UTC	15% TBR		0.100%
	08:00 UTC+8			0.030%
	02:00 UTC	10% TBR	00:00 UTC	0.030%
	00:00 UTC-4	10% TBR / OOR	00:00 UTC+8	0.300%
Variable	N/A		00:00 UTC	0.010%
	00:00 UTC			0.005%
Fixed / Variable	08:00 UTC+8	14% TBR	23:45 UTC+8	0.045%
Fixed	00:00 UTC	20% TBR	00:00 UTC	0.023%

Table 5.2: *Left table:* Number of issued LVTs with delayed or missing futures products, analyzed using historical data and undisclosed information. To our knowledge, 41% of the issued LVTs did not have sufficient financial backing at the time of launch. *Right table:* Rebalancing and fee deduction schedules, which we collected manually from the issuers' websites. Regardless of leverage type, rebalancing is performed daily at different times. Additionally, Threshold-Based (TBR) or Out of Range (OOR) rebalancing methods are used to trigger interim rebalancing. Fund expenses are also deducted at various times with variable percentages.

The above example is one of 390 issued tokens lacking a corresponding futures product (see column B of Table 5.2). Based on available historical data and information from the issuer's website, 24% of LVTs did not have the necessary futures product offered by the same issuer and instead relied on futures from other exchanges.

Delayed Futures Product

Missing futures products are not the only issue with centralized LVTs. For 264 tokens, the corresponding futures product was only offered after the issuance of the tokens (see column A of Table 5.2). In other words, at the time of the launch of 17% of LVTs, the required futures may not have existed. According to the LVT documentation on issuers' websites, these issuers did not disclose using futures from other crypto exchanges. Internal futures trading was introduced later, after the token was launched.

Example 18. MEXC issued 3x Long/Short Cardano (ADA3L/S) in February 2020, while ADA-Perp was only launched in July 2020, resulting in a 154-day delay. They did not disclose using futures from other exchanges, indicating the fund might have been operating without financial backing during this period.

According to available information, on average, 41% of LVTs have missing or delayed futures products (see column E of Table 5.2). The main financial issue with LVTs is the lack of transparency in the fund management system. Centralized LVTs function like a black box to investors and are fully managed by the issuer. Even for tokens with proper futures backing (*e.g.,* Binance, ByBit, and KuCoin), investors can rely solely on numeric assertions made on the issuer's website.

RQ4: What are the possibilities of front-running?

Front-running is an illegal practice in the equity market where non-public information is used to purchase shares of a company before the price moves [64, 191, 7]. For instance, FINRA[16] announced a \$700K fine against *Citadel Securities*[17] in 2020 for front-running activities between 2012 and 2014 [119]. In the design of LVTs, certain well-known events can be exploited by traders to benefit from anticipated price movements. They can engage in similar front-running practices that may impact the price of the underlying asset and the token itself. We review these events and explore possible front-running scenarios as follows.

[16]The Financial Industry Regulatory Authority (FINRA) is a government-authorized organization that oversees U.S. equity markets by regulating member brokerage firms and exchange markets.

[17]Citadel Securities is the largest designated market maker on the New York Stock Exchange (NYSE).

Event I: Impending Fund Rebalancing

To keep the leverage at the stated ratio, issuers perform periodic rebalancing. This can be triggered at predefined intervals (*e.g.,* every day, every 8 hours, or every n blocks), or upon meeting certain conditions (*e.g.,* after exceeding a specific threshold). All LVT issuers perform regular daily rebalancing and trigger interim rebalancing in volatile markets (see columns B and C of Table 5.2). They may trigger rebalancing when the underlying asset's price fluctuates by more than $X\%$, or when the leverage passes a threshold. The *fund management algorithm* governs the rebalancing process, adjusting futures positions and restoring the leverage ratio to the target level.

The number of contracts that must be bought or sold to restore the leverage is predictable, making front-running possible. Let ΔB_{t_n} represent the number of required futures contracts to rebalance the fund at time t_n. ΔB_{t_n} can be easily calculated by considering the return of the underlying asset from time t_{n-1} to t_n ($R_{t_{(n-1)\to n}}$). The number of required futures contracts to restore the fund leverage can be calculated by subtracting the notional value of the tokens (equation 5.3) from the notional value of the fund (equation 5.2):

$$\Delta B_{t_n} = (1 + kR_{t_{(n-1)\to n}})kN_{t_{n-1}}P_{t_{n-1}} - (1 + R_{t_{(n-1)\to n}})kV_{t_{n-1}}B_{t_{n-1}} - (\rho_{t_n} + \phi_{t_n})L_{t_n} + \epsilon_{t_n}$$

This equation is quadratic and can be simplified as $ax^2 - bx - c$ for long tokens, and $-ax^2+bx-c$ for short tokens. When the underlying return is positive ($R_{t_{(n-1)\to n}} > 0$), ΔB_{t_n} is always positive ($a > 0$), and when the return of the underlying is negative,

ΔB_{t_n} is negative as well ($a < 0$). In simpler terms, for long LVTs, futures exposure must be increased when the underlying price is rising, and decreased when the underlying price is falling.

There is also a fund expenses term $(\rho_{t_n} + \phi_{t_n})L_{t_n}$, which is usually deducted from the fund's value to cover operating expenses. However, this term can turn positive when the received funding fees (ϕ_{t_n}) exceed the fund expenses (ρ_{t_n}). ϵ_{t_n} represents a disturbance term that captures the effects of news or shocks in the underlying. Since rebalancing is a predictable event, by buying or selling ΔB_{t_n} of the leveraged product, other traders can front-run the trade, potentially impacting the price of the token or even the underlying asset.

Example 19. Consider the following sequence in which Alice calculates ΔB_{t_n} to potentially front-run the rebalancing trade:

1. Alice checks the issuer's website for the upcoming rebalancing of BTC5L (5x Long Bitcoin). She notices the next daily rebalancing is scheduled for 00:00 UTC.
2. Alice calculates the number of contracts that will be bought or sold by the issuer to maintain the 5x target leverage of BTC5L.
3. Alice front-runs the rebalancing trade by placing an order just before 00:00 UTC (ahead of the rebalancing trade). If she anticipates that the algorithm will buy Bitcoin futures, she may buy Bitcoin futures expecting increased demand, driving up the price, and giving her the opportunity to sell futures at higher prices. Conversely, if BTC5L will be selling the fund's positions, leading to

increased supply, she may sell Bitcoin futures.

Front-running in the above example may not only manipulate the price of Bitcoin futures but also inflate the price of BTC5L. Consider the following scenarios (A) and (B), with corresponding calculations in Table 5.3.

- Alice calculates the *Basket Delta* of BTC5L prior to the rebalancing schedule and realizes that the algorithm will purchase 594.21 new contracts at $33,000 (Scenario A in Table 5.3). Assuming this purchase increases the price of Bitcoin futures by 1%, she can buy contracts just before the rebalancing trade at $33,000 and sell them afterward at $33,330. A 1% increase in the underlying price inflates the token price to $15.05. This provides Alice an additional opportunity to buy the token for $15 before the rebalancing and sell it at a higher price afterward.
- The effect of Alice's strategy in the previous scenario may be amplified if many traders engage in front-running. The increased demand may raise the price of Bitcoin futures even before the rebalancing trade. If the influx of other traders pushes Bitcoin futures up by 1%, and the rebalancing trade further increases the price by another 1%, this secondary effect could also inflate the token price further and create more price distortion (Scenario B in Table 5.3).

Event II: Management Fee Deduction

Similar to LETFs, daily fees and expenses are deducted from the leveraged fund to cover associated costs. The fee rate and daily schedule vary by issuer (see columns

Initial State		No Front-running	Front-running (Scenario A)	Front-running (Scenario B)
Funding Fee (3 * 0.01%)	$9,000	-$9,900	-$9,900	-$9,900
Management Fee (0.3%)	-$90,000	-$99,000	-$99,000	-$99,000
Futures Basket	1,000	1,590.91	1,590.91	1590.91
Futures Price	$30,000	$33,000	**$33,330**	**$33,660**
Fund Value	$30,000,000	$52,500,030	$53,025,030	$53,550,030
Issued Tokens	600,000	700,000	700,000	700,000
Leverage	5.00	5.00	5.05	5.10
BTC5L Price	$10	$15	**$15.05**	**$15.10**
Tokens Value	$6,000,000	$10,500,000	$10,535,000	$10,570,000
Basket Delta	2.70	594.21		

Table 5.3: The effect of front-running on the price of the underlying asset and the token during fund rebalancing is shown. The assumption in this example is a downtrend, where the fund initially receives funding fees from short traders and later pays funding fees as the price moves in the long direction of the token.

D and E of Table 5.2). Management fees are deducted at specific times, allowing adversaries to exploit this known event, potentially coinciding with rebalancing. The simultaneous occurrence of events I and II can intensify the front-running effect during the rebalancing process.

Example 20. Consider the same sequence as the previous example, where Alice calculates the *Basket Delta* of BTC5L at the same time as the management fee deduction. The issuer's withdrawal of $99,000 (Management fee row in Table 5.3) reduces the fund's value. To compensate, $99,000/$33,000 = 3 additional contracts need to be purchased. The coincidence of these two events causes the rebalancing algorithm to slightly increase demand by purchasing 597.21 contracts instead of 594.21.

Event III: Futures Funding Fee Exchanges

Funding fee is a mechanism in Perps to converge the price of contracts with the price of the underlying crypto. It is calculated based on the notional value of the futures position and is exchanged between short and long traders who keep their positions open. Shorts pay longs when the funding rate is negative, and longs pay shorts when the rate is positive(see 2.7.1 for more details). The intervals for *Funding fee exchange* are public and displayed on the issuer's website, typically occurring every 8 hours at 00:00 UTC, 08:00 UTC, and 16:00 UTC. Since the fund is composed of futures, it either pays or receives funding fees at these times. This predictably increases or decreases the value of the leveraged fund, which can be exploited to amplify the effects of front-running.

The impact of front-running can be exacerbated when events I, II, and III occur simultaneously. Such concurrency may force the algorithm to buy or sell more contracts than would be required for fund rebalancing alone (*i.e.,* $\Delta\tilde{B_{t_n}} = \sum_{n=1}^{3} \Delta B_{t_n}$).

Example 21. As calculated in the *Basket Delta* row of Table 5.3, in the first rebalancing cycle, an additional 2.70 futures contracts are required to cover the 0.3% daily management fee deduction and 0.03% daily funding fee exchange. This means that 2.7 more contracts will be added if events II and III coincide with event I. The impact on basket delta can be even more significant as the value of the fund increases in a volatile market.

To mitigate front-running in LVTs, issuers should avoid rebalancing the fund on predetermined schedules. Techniques such as intraday or randomized rebalancing, or algorithmic trading, can help reduce the visibility of rebalancing trades.[18]

[18]Iceberg orders, which are large orders broken into smaller lots, are a sophisticated trading algorithm used to execute rebalancing trades in smaller, more discrete chunks over time.

Only Binance avoids regular daily rebalancing, instead triggering it when the underlying price fluctuates more than 10% or when leverage falls outside the range of $[-4.0, -1.25] \cup [1.25, 4.0]$. This means 97% of current LVTs perform fund rebalancing at specific intervals, increasing the likelihood of daily front-running (see column B of Table 5.2).

RQ5: How well do LVTs track their asserted leverages?

The leverage ratio of LVTs is determined by the issuer and can be either variable (dynamic) or fixed. If an LVT uses the *Underlying+Leverage+Long/Short* naming convention, the leverage is most likely fixed. The *Underlying+Up/Down* format is used for LVTs with variable leverage.

Example 22. KuCoin has issued ETH3L as a 3x long token tracking Ether as the underlying. Binance similarly offers ETHUP and ETHDOWN tokens with a target leverage in the range of [1.25, 4] and [-4, -1.25], respectively.

Very high leverage factors such as ±10x or ±15x are not common in currently issued LVTs, as the majority of them provide ±3x leverage. Low leverage is aimed at minimizing losses and extending the liquidation point during periods of high volatility. Highly leveraged LVTs lose value in the same proportion as the underlying asset and may not be attractive to investors. Crypto exchanges advertise LVTs as an investment vehicle providing leveraged exposure to crypto-assets with minimal liquidation risk. However, LVTs with high leverage factors defeat this promise.

Market	Symbol	Issuer	Underlying Index/Asset	Direction	Target Leverage	Leverage Deviation
Equity	SPXL	Direxion	S&P500 (SPX)	Long	3	0.13%
	SPXS			Short	3	0.21%
	SSO	ProShares		Long	2	0.08%
	SDS			Short	2	0.19%
	UPRO			Long	3	0.12%
	SPXU			Short	3	0.28%
	QLD		NASDAQ-100 (NDX)	Long	2	0.07%
	QID			Short	2	0.23%
	TQQQ			Long	3	0.13%
	SQQQ			Short	3	0.28%
Crypto	BTC3L	MEXC	Bitcoin (BTC-Perp)	Long	3	1.22%
	BTC3S			Short	3	1.15%
	BTC3L	ByBit		Long	3	1.21%
	BTC3S			Short	3	1.10%
	BTC3L	KuCoin		Long	3	1.84%
	BTC3S			Short	3	1.58%
	BTC3L	Gate.io		Long	3	2.12%
	BTC3S			Short	3	2.10%

Table 5.4: Comparison of leverage deviation in sample LVTs and LETFs from May 2022 to May 2023. A total of 268 data samples were collected from the issuers' websites.

Inconsistency of Fixed Leverage

Approximately 16% of LVTs are issued with variable leverage, fluctuating in the range of $[-4.0, -1.25] \cup [1.25, 4.0]$. Additionally, 9%, 59%, and 11% of LVTs have fixed 2x, 3x, and 5x leverage ratios, respectively. As shown in Table 5.4, LVTs with fixed leverage may not always provide exactly the promoted leverage. One aspect of risk involves leverage deviation (also called tracking error). Furthermore, some issuers do not rebalance the fund in the same way.

Example 23. MEXC and KuCoin adjust the leverage of only the tokens that have lost value. Consider a volatile market where Bitcoin loses 10% in a day. These issuers adjust only the leverage of BTC3S, while the leverage of BTC3L remains unchanged, as it gained value [86, 106]. For example, if the price of Bitcoin is \$30K and the fund holds 600 contracts, the fund's initial value is \$18M. Assuming 600K issued tokens at an initial offering price of \$10, the target leverage is 3x (*i.e.,* $k = (600 \times \$30K)/(600K \times \$10) = 3$). A 10% increase in Bitcoin's price changes the leverage of BTC3L and BTC3S to 2.53x and 4.71x, respectively. However, these exchanges correct the leverage of BTC3S to prevent further capital loss in case of more price decline. As a result, this rebalancing process undervalues the BTC3L fund (*i.e.,* inflates the value of BTC3L). Instead of only rebalancing the losing side, both BTC3L and BTC3S positions should be adjusted simultaneously to bring the leverage back to 3x as advertised by the issuer.

We compared the leverage deviation of Bitcoin LVTs with LETFs over the course of a year. Analysis details are provided and Table 5.4. As can be seen, LVTs exhibit higher leverage deviations than similar products in the equity market. This issue becomes more apparent when comparing the standard deviation of returns in the equity and crypto markets.

Leverage deviation leads to underperformance or overperformance of tokens, causing investors to experience returns that deviate from the intended amplification effect of LVTs. This is particularly important in light of previous research on LETF returns, which shows that LETFs, on average, do not negatively impact investor short-term returns [113]. Results indicate that the daily return distribution using real-world historical data is significantly more leptokurtic than the normal distribution. However, in LVTs, returns tend to have a wider or flatter shape (platykurtic) due to higher leverage deviation.

Disadvantages of Variable Leverage

LVTs with variable leverage aim to: (i) minimize the impact of volatility drag, and (ii) reduce the possibility of front-running (as discussed in Section 5.5). LVTs are advertised as an investment vehicle that amplifies returns relative to a certain multiplier, although this factor changes constantly in tokens with variable leverage. This introduces an additional risk dimension, requiring regular monitoring and adjustment of positions as the leverage fluctuates.

Additionally, these types of tokens rebalance on an as-needed basis with no predetermined schedules. Rebalancing can be triggered by (i) a sudden fluctuation in the underlying price (such as more than 15%), (ii) exceeding the expected leverage range (such as above 4x or below 1.25x for long LVTs), and (iii) handling subscription or redemption requests, which change the total supply. One disadvantage of this type of rebalancing is that funds can remain undervalued or overvalued for extended periods.

Example 24. The rebalancing events of BTCUP (a Long BTC LVT by Binance) over the past 3 years are listed in Table 5.5. A rebalancing event occurred on 03-Jan-2023, 204 days after the previous one on 13-Jun-2022. During those 204 days, no rebalancing event was triggered because the changes in Bitcoin's price did not exceed the 10% threshold limit, and the fund's leverage fluctuated within the expected range of [1.25x, 4x]. During this period, the fund's value was much lower (or higher) than the amount required to support the value of issued tokens, but no rebalancing occurred.

#	Rebalancing Time	Gap (days)	Leverage Before	Leverage After	Leverage Adjustment
1	03-Apr-2024 04:44	22	2.004	1.582	0.422
2	12-Mar-2024 02:26	5	1.740	1.529	0.211
3	06-Mar-2024 06:44	10	2.492	2.716	-0.224
4	25-Feb-2024 20:59	11	2.266	2.033	0.233
5	14-Feb-2024 11:01	24	1.872	1.980	-0.108
6	22-Jan-2024 19:28	21	2.064	2.017	0.047
7	02-Jan-2024 07:39	71	2.257	2.484	-0.227
8	23-Oct-2023 23:38	294	1.924	1.770	-0.154
9	03-Jan-2023 22:45	204	1.996	1.511	-0.485
10	13-Jun-2022 10:51	143	3.766	2.469	-1.297
11	21-Jan-2022 01:17	22	3.444	2.529	-0.915
12	30-Dec-2021 19:59	52	3.120	2.514	-0.606
13	08-Nov-2021 07:55	25	2.269	1.807	-0.462
14	14-Oct-2021 22:45	8	2.566	2.205	-0.361
15	06-Oct-2021 15:26	15	1.935	2.483	0.548
16	21-Sep-2021 20:04	11	3.033	2.519	-0.514
17	10-Sep-2021 18:37	147	3.208	2.521	-0.687
18	16-Apr-2021 12:24	0	2.029	1.956	-0.073

Table 5.5: Binance BTCUP rebalancing history from May 2021 to April 2024. The "Gap" column indicates the number of days between rebalancing events.

Undervalued funds may benefit the issuer, while investors hold inflated tokens.

Another disadvantage of dynamic leverage is the imbalance in rebalancing triggers. Some issuers initiate the rebalancing at different ranges for long and short tokens.

Example 25. Pionex triggers rebalancing when the leverage of long tokens exceeds the range of $[2.2, 4.0]$, while this range is $[1.8, 4.8]$ for short tokens [131]. This inconsistency increases the complexity of position management, leading to unfavorable outcomes for investors.

Dynamic leverage may reduce the chances of front-running in LVTs but at the expense of token transparency and complicating position management. Each issuer puts in place its own algorithm for rebalancing of LVTs with dynamic leverage, since there is no uniform standard for the rebalancing process. This might be confusing for investors who switch from one issuer to another one, expecting similar performance.

Market	Symbol	Issuer	Underlying Index/Asset	Leverage	Annual Expense Ratio	Daily Expense Ratio
Equity	IVV	BlackRock	S&P500	+1x	0.0300%	0.000119%
	VOO	Vanguard	S&P500	+1x	0.0300%	0.000119%
	SPY	SSGA	S&P500	+1x	0.0945%	0.000375%
	QQQ	Invesco	NASDAQ-100	+1x	0.2000%	0.000794%
	TQQQ	ProShares	NASDAQ-100	+3x	0.8600%	0.003413%
	SH	ProShares	S&P500	-1x	0.8800%	0.003492%
	SSO	ProShares	S&P500	+2x	0.8900%	0.003532%
	SDS	ProShares	S&P500	-2x	0.9000%	0.003571%
	SPXU	ProShares	S&P500	-3x	0.9000%	0.003571%
	UPRO	ProShares	S&P500	+3x	0.9100%	0.003611%
	PSQ	ProShares	NASDAQ-100	-1x	0.9500%	0.003770%
	QLD	ProShares	NASDAQ-100	+2x	0.9500%	0.003770%
	QID	ProShares	NASDAQ-100	-2x	0.9500%	0.003770%
	SQQQ	ProShares	NASDAQ-100	-3x	0.9500%	0.003770%
	SPXL	Direxion	S&P500	+2x	1.0000%	0.003968%
	SPXS	Direxion	S&P500	-2x	1.0800%	0.004286%

Market	Symbol	Issuer	Underlying Index/Asset	Leverage	Annual Expense Ratio	Daily Expense Ratio
Crypto	BTC3L	ByBit	Bitcoin	+2x to +4x	1.8250%	0.005000%
	BTC3S	ByBit	Bitcoin	-2x to -4x	1.8250%	0.005000%
	BTCUP	Binance	Bitcoin	+1.25 to +4x	3.6500%	0.010000%
	BTCDOWN	Binance	Bitcoin	-1.25 to -4x	3.6500%	0.010000%
	BTC3L	Pionex	Bitcoin	+2.2x to +4x	10.9500%	0.030000%
	BTC3S	Pionex	Bitcoin	-2.2x to -4x	10.9500%	0.030000%
	BTC3L	ByDFi	Bitcoin	+3x	10.9500%	0.030000%
	BTC3S	ByDFi	Bitcoin	-3x	10.9500%	0.030000%
	BTC3L	KuCoin	Bitcoin	+3x	16.4250%	0.045000%
	BTC3S	KuCoin	Bitcoin	-3x	16.4250%	0.045000%
	BTC3L	MEXC	Bitcoin	+3x	36.5000%	0.100000%
	BTC3S	MEXC	Bitcoin	-3x	36.5000%	0.100000%
	BTC3L	Gate.io	Bitcoin	+3x	36.5000%	0.100000%
	BTC3S	Gate.io	Bitcoin	-3x	36.5000%	0.100000%
	BTC3L	AscendEX	Bitcoin	+3x	109.5000%	0.300000%
	BTC3S	AscendEX	Bitcoin	-3x	109.5000%	0.300000%

Table 5.6: Comparison of the daily expense ratio in the equity and crypto markets. Each day, the daily fee is deducted from the price of ETFs/LETFs/LVTs, which negatively impacts the ROI. Therefore, investing in assets with lower daily rates (green rows) is less risky with higher return.

RQ6: Are LVT fees in-line with traditional LETFs?

Issuers of LETFs/LVTs charge daily fees to compensate for the associated cost of operating the fund. Over the past few years, these fees have generally come down on average, where the Management Expense Ratio (MER) for traditional ETFs and LETFs annually averaged 0.45% and 0.95%, respectively. We investigated the daily fees for Bitcoin LVTs, and summarized our findings in Table 5.6.

The annual MER for these LVTs ranges from 1.83% to 36.5% depending on the issuer. By contrast, the standard deviation of MER in LETFs and LVTs is 0.38% and 34.21%, respectively. That would indicate that the fees for LVTs are less predictable and more volatile than for LETFs-by as much as 90-fold. This impacts the overall expense ratio and net returns of LVTs. Additionally, high fees typically characterize a market in its development stage with a limited issuer base. This may signal some risk for investors who would expect to see low fees once the market starts to mature and face some competition.

5.6 Contributions

In particular, the purpose of this chapter is not limited to introducing LVT but conducting an in-depth study from the underlying, interaction with blockchains, types of leveraged products, and algorithms for managing funds. We investigated more than 1,600 LVTs offered by 10 different issuers. Then, with a careful explanation of the mechanics and constituent components of LVT, readers would be able to understand how leveraged fund works, the rebalancing mechanism, and smart contracts in LVTs. Based on this, we answer the following six research questions about LVTs:

RQ 1: What information is visible to traders of an LVT? In some of the centralized LVTs, important parameter such as the total supply, transactions, and holders not available to investors. Total supply an essential parameter for calculation of the tokens' fair value. It is also used by auditors when assessing the consistency and efficiency of LVTs. Transparency of the transactions helps investors, auditors, and all participants of the LVT ecosystem to analyze token flow, detect suspicious activity, enhance security, ensure compliance, and verify whether tokens are working the way they should be. Centrally issued tokens do not show how many holders there are, and their investment risks are much higher compared to their decentralized counterparts.

RQ 2: To what extent are LVTs locked to the offering exchange? The inability to self-custody of centralized LVTs raises more concerns compared to their decentralized counterpart. It also increase the risk of investment in those crypto-assets. Another important advantage of tokens issued on the blockchain is their good interoperability with other dApps, crypto exchanges, and in general with the DeFi ecosystem. The LVTs interacting with DeFi enable tapping into new opportunities of participation in a more open and transparent financial system operating without intermediaries.

RQ 3: Are the LVTs offered today adequately backed? The value of LVTs is derived from the value of a leveraged fund, which in itself is derived from the value set by futures. As no futures existed at such a point when the LVTs were offered, some investors may be investing in LVTs, which have either an inadequate or no financial backing. The analysis of publicly available historic data from the issuer's website gives evidence that on average, out of each 5 LVTs issued, 2 were seemingly not fully financially covered at the time of their issuance. Although over time, the exchange might have been addressed the issue, but at the time of issuance, required futures contracts were not offered by the exchange. Without external audits, investors must take the exchange's word for it and hope LVTs are indeed financially backed.

RQ 4: What are the possibilities of front-running in LVTs? Predictability of front-running arises during fund rebalancing, deductions of management fees, and futures funding fee exchanges. An attacker would be able to take advantage of brief distortions in supply and demand created by the fund rebalancing trade. The effect could be amplified if all three events happen around the same time or multiple traders participate in the front-running practice at very high volumes. Front-running has also been an issue with LETFs, that is minimized due to constant monitoring by regulatory bodies like the SEC and FINRA.

RQ 5: How well do LVTs track their asserted leverage ratios? While the value of the underlying changes, the value of the fund and the reflected leverage change at different rates. The token price may be at a premium or discount against the actual fund value. All the LVT issuers have a daily rebalancing schedule that reconciles the sum of the tokens' value and that of the fund. However, different issuers implement this process in a different way. This results in the actual effective leverage deviating from the target leverage. Rebalancing mechanism in both fixed and dynamic LVTs has some drawbacks. Fixed leverage is considered more suitable for LVTs as close to 100% of all traditional LETFs use a fixed leverage factor [165]. However, applicable algorithms in LVTs should be revised to decrease the already high deviation of leverage from LETFs.

RQ 6: Are LVT fees in-line with traditional LETFs? High daily fees in LVTs are a constant drag, eroding returns and causing underperformance. It also makes LVTs less attractive investment vehicles. Daily costs in LVTs should be much lower when compared to LETFs, since futures transactions occur internally and there is very little regulatory overhead. Besides that, higher fees are indicative of developing markets with low competition and inefficient cost management.

5.7 Discussion

Similar to leveraged ETFs, the foremost objective of LVTs is making leveraged investment easier while minimizing the hassle of managing such positions and limiting liquidation risks. In the course of our study, we found many shortcomings with LVTs-mostly not transparent, custodied with the issuing exchange, and may be inadequately backed. 99.9% of LVTs are deployed as centralized products, available only within the ecosystem of the exchange itself. 80% of them do not interact with the blockchain at all; due to this, there is no transparency in total supply, transactions, and holders. Some of these issues, along with various financial and security concerns in LVTs, were reviewed.

Since the issuers of 53% of the LVTs do not publish their total supply, it is challenging for most investors to figure out the NAV and trade it at a fair price. Apart from this, 41% of LVTs may also be launched without financial backing as the necessary futures contracts happen to be issued too late or may not have existed when the LVT was initially issued. 97% of total LVTs are susceptible to front-running during generally known events. It enables the adversary to take advantage of rebalancing trades. Moreover, LVTs have higher leverage deviation from the stated ratio due to either inconsistency in management in token funds with fixed leverage or inefficiency in the rebalancing algorithm in LVTs with dynamic leverages. Generally speaking, LVTs have higher management fees compared to LETFs, which eats into the return of the fund in order for it to achieve the expected return. Due to the compounding effect, LVTs tend to underperform over an extended period of time, such as a month or a week, making them unsuitable as a long-term investment.

In fact, all our findings say the same: investors expecting simple leveraged positions that "just work" will be disappointed by leveraged tokens. Due to their peculiar properties, LVTs can only be safely used after considering them carefully, and as such, it is best for sophisticated traders.

Increased scrutiny by regulators and compulsory audits may be other ways to compel LVTs to be adequately backed. The fact that on-chain LVTs exist would allow self-custody and for greater transparency with respect to supply, transactions, and holders. For example, front-running mitigation should be explored using randomized rebalancing or using stealth trading (*e.g.,* iceberg orders). The LVT algorithms should also be modified to decrease deviations from the stated leverage. In summary, incorporating these methods better align LVTs with the natural investor expectations that they create. We implement all of these enhancements in a new hybrid design, LeverEdge, which we describe in detail in Chapter 6.

Chapter 6

LeverEdge: On-Chain Leveraged Tokens

This chapter is based on the work "LeverEdge: On-Chain Leveraged Tokens" supervised by Dr. Jeremy Clark and currently under submission.

6.1 Introduction

In this chapter, we provide a summary of our research focused on the design and proposal of LeverEdge, a fully decentralized leveraged token (LVT) that addresses the deficiencies outlined in Chapter 5. As mentioned, LVTs have been available to investors since 2019. They are often perceived as less risky than other forms of leveraged trading. However, recent research has identified ten critical deficiencies in the current offerings, primarily due to the lack of a standardized implementation framework. We review these shortcomings and explore potential solutions. To this end, we focus on several research questions and propose a new design model, LeverEdge, supported by a prototype deployed on the Ethereum blockchain.

Unlike existing centralized and semi-centralized implementations, LeverEdge is fully decentralized and relies solely on-chain. This approach resolves most deficiencies, but operating fully on-chain is non-trivial because of the inherent complexities and limitations of blockchain technology, such as higher latency, scalability issues, and gas fees, especially when compared to the faster and more straightforward operations of centralized counterparts. As a result, we had to develop a new L1-L2 hybrid model which may be of independent interest. LeverEdge passed security checks and can serve as a potential reference model for future leveraged token deployments on EVM-compatible chains.

6.2 Related Work

The absence of uniform standards in LVT implementation has led to unpredictable technical and financial performance. The research literature [142, 104, 166, 164] has shown that 99.9% of LVTs are centralized, which implies they are only accessible internally within the ecosystem of the exchange itself. 80% of them do not interact with the blockchain, leading to the lack of transparency in transactions, holders, custody, and auditing. Moreover, 53% of the issuers do not disclose the total supply, making challenging for investors to trade LVTs by their fair market price.

Additionally, 41% of LVTs may have been issued without sufficient financial support upon their launch where required future products were launched with delay. About 97% of them are vulnerable to front-running during well-known events. LVTs have higher leverage deviation from the advertised ratio compared to LETFs because of inconsistencies in the management of funds or inefficiencies in rebalancing algorithms. Holding both LETFs and LVTs for extended periods of time impacts the performance, referred to as volatility decay. LVTs also normally have higher management fees than LETFs, which negatively affects the performance of the fund relative to the expected return.

6.3 LVT Return Dynamics

A distinctive feature of the current decentralized LVTs is the variation in daily returns calculations. Let $R_{t_{(n-1)\to n}}$ denote the return of a k-times LVT when the underlying price S fluctuates between times t_{n-1} and t_n ($n \geq 1$). Assuming zero daily interest expense and borrowing rate for simplicity, daily return is typically calculated as Equation (6.1). In contrast, some LVTs employ an alternative power function, as shown in Equation (6.2), leading to non-linear return dynamics.

$$R_{t_{(n-1)\to n}} = 1 + k\Big(\frac{\Delta S_{t_n}}{S_{t_{n-1}}}\Big) \qquad (6.1) \qquad\qquad R_{t_{(n-1)\to n}} = \Big(1 + \frac{\Delta S_{t_n}}{S_{t_{n-1}}}\Big)^k \qquad (6.2)$$

As illustrated in the dotted lines of Figure 6.1, Equation (6.2) amplifies the return of LVT compared to the linear behavior of Equation (6.1), especially when the underlying price decreases and k takes more negative values. This is because the negative

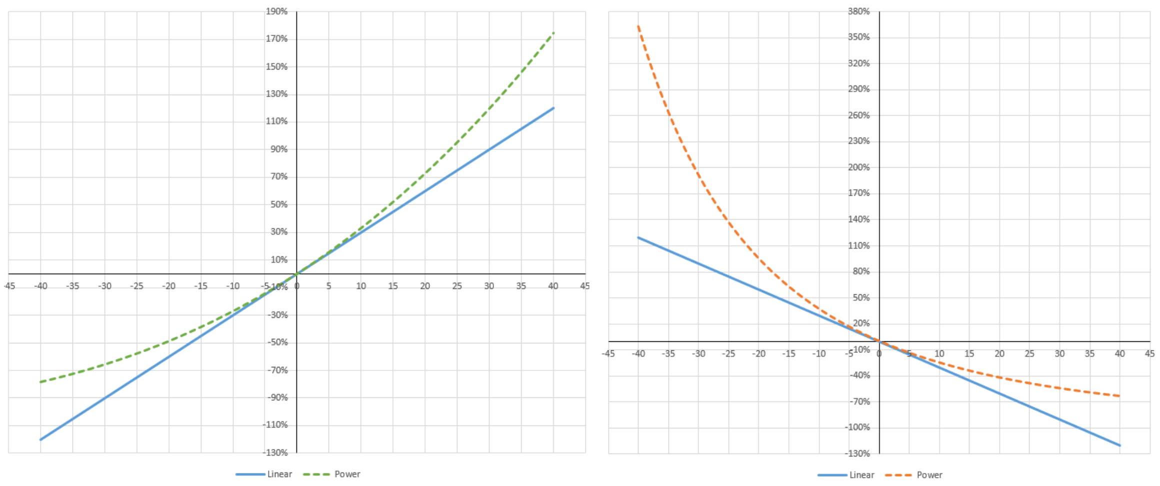


Figure 6.1: The return of a 3x LVT in response to changes in the underlying asset's price is characterized as follows: The return described by Equation (6.1) follows a linear pattern (blue lines), while the return in Equation (6.2) follows a power function (green and red dotted lines). The x-axis represents the percentage return of the underlying asset, and the y-axis denotes the return factor of the LVT. The returns of long and short tokens are illustrated in the left and right diagrams, respectively. In Equation (6.2), the negative k for short tokens amplifies positive returns when the underlying asset's price experiences a sharp decline, leading to the return discrepancy between long and short tokens. This divergence may not be ideal for all users.

exponent enhances the return when the base (*i.e.,* change in the underlying price) is negative. This resulting in a larger positive return for short positions compared to long ones. For small changes in the underlying price (*e.g.,* approximately less than 10% when $k = 3$), both equations yield almost identical results, as the primary difference remains negligible for such small fluctuations. However, as the change in the underlying price becomes more significant, the returns from the two equations diverge sharply. This can result in varying financial outcomes for long and short positions.

#	Deficiency	Proposed Solutions
1	Inability to custody	Implementing tokens on the blockchain enables holders to directly control their tokens. The total token supply is easily accessible, and all token transactions are recorded on an immutable ledger. This allows investors to see how many tokens are held by each address. Auditors can also verify token code and events to ensure proper functionality and financial backing.
2	Transparency in total supply	
3	Transparency in transactions	
4	Transparency in token holders	
5	Interoperability with DeFi	
6	Challenges in auditing	
7	Inadequate financial backing	
8	Possibility of Front-running	Randomized rebalancing or using iceberg orders.
9	Higher tracking error	Optimization of rebalancing algorithms.
10	Higher management fees	Cross-trading or implementation on the layer 2 chains.

Table 6.1: Summary of deficiencies in existing LVTs and proposed solutions.

6.4 Evaluating Decentralized LVTs

As summarized in Table 6.1, seven of ten deficiencies can be overcome by deploying the token on the blockchain. Therefore, decentralization should be considered as the primary approach, since the future centralized tokens is likely to suffer the same drawbacks. The remaining three issues relate to the functionality of the token itself, that can be addressed by optimizing internal algorithms. We therefore focus on on-chain LVTs, where some advances has already been seen. We further examine these tokens and assess the extent to which the identified issues are mitigated.

6.4.1 FLI Tokens

Index Coop launched FLI tokens in 2021 based on the Sets protocol [34]. These ERC-20 tokens consist of 4 smart contracts on the Mainnet and 6 on the Arbitrum blockchain. Tokens on the Mainnet and Arbitrum offer exposure to BTC and ETH up to 2x and 3x, respectively. Unlike most LVT issuers that create leverage through perpetual futures, FLI tokens directly use the lending market to create an over-collateralized leveraged position. While the FLI tokens are a step forward in decentralizing LVTs, it appears that they need optimization in some aspects:

6.4.1.1 *Usability constraints:* FLI tokens may face usability issues due to supporting only ETH and BTC. Inverse and 3x tokens are available only on Arbitrum. Mainnet tokens have only 2x long exposure, and it is not possible to short BTC or ETH. Furthermore, U.S. residents are prohibited from buying these tokens [35]. FLI tokens also has a *Supply Cap* that in the past drove token prices higher [138].

6.4.1.2 *Fee impact:* Resorting to the debt market involves the payment of interest on the value borrowed. FLI also imposes transaction fees through swapping USDC against BTC/ETH. These costs are in addition to the 3.65-5.48% management fee that may reduce the token return.

6.4.1.3 *Rebalancing risk:* Given that rebalancing involves constant price monitoring, FLI taps into off-chain servers for tracking leverage conditions before initiating the rebalancing process. Even though Index Coop uses a network of redundant servers for resilience, it still creates a single point of failure if issues arise with the company or all its servers become unreachable. Moreover, its *Re-centering Speed* needs to be dynamically adjusted by market volatility for an optimal balance between tracking error minimization and rebalancing cost control.

6.4.2 Contango Tokens

Contango protocol is developed by a team incubated by Alpha Venture in 2021 [159]. It utilizes the lending and spot markets to generate leveraged positions through Automated Stacking and Flashswap. Users open new positions by depositing collateral into the Contango protocol. The operating contract posts collateral (ETH) on a lending platform (Aave, Compound), borrows another asset (USDC) against it, swaps it for the original asset (via Uniswap, Paraswap), and repeats the process over multiple loops to increase exposure. The high costs associated with executing these loops are minimized by Flashswap, enabling desired leverage in a single atomic transaction.

The protocol automatically mints an ERC-721 NFT whenever a user opens a new position. The NFT encapsulates details such as the underlying asset, leverage ratio, borrowing rates, and collateral. The designers claim that the implied funding rates for these positions are 2-3 times lower compared to traditional perpetuals. Volatility of the funding rate is also 2-4 times less, making Contango a relatively attractive alternative [51, 5]. Despite these advantages, Contango comes with certain limitations:

6.4.2.1 *NFT transferability:* Secondary market for NTFs may face a lack of liquidity due to the specific features of each position: leverage, trading pair, lending market, and chain. As a result, liquidation of NFTs can be done only through smart contracts, which may leads to low trading volumes, market fragmentation, and high price volatility [153].

6.4.2.2 *Lack of liquidation protection:* To extend liquidation point, LVTs rebalance the fund automatically in case of sharp price swings. Contango, on the other hand, relies on users to actively monitor and manage positions manually to prevent liquidation [52]. This activity directly contradicts one of the key value premise of LVTs: simplifying position management. If a user is inactive or adjusts the position late, it could result in notable losses.

6.4.2.3 *Multisig control:* Contango is not fully immutable; its core team is still capable of upgrading the smart contracts through a 3-of-5 multisignature process. This may introduces centralization risks.

6.4.2.4 *Liquidity risks:* During periods of market volatility or when liquidity is poor, slippage may increase in DeFi spot or lending markets [23], making it harder to execute trades or close positions at favorable prices.

6.4.2.5 *Variable borrowing costs:* Although Contango fees are typically much lower than those in traditional perpetual contracts, relying on variable-rate lending markets can cause borrowing costs to fluctuate. It may potentially resulting in higher-than-expected costs during periods of market volatility.

6.4.3 Cube Tokens

Charm Finance launched Cube tokens in 2021 based on *Parimutuel Pool* concept.[1] Users deposit ETH into a smart contract, CubePool, to mint Cube tokens. These tokens can later be burned to withdraw the equivalent amount of ETH. The Cube token's value dynamically changes due to the underlying prices, pulled from Chainlink price feeds. While attempts at mitigating front-running and minimizing tracking

[1] A betting system where all bets of a particular type are aggregated into a pool, and the payout odds are determined by distributing the pool among all winning bets.

errors through continuous rebalancing are in place, the current design falls short of meeting investor expectations on the following grounds:

6.4.3.1 *Unequal profit distribution:* The return of LVTs is generally derived from the performance of the fund. In Cube, however, there is no fund—there are only losers paying winners, resembling a poker game where players compete for a shared prize or pot.

6.4.3.2 *Inaccurate leverage tracking:* Due to the normalization process, Cube tokens do not accurately track the leverage, making them unsuitable for investors seeking precise returns.

6.4.3.3 *Disincentive for large investments:* Those holders who have large pool ownership earn the least profit because of losses incurred during the normalization, while owners with smaller pool shares experience fewer losses. This dynamic discourages large investments, making them economically unjustified.

6.4.3.4 *Elevated performance risk:* Performance of Cube tokens is relative to each other. For instance, if Bitcoin goes up by 1%, the owner of BTC tokens may still lose value if other tokens in the pool have increased by a greater amount.

6.4.3.5 *Leverage dilution:* When too many users buy the same token, its leverage can decrease, potentially dropping below 1x or even turning negative.

6.4.3.6 *Hindered token adoption:* The safety mechanism of CubePool sets the limit on the Total Value Locked at 100 ETH. When this limit is reached, users cannot buy tokens until others sell [15]. This may limit token usability.

6.4.3.7 *Management risk:* Each pool maintains two super accounts, namely: Governance and Guardian. They have the privilege to execute emergency ETH withdrawals and to pause/unpause pool operations [72]. This kind of centralization may not precisely meet the risk appetite of some investors.

6.4.4 Squeeth Tokens

In 2021, Opyn launched Squeeth token (oSQTH). It offers nearly 2x exposure to ETH's volatility. Users can mint it via Opyn's app or trade it for ETH on DEXs [54]. However, oSQTH may face the following challenges in adoption:

6.4.4.1 *Volatility speculation:* Squeeth's payoff is non-linear and tracks ETH^2. This feature increases sensitivity to changes in the ETH price as shown in Figure 6.1. oSQTH acts more as of a proxy for ETH volatility, and traders holding them speculate on price movement "patterns" rather than directly on the underlying asset's price. This contrasts with the expected linear return of LVTs [137].

6.4.4.2 *Premium costs:* The daily funding fee in most of the perpetual futures is 0.03-0.05% [26], while users pay a 0.1-0.5% premium in Squeeth to benefit from higher risk-to-reward ratio [126]. As shown in Figure 6.1, its power function return works in favor of users in volatile markets. It is worth mentioning that if the price of crypto-asset remains flat for an extended period—something that happens about 50-60% of the time [12, 62]—returns can be significantly eroded due to this heightened funding premium.

6.4.4.3 *Complexity:* To counter token value inflation, Squeeth undergoes regular nor-

Token Name	**Underlying**	**Direction**	**Stated Leverage**	**Leverage Range**	**Layer 2 deployment**			
					Polygon	**Optimism**	**Arbitrum**	**Base**
ETHBEAR1X	ETH	Short	-1x	[-1.1, -0.9]	✓	✓	✓	
BTCBEAR1X	BTC				✓	✓	✓	
MATICBEAR1X	MATIC				✓			
MATICBULL2X		Long	2x	[1.8, 2.2]	✓			
ETHBULL2X	ETH					✓	✓	
BTCBULL2X	BTC					✓	✓	
SOLBULL2X	SOL					✓		
STETHBULL2X	SETH							✓
ETHBULL3X	ETH		3x	[2.7, 3.3]	✓	✓	✓	
STETHBULL3X	SETH							✓
SOLBULL3X	SOL					✓		
BTCBULL4X	BTC		4x	[3.5, 4.5]		✓		
STETHBULL4X	SETH							✓

Table 6.2: List of Toros tokens on Layer 2 blockchains, each featuring varying leverage range, stated leverage, and different underlying assets (source: toros.finance).

malization. Additionally, short positions require users to post collateral to hedge against volatility risk, while this is not required for long positions. Moreover, oSQTH is not available to U.S. persons and certain residents of 15 more countries [55]. These factors add complexity, especially for traders unfamiliar with the specific mechanics of trading Squeeth token.

6.4.5 Toros Tokens

Toros Finance issued Toros tokens in 2023 on L2 chains (see Table 6.2 for the list of supported tokens). These tokens use different smart contracts from dHEDGE, Aave, and 1inch to automate both leverage and asset management. The system, while decentralized in nature, still suffers from some limitations:

6.4.5.1 *Usage constraints:* With the view to minimize transaction costs, Toros tokens are deployed solely on L2 chains, which results in restricted access to greater liquidity available on the Mainnet. Moreover, trading flexibility is limited as usually only ETH and BTC are supported across all L2 chains.

6.4.5.2 *Transparency risks:* The absence of token contract code on the blockchain for specific tokens can result in reduced adoption and create security risks since auditors are unable to confirm the token's functionality.[2]

6.4.5.3 *Upgradability risks:* Toros tokens can be upgraded [59], adding complexity and resulting in increased gas costs because of the proxy pattern used. This ability to upgrade also brings security risks if admin control is ever compromised. Additionally, it raises concerns about possible changes to the fundamental logic of the token.

6.4.5.4 *Manual rebalancing:* The rebalancing process in the dHEDGE protocol (and by extension, Toros) is not automated or on a fixed schedule. dHEDGE offers manual rebalancing, enabling fund managers to modify their positions according to their strategies and market conditions [44]. Nevertheless, this method comes with the potential drawback of experiencing delayed rebalancing in unpredictable markets, which could harm overall performance. Additionally, novice users may not have the appropriate resources to adequately track prices and react promptly, which could heighten the risk of incurring significant losses.

6.4.6 TLX Tokens

TLX tokens were introduced in 2024, offering up to 20x leverage exposure to more than 50 assets. Every token is supported by Synthetix Perp v2 futures. Individuals

[2]arbiscan.io/address/0xf715724abba480d4d45f4cb52bef5ce5e3513ccc#code
arbiscan.io/address/0xad38255febd566809ae387d5be66ecd287947cb9#code

have the option to deposit either USDC or sUSD[3] and choose their desired asset, leverage, and trading direction. The procedure involves creating TLX tokens that can be exchanged for USDC or sUSD at a later time [60, 58]. To gain wider market acceptance, TLX tokens might need some optimizations:

6.4.6.1 *Chain restrictions:* The limitation of TLX tokens to the Optimism network hinders users from accessing it on other L2 chains. Moreover, individuals familiar with the Mainnet or other L2 alternatives might experience decreased liquidity and may not perceive a benefit in transitioning to Optimism, resulting in less usability.

6.4.6.2 *Synthetix v2 limitations:* Presently, TLX tokens are dependent on Synthetix v2, therefore requiring an upgrade to Synthetix v3. Staying connected to the outdated version could leave the protocol vulnerable to risks of updates.

6.4.6.3 *Leverage imbalance:* The different leverage ratios of [1.87x-2.18x] for 2x long and [1.61x-2.53x] for 2x short tokens create an uneven risk exposure. The inconsistency in leverage is widespread among all TLX tokens, making it challenging for traders to keep balanced hedging strategies.

6.4.6.4 *Audit concerns:* The audit report expresses concerns about off-chain rebalancing signals, as well as risks associated with liquidation, immutability, and treasury exposure [187].

Even though there have been notable attempts to decentralize LVTs, a closer look at their functionality shows ongoing shortcomings that need to be addressed.

[3]sUSD is a synthetic stablecoin issued by the Synthetix protocol.

Design parameter	**Cube**	**FLI**	**Contango**	**Squeeth**	**Toros**	**TLX**	**LeverEdge**
Market	Prediction	Lending	Lending	Derivatives	Lending	Derivative	Derivative
Leveraged Product	Parimutuel	Debt	Debt	Power Perp	Debt	Synthetix	Perp
Oracle price feed	N/A	Chainlink	Chainlink	Uniswap GMA	Chainlink	Chainlink	Chainlink
Maximum Long Leverage	3x	3x	3x	2x	3x	20x	5x
Maximum Short Leverage	-3x	-1x	-1x	-1x	-1x	-20x	-5x
Leverage Deviation	Very high	Moderate	Moderate	Very high	Moderate	High	Low
Liqudation protection	No	Yes	No	No	Yes	Yes	Yes
Front-running protection	No	No	No	No	No	No	Some
Supported Underlying	Limited	2	Platform	1	2	Platform	Platform
Return function	Non-Linear	Linear	Linear	Non-Linear	Linear	Non-Linear	Linear
Token type	ERC-20	ERC-20	ERC-721	ERC-20	ERC-20	ERC-20	ERC-20
Slippage	N/A	1-2%	1-2%	1-2%	2-3%	N/A	1-2%
Non-Upgradable Contract	Yes	Yes	Yes	Yes	No	Yes	Yes
Verfied Contract Code	Yes	Yes	Yes	Yes	No	Yes	Yes
Regional restriction	No	Yes	No	Yes	No	No	No
Fees	3.5-5%	3.65-5.48%	2-3%	3-5%	2-3%	2.5-3.5%	2-3%
Daily funding rate	N/A	0.01-0.02%	0.01-0.02%	0.1-0.5%	0.01-0.02%	0.01-0.03%	0.01-0.03%

Table 6.3: Comparison of on-chain LVTs. Deficiencies in the parameters highlighted in green have been fully rectified, and partially for parameters in yellow.

In this regard, we propose and evaluate a new L1-L2 hybrid model, LeverEdge, and compare how well it performs against others. As summarized in Table 6.3, LeverEdge addresses most of the existing problems and also brings additions like front-running alleviation and lowered fees.

6.5 LeverEdge: New Proposal

6.5.1 Leveraged Product Selection

Our research concludes three possible choices for leveraged products in creating a fund for LVTs: (i) futures contracts, (ii) leveraged positions based on debt, and (iii) synthetic assets. Although the debt market may have lower and more consistent funding rates, perpetual contracts offer unique benefits such as easy access to both long and short positions and efficient rebalancing.

The selection of leveraged products involves various important factors: Firstly, LVTs are designed for experienced traders who have a thorough understanding of market dynamics. They generally focus on immediate price changes rather than long-term holding strategies. Day traders face one of two outcomes: either the price trend matches their initial analysis, enabling them to make a profit within the same trading day, or the analysis proves to be incorrect, leading to a swift exit to reduce losses. They do not stay in positions for an extended amount of time to take advantage of the reduced costs in debt-based LVTs. Additionally, they are aware that if the market goes against them, the probability of a positive turnaround on the same trading day is slim [81]. As a result, LVTs are specifically made for skilled traders, with inexperienced users being more prone to losses, if they hold LVTs for a long time.

The second important factor riles on the fact that, the debt market is not efficient for establishing short positions. Lending protocols usually demand that stablecoins be over-collateralized in order to borrow crypto-assets at a 1:1 ratio. Creating short tokens is less lucrative and more difficult than creating long positions (see Figure 6.2 and compare the processes of creating long and short positions). This assertion is additionally supported by Table 6.3, demonstrating that debt-based LVTs solely provide -1x short tokens, not leverages like -2x, -3x or -5x. Having this restriction limits the capacity to form short positions that are similar to long ones.[4] Given these constraints in the lending market, our design is (i) structured around perpetual futures rather than the debt market, (ii) more suitable for short-term traders.

[4]In other words, debt-based short LVTs are restricted to an inverse exposure of 1:1, as opposed to higher leverage x:1 ratios such as 3:1 or 5:1.

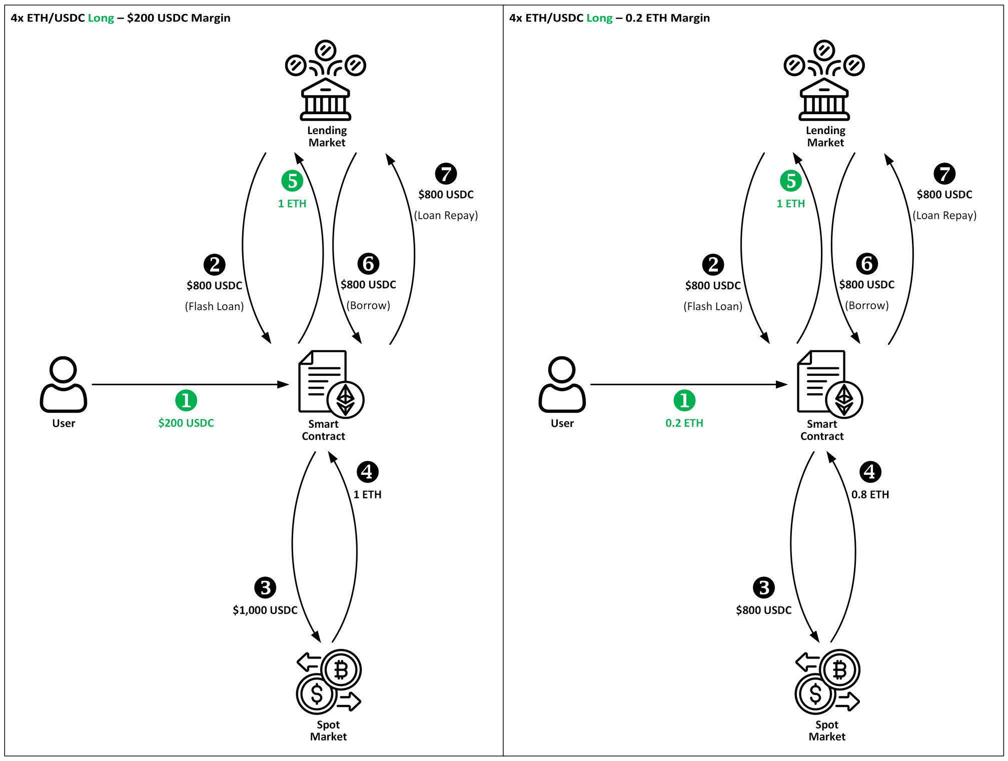
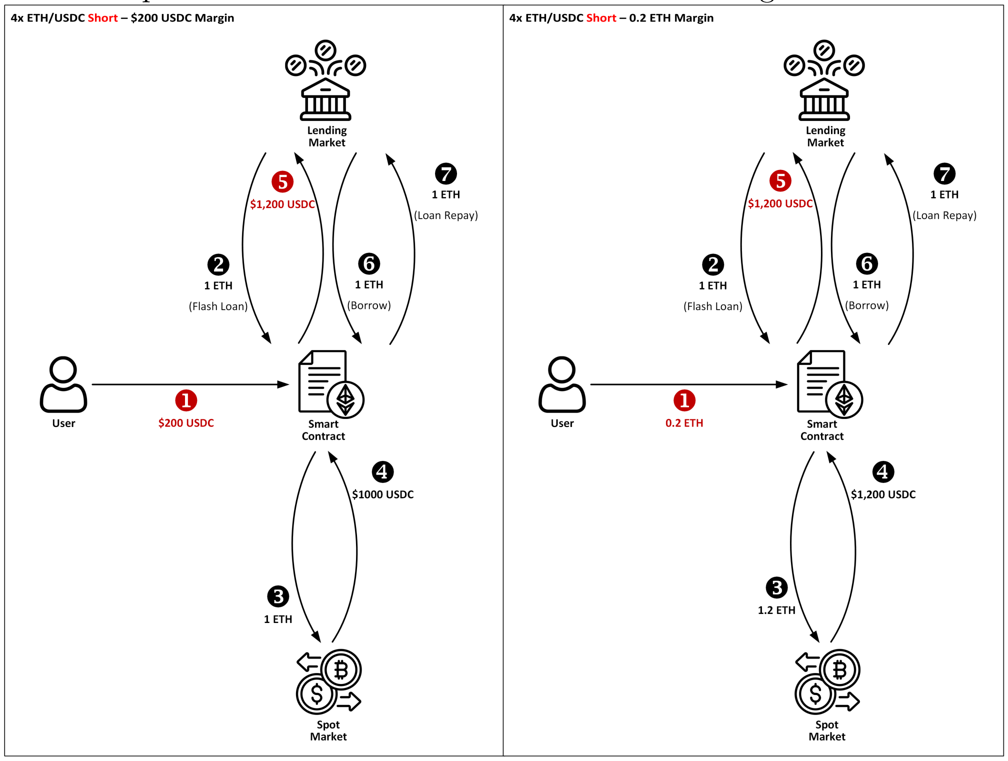


Figure 6.2: Diagrams illustrating the creation of a 4x long (top) and short (bottom) position through automated looping, where users either deposit stablecoins (left) or post crypto as margin (right). Short positions differ from long positions in that they require full exposure to the notional value of the asset being shorted.



6.5.2 Resolving LVT Shortcomings

Mentioned risks affect the financial efficiency and reliability of LVTs, leading to less acceptance by investors. Implementing LVT on the blockchain can address these issues by bringing transparency to the total supply, transactions, holders, custody, auditing and financial backing. On-chain implementation should therefore be used by LVTs to mitigate the associated risks in centrally issued tokens.

Another discussed issue is the effect of front-running which may place LVT in a position where it needs to execute more trades than otherwise required for simple rebalancing (see Section 5.5 for more details). Front-running has always been a challenge in traditional markets, but most issues have been controlled through regulatory oversight [151]. Lacking that regulation in crypto, the risk can be mitigated by: (i) camouflaging trading intent through enhanced algorithms that randomize both the timing and size of rebalancing trades, making it harder to predict when and how much of the underlying asset will be bought or sold; (ii) cross-trading between long and short pairs, whereby issuers internally swap matching positions to minimize the need for open market transactions, as token issuers can act as the levered product and match orders for different LVTs in-house; and (iii) collaborating with liquidity providers and market makers (*e.g.,* permissioned pools by Aave Arc and Fireblocks [68]), allowing issuers to discreetly source liquidity and execute large orders without revealing their trading intentions.

Although we proposed solutions for front-running, there would be still a possibility of front-running by miners who may intentionally include the rebalancing transaction in upcoming blocks and delay it to prioritize their own. Therefore, the risk of front-running can be alleviated and not completely eliminated.

A further issue with LVTs is tracking error (leverage deviation). Some of the ways to reduce tracking error include: (i) increasing rebalancing frequency, possibly every n blocks instead of the default daily rebalancing, to more precisely maintain the target leverage ratio. This may involve using L2 chains that have lower transaction fees; (ii) performing interim rebalancing at threshold crossings of the underlying asset price (*e.g.,* $\pm 10\%$), which would mitigate the impact of sudden price changes by rebalancing leverage more than once a day; and (iii) having a range allowing variation in leverage (*e.g.,* [1.95x, 2.05x] for a 2x long token), permitting small deviations without rebalancing. Thus, for both long and short tokens, this range must be set carefully to avoid major tracking errors. This approach should also be fully explained to investors to manage expectations properly, as it represents a workaround rather than a definitive solution.

Another notable difference in LVTs is high management fees compared to LETFs that can be minimized by: (i) using advanced algorithms (iceberg or reverse orders), which enable the optimization of the timing and size of trades to minimize slippage and bid-ask spread costs; (ii) cross-trading between long and short funds, since issuers operating both can internalize trades by directly matching orders of funds sharing the same underlying assets. This internal matching at better prices further reduces slippage and transaction costs; (iii) using L2 chains [82] in combination with MultiCall transactions [98], which further lower the operating cost.

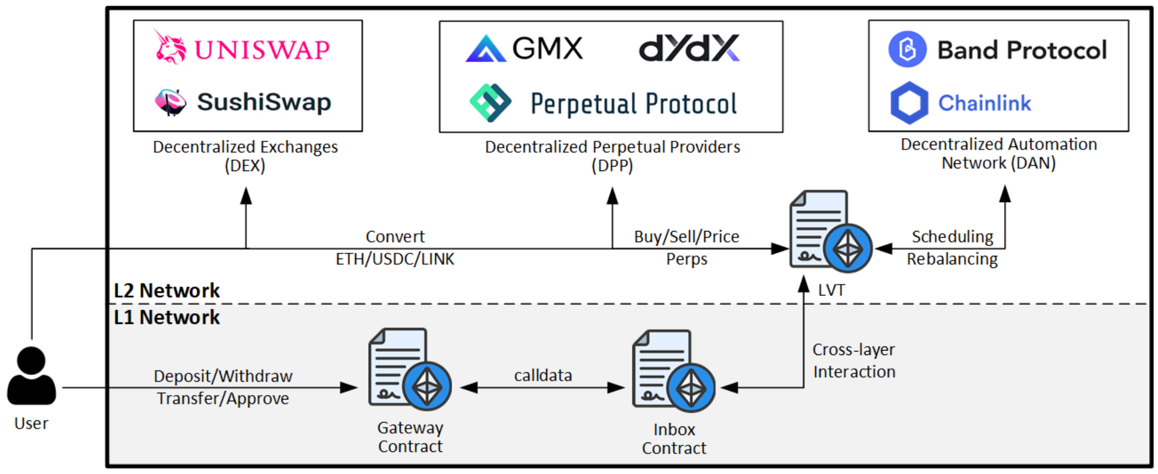


Figure 6.3: The core components of LeverEdge consist of the DEX, responsible for handling necessary crypto exchanges. Futures positions are managed via DPP, while the rebalancing process is triggered by DAN. Users can interact directly with the token on L2 or rely on the gateways contract, which streamlines cross-layer contract interactions.

6.5.3 Design Model

LeverEdge follows a hybrid L1-L2 architecture, with smart contracts deployed on both L1 and L2 chains. The L2 contract acts as the primary code, managing the core functionality of the token. Users can directly interact with the L2 contract if their funds are already on L2, or they can access the token via the gateway contract on L1, which accepts funds and facilitates communication with L2 (see cross-layer interaction in Figure 6.3). The gateway contract serves as the main entry point for L1 users, enabling interaction with L2 through Bridge or Inbox contracts. Thus, user-facing operations in LeverEdge can occur on either L1 or L2, depending on where users hold their funds, while trading activities remain on L2 to ensure faster, more cost-efficient execution. This hybrid architecture optimizes scalability and security by allowing the L1 contract to handle secure finality, while L2 focuses on high-speed execution and cost reduction.

LeverEdge design allows issuers to choose DeFi partners according to factors like service fees, security, and performance. As shown in Figure 6.3, it incorporates three main groups of DeFi providers: (i) DEX for trading crypto assets on-chain, (ii) DPP for establishing leveraged positions with perpetual contracts, and (iii) DAN for monitoring conditions and executing smart contract functions upon meeting predetermined criteria, such as daily or interim rebalancing.

Example 26. A LVT issuer may decide to use Uniswap instead of Sushiswap for making a 2x long ETH token (ETH2L), while another might select GMX on Arbitrum rather than dYdX on StarkEx for handling perpetual futures. dYdX appeals to high-volume traders with its gasless trading, whereas GMX attracts retail traders with its higher leverage options. Ultimately, both tokens deliver twice the yield of ETH, but through different DeFi services: the first approach reduces token operating costs, while the second prioritizes user adoption.

The process of incorporating DeFi services into LeverEdge is made more efficient with standardized interfaces. LVT issuers can choose their preferred DeFi partners without impacting the token's primary functions. Essentially, LeverEdge creates interfaces that encapsulate DeFi services, enabling the token to function smoothly with various providers, regardless of their internal procedures.

Decentralized Exchange (DEX)

Most DeFi systems use USDC as the primary transaction currency, and LeverEdge operates seamlessly with it. It accepts USDC by default while also supporting ETH-to-USDC conversion. If users send ETH to the contract, it is automatically converted to USDC. Similarly, when users sell tokens, they can choose to receive either USDC or ETH. Interacting with LeverEdge using USDC minimizes conversion fees. In line with a fundamental principle of blockchain, the requester is responsible for paying all associated fees. This means that users initiating buy or sell transactions are required to cover all relevant costs, including any fees for converting USDC to ETH.

In addition to USDC, LINK tokens are required to automate certain tasks. These tokens are consumed by the Chainlink network to trigger the rebalancing process. During each rebalancing event, a portion of the perpetual contracts are sold and converted from USDC to LINK, which is then sent to Chainlink as an incentive to maintain the rebalancing mechanism.

Interaction with decentralized exchanges is essential for LeverEdge to handle both internal and external currency conversions. By default, Uniswap is used due to its lower exchange fees and higher liquidity. However, the modular design of LeverEdge allows for integration with other DEX providers, such as Sushiswap, Paraswap and Kyberswap.

Decentralized Perpetual Provider (DPP)

Decentralized perpetuals offer a decentralized alternative to dominant centralized platforms, allowing LeverEdge to create long or short exposure to various crypto assets. When it comes to decentralized perpetuals that can only be traded on L1, the options are relatively limited due to scalability constraints and high gas fees. The more viable solution is to use DPPs on L2 networks such as Polygon, Optimism, Arbitrum, and StarkNet. However, there is a need to make communication between L1 and L2 seamless to improve the user experience. In this regard, we implement a gateway contract on L1, which facilitates user interaction with L2.

Users do not need to bridge their funds manually, as the gateway contract handles this process for them. The gateway contract works in conjunction with the bridge contract of the corresponding L2 chain, enabling LeverEdge to securely transfer user assets between the two layers while maintaining decentralization and security. The gateway contract also communicates with the main contract on L2 through a messaging system. The primary mechanism for transferring data between L1 and L2 involves sending messages via LeverEdge smart contracts deployed on both layers.

dYdX and GMX are two popular DPP platforms on L2 that can be integrated with LeverEdge. On dYdX, USDC is the only accepted collateral, and all deposits must be made in USDC. dYdX has migrated its perpetual v3 trading entirely to the StarkEx L2 chain. StarkEx, utilizes zk-rollups to enhance transaction throughput and minimize gas fees. It uses StarkEx to settle perpetual contract trades on its platform, offering a more efficient and scalable trading experience while maintaining security by relying on L1 for data availability. dYdX has implemented the *Currency Converter* and *StarkEx Bridge* contracts on L1, which LeverEdge uses to communicate with the main dYdX contract on L2.

Example 27. Alice deposits $100 USDC from Ethereum L1 into the ETH2L token, which is configured to use GMX on Arbitrum. Instead of sending USDC directly to the main ETH2L contract on the Arbitrum network, Alice sends her USDC to the ETH2L gateway contract on L1. The gateway contract then interacts with the Arbitrum Bridge contract deployed on L1 to transfer Alice's $100 USDC to the main ETH2L contract on Arbitrum. Additionally, the gateway contract appends metadata to the transaction, linking the deposited funds on Arbitrum with Alice's L1 address.

LeverEdge allows seamless selection of providers due to the modular design of the DPP contract. This decouples the internal mechanics of each provider. The DPP contract implements the `buyPerp()` and `sellPerp()` functions to interact with DPP. These functions are called by the `subscribe()` and `redeem()` methods for buying and selling futures. Since the internal buy or sell logic is abstracted away from these methods, token issuers can select the appropriate DPP when initially creating tokens.

LeverEdge supports cross-layer contract interaction, where it sends a transaction from L1 that not only bridges the funds but also triggers a function call on the main contract in Arbitrum. This is achieved using the *Inbox* contract deployed on L1. The Inbox contract allows the passing of *calldata*, which specifies the contract to call and the function to execute once the funds are deposited on L2. LeverEdge uses this mechanism to communicate with the main contract.

Example 28. In the Example 27, the L1 Bridge contract of Arbitrum locks Alice's USDC on L1 and generates a message sent to the L2 chain. This message is bundled into a batch of transactions that are eventually processed by L2, crediting the main ETH2L contract on Arbitrum. Conversely, when ETH2L sells GMX positions and withdraws funds from Arbitrum, the L2 chain sends a message to the L1 Bridge contract, which releases Alice's USDC after a challenge period (*i.e.,* one week in the optimistic rollup used in Arbitrum).

Decentralized Automation Network (DAN):

Smart contracts cannot autonomously execute their functions based on predefined conditions or time. They require external triggers, either manual or via centralized mechanisms, both of which have significant limitations. Manual triggers are impractical, and centralized solutions are vulnerable to failure or exploitation.

Decentralized solutions like Chainlink address this issue by enabling smart contracts to automate key functions. Chainlink provides a Distributed Automation Network (DAN) that continuously monitors predefined conditions, which may be based on time, events, computations, or a combination of these factors. Nodes within the Chainlink network initiate on-chain transactions once the conditions are met, triggering the smart contract's functions. In LeverEdge, we use Chainlink to automate both regular and interim rebalancing processes.

Modular Architecture

LeverEdge is designed in a modular fashion by defining specific interfaces. These interfaces outline the required function signatures for inherited contracts without providing their implementation. By using interfaces, LeverEdge enables standardized interaction, fostering a modular and extensible contract architecture. Developers can create custom modules and integrate different providers with the token, as long as they adhere to the same interface specifications. This flexibility ensures seamless integration while maintaining compatibility across various components.

Example 29. The `swapEth()` method is designed to convert ETH to USDC, with the underlying DEX provider being either Uniswap or Sushiswap. From a functional perspective, the method's sole purpose is to perform the conversion, regardless of the internal mechanism of the chosen DEX. If issuers are not fully satisfied with the service quality, they can select competing providers. Methods such as `setDexProvider()`, `setDppProvider()`, and `setDanProvider()` facilitate this flexibility, allowing issuers to specify providers when initially deploying the smart contract.

If changing the provider does not impact the token's functionality, it can be performed even after token deployment. In Example 29, the switch from Uniswap to Sushiswap can occur live, without affecting the fund's value or futures positions. This transition ensures that Sushiswap handles the conversion instead of Uniswap, while maintaining the same currency conversion functionality. The issuer may plan this change to reduce costs or optimize the conversion process. Similarly, issuers can change the DAN live, as it triggers the rebalancing process at specific intervals.

Users Experience

The initial token price is usually set at \$1, with a starting supply of zero tokens. As users deposit funds, the token supply increases proportionally. When users sell tokens, those tokens are removed from circulation, maintaining a dynamic supply that adjusts based on user activity. End-user interaction with the token begins by sending USDC or ETH to the `subscribe()` method. If users bypass this method and send funds directly to the smart contract address, they must later use the `claim()` method to receive the equivalent value in tokens. This mechanism addresses cases where funds are mistakenly sent to the smart contract without invoking the `subscribe()` method.

Upon receiving funds, the smart contract purchases futures aligned with the token's leverage. It calls the `buyPerp()` function from the DPP module to interact with the futures provider and increase contract positions. Atomic transactions ensure the consistency of the transaction. Whether it fails or succeeds, the transaction cost is incurred by the user initiating the new position.

These newly acquired perpetuals directly impact the overall fund value, which can be validated by calling the `getBasketValue()` method. This method is usually called by auditors to ensure proper financial backing. Additionally, the number of contracts on the DPP can be directly verified, as each buy and sell transaction triggers a `BuyPerp` or `SellPerp` event logged on the blockchain.

Example 30. Bob sends \$10K USDC to the `subscribe()` method of a 3x leverage

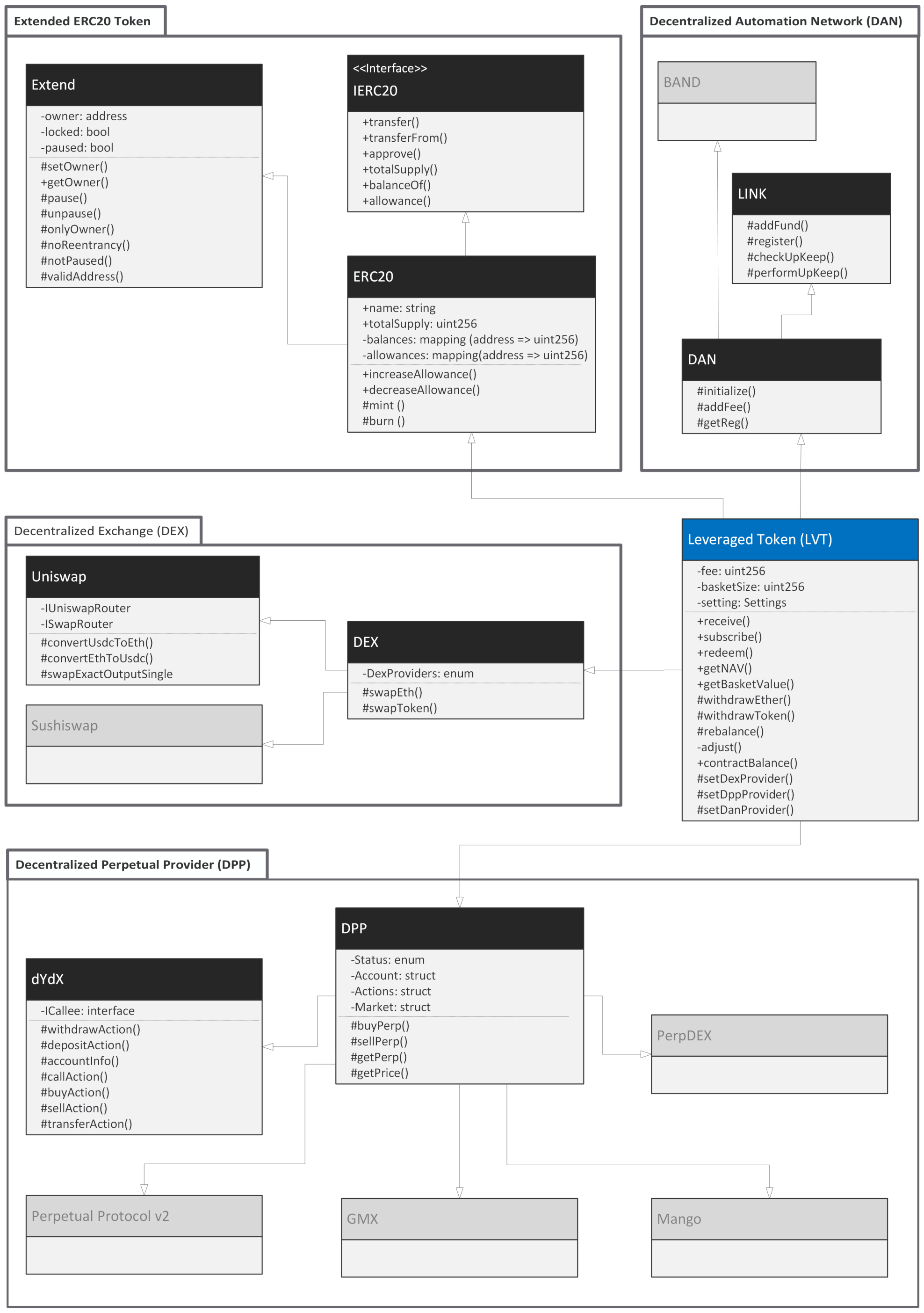


Figure 6.4: The design model of LeverEdge, our proposed decentralized leveraged token, is presented as a UML class diagram. The LVT contract inherits from the ERC-20, DPP, DEX, and DAN contracts. Contracts marked in gray are not implemented and are included only to illustrate possible alternatives. For example, a developer might prefer to use GMX instead of dYdX in a custom implementation.

token when the Bitcoin perpetual price is \$30K. As a result, \$30K worth of futures contracts are purchased and added to the fund value. The `subscribe()` method also invokes the `mint()` function of the smart contract to update Bob's token balance (refer to the ERC-20 interface in Figure 6.4). Bob can later check his token balance by calling the `balanceOf()` method of ERC-20 interface.

Net Asset Value

The Net Asset Value (NAV) is typically used by users engaging in bulk transactions, as it reflects the token's true value. Throughout the trading day, the token price may fluctuate due to market volatility, resulting in a premium or discount. The NAV price can be retrieved using the `getNAV()` method, which calculates the actual price based on the underlying asset's value and the token's leverage, after deducting fees. The returned value, in USDC, serves as a reference for determining the token's fair value.

It is worth noting that fees are automatically deducted daily, causing value of LeverEdge to gradually decrease relative to the futures it holds. This mechanism ensures that fees are paid from the token's total assets, reducing the NAV proportionally. Consequently, daily fee deductions impact the token's price over time, making it more suitable for short-term investments. In a prolonged sideways market, fees may reduce returns in buy-and-hold scenarios.

Example 31. To sell tokens in the Example 30, Bob can call the `getNAV()` method to get the NAV price he will receive. Then, he may invoke the `redeem()` method and specify the number of tokens he wishes to sell. The `redeem()` method calculates the equivalent USDC value and transfers it to Bob after selling the corresponding futures. Bob also incurs all necessary transaction fees when initiating the sale.

6.5.4 Price Dynamics

LeverEdge amplifies exposure to price movements of the underlying (the price of perpetual futures), both upward and downward. Let P_{t_n} represent the price of the k-times leveraged token at time t_n. The token's price evolves in response to changes in the underlying asset S_{t_n} between time t_{n-1} and t_n. In continuous time, this price change can be described by a differential equation that incorporates: (i) token leverage, (ii) underlying price fluctuations, (iii) funding and management fees. The differential form of P_{t_n} is given by the following stochastic differential equation (SDE):

$$\frac{dP(t)}{P(t)} = k\frac{dS(t)}{S(t)} - \big(F(t) + M(t)\big)dt$$

Where the first term represents the leveraged return from the underlying perpetual futures (tracking asset), amplifying the token return by the leverage factor k. $F(t)$ is the continuous funding fee rate, typically expressed as a daily rate per second. It reflects the cost (or sometimes gain) of holding the perpetual futures during the given period. $M(t)$ is the continuous management fee rate, expressed annually but calculated per second. By solving the SDE over the interval $[0, t]$, we obtain the continuous-time price formula:

$$P(t) = P(0) \cdot \exp\left(\int_0^t k \cdot r\big(S(\tau)\big)\, d\tau - \int_0^t \big(F(\tau) + M(\tau)\big)\, d\tau\right) \tag{6.3}$$

Where P(t) is the value of the token (or NAV) at time t. $P(0)$ is the initial token price at time $t = 0$. The first term reflects the cumulative leveraged return of the underlying over the time period $[0, t]$. $r\big(S(\tau)\big)$ is the return on the underlying asset S. The second term accounts for the accumulated funding and management fees over the same period. In summary, equation 6.3 expresses LeverEdge's price as a function of the underlying futures return, adjusted for continuously accruing funding and management fees. This formulation captures the real-time dynamics of LeverEdge's price evolution in response to market movements and the associated costs.

Since funding and management fees are usually expressed as daily or yearly rates, they can be discretized into small intervals (such as seconds or minutes) to approximate the continuous accrual process. This method ensures that fees are seamlessly integrated over time, preventing users from avoiding fees by timing their buy or sell actions just before daily fee deductions. In this regard, formula (6.4) is used to calculate LeverEdge's price in discrete time between t_{n-1} and t_n for intraday token buy or sell transactions:

$$P_{t_n} = P_{t_{n-1}} \left(1 + k\frac{\Delta S_{t_n}}{S_{t_{n-1}}}\right) \times \big(1 - (F_{t_n} + M_{t_n})\big), \quad \forall\, t \geq 1 \tag{6.4}$$

M_{t_n} and F_{t_n} represent the management and funding fees, respectively. P_{t_n} is calculated based on the percentage change in the underlying asset S_{t_n}. The deduction is incorporated into the token's price, meaning holders indirectly pay the fee as the token value is adjusted daily. There is no need for holders to manually send any funds; instead, the token's value continuously reflects the deduction of the fee.

Example 32. Alice deposits USDC into a ETH5L token at 7 a.m. when the token price is \$1 per token. She receives the equivalent ERC-20 tokens in return. By 11 p.m., the price of ETH has increased by 10%. As a result, the ETH5L token rises by $10\% \times 5 = 50\%$, reaching \$1.50 per token. This amplifies Alice's initial investment by a factor of $k = 5$.

$$P_{11pm} = P_{7am}\left(1 + 5\frac{\Delta S_{11pm}}{S_{7am}}\right) = \$1(1 + 5 \times 10\%) = \$1.5$$

Example 33. In the previous example, Alice decides to take profits by selling all her tokens at 11 p.m., just before the fund rebalancing. However, the smart contract applies a slight reduction to the token price, reflecting the 2.5% annualized fee and the 0.03% daily futures funding fee. Consequently, Alice's tokens are valued slightly lower to account for the operational costs of the smart contract. This approach ensures that fees are continuously incorporated, preventing Alice from avoiding these deductions by selling her tokens just before the fees are applied at midnight.

$$\begin{aligned}
P_{11pm} &= P_{7am}\left(1+5\frac{\Delta S_{11pm}}{S_{7am}}\right)\times(1-0.03\%)\times\left(1-\frac{2.5\%}{365}\right)\\
&= \$1(1+5\times 10\%)\times(1-0.0003)\times(1-0.00006849)\\
&= \$1.5\times 0.9997\times 0.9999 \approx \$1.499448
\end{aligned}$$

When users close their positions, the smart contract sells the corresponding futures positions to match the fund's notional value with the value represented by the tokens. Since users initiate the transaction, they incur the transaction costs, including selling perpetual contracts, withdrawing USDC from the futures provider, deducting fees, and transferring the USDC to their wallet. It is important to note that the transaction is atomic, ensuring the seamless execution and consistency of all operations.

6.5.5 Fund Rebalancing

LVTs aim to deliver a multiple of the daily performance of an underlying asset (*e.g.*, 2x or 3x). Daily rebalancing ensures that the LVT's leverage ratio is consistent despite market fluctuations. This is critical as the notional value of both the LVT and fund can change significantly throughout the day, as the price of the underlying changes.

Let B_{t_n} represent the numbers of perpetual contracts in a k-times fund. Equation 6.5 suggests that with the change in the price of the underlying (S_{t_n}) and consequently the price of the leveraged product (V_{t_n}), the notional value of the leveraged fund (L_{t_n}) adjusts, causing the realized leverage ratio of LVTs ($\tilde{k_{t_n}}$) to deviate from the stated leverage. Mathematically, $\mathbb{E}[\tilde{k_{t_n}}]$ represents the expected change in $\tilde{k_{t_n}}$ in response to the underlying price movement, expressed as:

$$\mathbb{E}[\tilde{k_{t_n}}] = \frac{kV_{t_n}B_{t_n}(1 + R_{t_{(n-1)\to n}})}{kN_{t_n}P_{t_n}} = \frac{V_{t_n}B_{t_n}(1 + R_{t_{(n-1)\to n}})}{N_{t_n}P_{t_{n-1}}(1 + kR_{t_{(n-1)\to n}})} \tag{6.5}$$

Where $R_{t_{(n-1)}} = \frac{\Delta S_{t_n}}{S_{t_{n-1}}}$ denotes the underlying return between time t_{n-1} and t_n. The fund notional value L_{t_n} is given by $kN_{t_n}P_{t_n}$, where P_{t_n} represents the token price, as defined in Equation 6.4. It is known that the fixed number of tokens N_{t_n} in the denominator, scaled by the factor k, fluctuates based on price movements, resulting in deviations from the intended k leverage ratio.

Example 34. Table 6.4 illustrates the rebalancing process of ETH5L (5x long Ethereum token) over a 5-day trading period. During the first three days, rebalancing occurred daily. However, on the fourth and fifth days, interim rebalancing was triggered due to price fluctuations exceeding 10% within a single trading day, reflecting heightened market volatility.

Example 35. If we take the second trading day of Example 34, the price of Ether contracts increased from \$3,000 to \$3,300, causing the token's value to grow by a factor of 5, resulting in a 50% increase. Consequently, the token price rose from \$1 to \$1.50. However, when comparing the fund's value to the token's value, we observe a deviation of $\frac{3.6\text{x}}{5\text{x}} - 1 = 28\%$ in the leverage. This occurs because the fund holds one contract (row 3), bringing the notional value of the fund to $1 \times \$3,300 = \$3,300$ (row 4). With 600 tokens in circulation (row 14), each 5x token at a price of \$1.50, the

	ETH5L	ETH5L - Before Rebalancing							
		Day 1	Day 2	Day 3	Day 4		Day 5		
1	**Futures Price (S)**	$3,000	$3,300	$3,135	$2,821	$2,961	$2,664	$2,931	$3,000
2	**Futures Price Change**	0.00%	10.00%	-5.00%	-10.02%	4.96%	-10.03%	10.02%	2.35%
3	**Fund Basket Size**	1.0000	1.0000	1.3636	1.1663	0.6171	0.7102	0.4000	0.5458
4	**Fund Notional Value (V)**	$3,000	$3,300	$4,275	$3,290	$1,827	$1,892	$1,173	$1,637
5	**Funding Fee (3*0.01%)**	$0.90	-$0.99	$1.28	$0.99	-$0.55	$0.57	-$0.35	-$0.49
6	**Volatility Factor**	1.0	1.0	1.0	2.5	1.0	2.5	3.5	1.0
7	**Mgmt. Fee**	-$52.22	-$57.44	-$54.57	-$122.77	-$51.54	-$115.93	-$178.57	-$52.22
8	**Fund Net Value**	$2,950	$3,243	$4,223	$3,171	$1,776	$1,779	$997	$1,586
9	**Token Supply Change**	0	0	50	-30	-20	10	0	-10
10	**Circulating Tokens**	600	600	650	620	600	610	610	600
11	**Token Price**	$1.00	$1.50	$1.13	$0.56	$0.70	$0.35	$0.52	$0.59
12	**Token Value (NAV)**	$0.98	$1.08	$1.30	$1.02	$0.59	$0.58	$0.33	$0.53
13	**Token Price Change**	0.00%	50.00%	-25.00%	-50.08%	24.81%	-50.15%	50.11%	11.77%
14	**Tokens Value**	$600	$900	$731	$348	$421	$213	$320	$352
15	**Tokens Notional Value**	$3,000	$4,500	$3,656	$1,741	$2,103	$1,066	$1,600	$1,759
16	**Target Leverage**	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
17	**Realized Leverage**	4.92	3.60	5.77	9.11	4.22	8.35	3.12	4.51
18	**Tracking Error (%)**	-2%	-28%	15%	82%	-16%	67%	-38%	-10%
		ETH5L - After Rebalancing							
19	**Fee Basket Size**	-0.0171	-0.0177	-0.0170	-0.0432	-0.0176	-0.0433	-0.0610	-0.0176
20	**Fund Basket Size**	0.9829	0.9823	1.3466	1.1231	0.5996	0.6669	0.3390	0.5282
21	**Fund Basket Delta**	0.0171	0.3813	-0.1804	-0.5060	0.1106	-0.2668	0.2068	0.0580
22	**Fund Basket Size**	1.0000	1.3636	1.1663	0.6171	0.7102	0.4000	0.5458	0.5863
23	**Fund Notional Value (V)**	$3,000	$4,500	$3,656	$1,741	$2,103	$1,066	$1,600	$1,759
24	**Rebalanced Leverage**	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
25	**Rebalancing Type**	Regular	Regular	Regular	Interim	Regular	Interim	Interim	Regular
26	**Rebalancing Schedule**	00:00 UTC	00:00 UTC	00:00 UTC	17:43 UTC	00:00 UTC	06:12 UTC	22:32 UTC	00:00 UTC

Table 6.4: The rebalancing details of the ETH5L (Ethereum 5x long token) over a 5-day trading period. Interim adjustments are triggered by market fluctuations that exceed the 10% threshold in a trading day.

total token value amounts to $600 \times 5 \times \$1.5 = \$4,500$ (row 15). In other words, the underlying price change reduces the token's leverage from 5x to $\frac{\$3{,}300}{600\times\$1.5} = 3.6$x (row 17), causing the fund to no longer accurately reflect the token's value. To resolve this issue, the rebalancing process increases the number of contracts from 1 to 1.3636 (row 22), ensuring that the fund's total value of $1.3636 \times \$3,300 \approx \$4,500$ matches the total value of tokens.

Example 36. The capital required to increase the number of contracts in Example 35 can either come from debt or the token treasury, funded by the proceeds from the sale of previous contracts. If debt is used, the interest costs are deducted during the daily rebalancing process. Alternatively, the token price can be adjusted to maintain balance. Instead of increasing the fund's value to $4,500, the token price would be reduced to \$1.10 per token (row 12).[5] In this scenario, the total value of the tokens, $600 \times \$1.1 \times 5 = \$3,300$, matches the value of the fund, $1 \times \$3,300 = \$3,300$, preventing inflation or deflation of the token price. It is important to note that users always buy and sell their tokens at the NAV price, which reflects the actual value of the fund. Both methods have distinct advantages and disadvantages, depending on the issuer's business model. From the users' perspective, sudden changes in token price, are generally not well-received. Adjusting the fund's value, rather than altering the token price, is a more preferred approach.

Rebalancing in LeverEdge managed by the `rebalance()` method, which logs an event after each fund position adjustment, including the number of futures contracts before and after rebalancing, and the delta positions. Fee deductions also take place during rebalancing to cover the costs of (i) adjusting fund positions through perpetual providers, (ii) maintaining rebalancing triggers on the automation network, and (iii) handling cryptocurrency conversions through decentralized exchanges like Uniswap.

6.5.6 Resolved Deficiencies

Taking advantage of all on-chain capabilities enables us to create a completely decentralized system, that is a new solution for deploying LVTs. To our knowledge, LeverEdge is the first design for LVTs that utilizes decentralized perpetual contracts instead of debt or synthetic positions. It tackles the shortcomings of existing central-

[5]For simplicity, we have omitted fees in example calculations. However, it is included in the table to ensure greater accuracy. As a result, there may be slight differences between the example and the values presented in the table.

Algorithm 1 LVT Rebalancing Algorithm with 30-Minute Random Ordering

```
**Inputs:**
$k$: Target leverage ratio (e.g., 2x, 3x, etc.)
$P_t$: Fund value at time $t$
$R_t$: Underlying asset return at time $t$
$F$: Rebalancing frequency (e.g., hourly, daily, etc.)
$\sigma_t$: Volatility at time $t$ (optional)
$H_t$: Hedging component at time $t$
$\Delta t$: Randomized 30-minute time adjustment for rebalancing
Initialize Fund value at $t = 0$: $P_0 \leftarrow \text{initialFundValue}$
$t \leftarrow 0$
**while** market is open **do**
    **Step 1:** Calculate Fund Value at time $t$:
    $P_t \leftarrow P_{t-1} \times (1 + k \times R_t)$
    **Step 2:** Determine if rebalancing is needed:
    **if** TimeSinceLastRebalance($t$) $\geq F + \Delta t$ **or** ConditionMetForRebalancing($t$) **then**
        **Step 3:** Calculate Target Exposure:
        $targetExposure_t \leftarrow k \times P_t$
        **Step 4:** Adjust Fund Exposure:
        $exposureAdjustment_t \leftarrow targetExposure_t - currentExposure_t$
        **Step 5:** Update Fund Value:
        $P_t \leftarrow P_t + exposureAdjustment_t$
        **Step 6:** Account for Rebalancing Slippage (if volatility or frictions exist):
        **if** $\sigma_t >$ threshold **or** market frictions exist **then**
            $slippage_t \leftarrow \text{CalculateRebalancingSlippage}(\sigma_t, \text{marketConditions}_t)$
            $P_t \leftarrow P_t - slippage_t$
        **end if**
        **Step 7:** Introduce random delay (up to 30 minutes):
        $\Delta t \leftarrow \text{Random}(0, 30 \text{ minutes})$
        LogRebalance($t$, $P_t$)
    **end if**
    **Step 8:** Move to the next time step:
    $t \leftarrow t + 1$
**end while**
**Function Definitions:**
**Function TimeSinceLastRebalance($t$):**
    Return the time interval since the last rebalancing event: $t - t_{lastRebalance}$
**Function ConditionMetForRebalancing($t$):**
    Additional condition checks (e.g., market volatility spikes or leverage greater than threshold): $return\ condition\_flag$
**Function CalculateRebalancingSlippage($\sigma_t$, marketConditions$_t$):**
    Calculate slippage based on volatility or market conditions:
    $slippage \leftarrow \sigma_t \times \text{transactionCosts} + \text{priceImpact}_t \times \text{liquidityConstraints}$
**Function Random(min, max):**
    Return a random number between min and max: $min + (max - min) \times rand()$
```

ized tokens and fixes issues in decentralized tokens, as outlined briefly below.

6.5.1 *Self-custodial:* LeverEdge is deployed on the blockchain, offering token holders complete control of ownership. This enables them to move the token to their cryptocurrency wallets or to other individuals whenever they want.

6.5.2 *Transparency in total supply:* LeverEdge incorporates the ERC-20 interface, which allows for the retrieval of the total supply using the `totalSupply` variable. It can be then used for calculating the NAV to trade at fair prices.

6.5.3 *Transparency in transactions:* All token flow is fully transparent due to the public recording of every transaction on the blockchain.

6.5.4 *Transparency in token holders:* The ERC-20 `Transfer` events can be used to retrieve the token holders and their corresponding token balances. Each token transfer is logged on the blockchain, allowing for the recognition of owners. This helps in evaluating liquidity and identifying potential risks.

6.5.5 *Interoperability with DeFi:* Being an ERC-20 token, LeverEdge is able to communicate with other DeFi platforms that support the ERC-20 standard.

6.5.6 *Transparency in financial backing:* The futures positions can be validated through perpetual providers or by verifying the fund value using the `getBasketValue()`.

6.5.7 *Ability to audit:* The source code is made available on the blockchain for auditors to confirm token functionality and check for security weaknesses.

6.5.8 *Front-running protection:* The Decentralized Autonomous Network (DAN) schedules rebalancing with a random 30-minute delay. LeverEdge uses a random ordering method to hide the exact rebalancing time. This alleviates front-running by making sure that trades are not executed before the rebalancing event (Algorithm 1). A network of automation servers distributed by Chainlink also guarantees required triggers for on-chain rebalancing.

6.5.9 *Adherence to leverage:* The token's leverage is fixed, converging to the stated multiplier on a daily basis. Furthermore, interim rebalancing is activated during volatile markets when a 10% threshold is reached.

6.5.10 *Lower management fees:* Futures contracts are used to minimize the impact of interest payments. Moreover, MultiCall transactions combine various operations in one block, reducing transaction cost. Executing token transactions on L2 provides additional savings in costs.

As shown in Table 6.3, most of the identified issues have been addressed in LeverEdge. But some inherent characteristics of the blockchain make it challenging to eliminate some issues completely, including: (i) the possibility of front-running cannot be fully addressed. Regardless of the technique used, ultimately miners have access to transactions and a malicious miner can prioritize its own transaction before the fund transaction. Therefore, in our design, the risk of front-running by other traders has been taken into account, (ii) to avoid frequent rebalancing and reduce costs, the rebalancing process can be delayed for minor deviations. For instance, the leverage of a 2x token can fluctuate in the range of [1.9x, 2.1x]. This range can be tightened, which increases the management fees of the token. Therefore, this is a trade-off between the fund's costs and the acceptable amount of deviation from the stated leverage, which is specific to the issuer.

In summary, in LeverEdge, we have fully addressed eight of the ten raised issues, and the other two are related to the nature of the blockchain and preference of the issuer, respectively.

6.5.7 Blockchain Implementation

LeverEdge is deployed on the Ethereum Testnet, where various DeFi partners run active instances.[6] This allows for thorough testing of the token before its official launch on the Mainnet. As is widely known, the EVM[7] can only execute low-level code known as bytecode (binary data). Developers typically write smart contracts in high-level programming languages like Solidity or Vyper, which are then compiled into bytecode for execution. This creates a challenge when analyzing smart contracts, as security assessments often rely on the bytecode rather than the original source code. Several techniques, similar to those used in traditional software quality assurance, are employed to inspect smart contracts for vulnerabilities (*e.g.,* static analysis, dynamic analysis, taint analysis, *etc.*). While these automated tools do not guarantee absolute security, they offer valuable insights and checks to help identify potential issues.

We used Slither (by Trail of Bits) and MythX (by ConsenSys) for auditing smart contracts of LeverEdge. Slither relies on static analysis and taint analysis. MythX uses a combination of static analysis, symbolic execution, and fuzzing for a more comprehensive dynamic analysis [120, 100, 46]. Slither flagged potential vulnerabilities related to the *low-level call* in the `call.value()` method, which addresses the

[6]LeverEdge implementation on the Testnet: `http://tiny.cc/ikblzz`

[7]A decentralized computing environment that executes smart contracts on the Ethereum network.

freezing Ether issue by allowing the owner to withdraw ETH. Following the *Istanbul* hard fork (EIP-1884), using `call.value()` is the recommended method for transferring ETH from smart contracts [77]. MythX, the second audit tool, identified a potential *Re-entrancy* attack in the *noReentrancy* modifier [31]. Modifiers in Solidity are commonly used to enforce checks before executing functions [63], and employing them is a standard technique for mitigating re-entrancy attacks [84]. Since Slither did not detect this issue, it appears specific to MythX. Both reported issues were deemed false positives, and the code successfully passed all required security checks.

6.5.8 Simulation Results

We demonstrate how LeverEdge can handle current challenges effectively. Introducing a 30-minute variation in the timing of rebalancing conceals the specific time of rebalancing, minimizing the possibility of front-running (Algorithm 1). Furthermore, the use of MultiCall transactions reduce management fees by as much as 14%. The majority of token operations have been carried out on Layer 2, leading to a reduction in transaction costs and daily rebalancing expenses of up to 99.6%. This reduces the negative effect of fees on the erosion of token value.

Performance Testing

We assess the performance of LeverEdge by connecting to a local copy of the Mainnet, replicating the current state of the Ethereum blockchain. This setup allows for comprehensive testing with minimal cost and delay, providing valuable insights into

LeverEdge's behavior. Compared to Ethereum Testnets, this method enables instantaneous transaction processing and offers full control over the blockchain's behavior during testing. Nonetheless, the contract code has also been published on the Goerli[8] test network for external validation and broader accessibility.[9]

Testing Platform: We employ Foundry, a smart contract development framework for Ethereum, created by Paradigm [79]. It is designed to streamline the testing, and deployment of smart contracts by offering the following key components:

- *Anvil:* It is an Ethereum node simulator, enabling developers to spin up a blockchain environment locally [78]. It supports fast testing, debugging, and simulating different conditions for Ethereum smart contracts using real blockchain data. Anvil supports methods like `eth_estimateGas()` for simulating how much gas a transaction would cost. Additionally, we can use tracing and debugging methods like `debug_traceTransaction()` to investigate failed transactions when testing LeverEdge.
- *Forge:* It is the core component of Foundry, allowing developers to compile, deploy, and test smart contracts. We used it to write and run unit tests and interact with LeverEdge through transactions, calls, and state modifications. Tests are written as functions in Solidity and can simulate different accounts and contract states.

In summary, we utilized the Forge testing framework to write and execute unit

[8] *Goerli* and *Sepolia* are two test networks launched in 2019 and 2021. Ethereum developers use them to test their applications before deploying on the Mainnet.

[9] LeverEdge implementation on the Testnet: `http://tiny.cc/ikblzz`

tests. The code interacts with the local instance of Anvil to mimic both the transaction and state behavior of the Ethereum network. Additionally, logs and required metrics are collected throughout the testing process for further analysis. Before proceeding with the analysis, it is crucial to estimate gas fees on Ethereum to understand potential operational costs more accurately.

Gas Fee Estimate: As we know, transaction fees on Ethereum are measured in Gwei[10] and fluctuate based on network congestion. Each transaction on the Ethereum network consumes a certain number of gas units, which varies depending on the complexity of the transaction (*e.g.,* sending ETH is less costly than interacting with a smart contract). The total gas cost in ETH for each transaction can be calculated as *Gas price in* Gwei $\times$ *Gas units used* $\times 10^{-9}$. This cost can then be converted to USD using *Gas cost in* ETH $\times$ ETH *price in* USD.

Example 37. Alice uses Uniswap to swap ETH for USDC with the gas price set at 20 Gwei, and the transaction requiring 210,000 gas units. The total transaction cost for Alice on the Mainnet is calculated as $20 \times 210,000 \times 10^{-9} = 0.0042\,\text{ETH}$. Assuming the price of Ether is currently \$2,500, the cost of this swap in USD would be $0.0042 \times \$2,500 = \$10.5\,\text{USD}$.

To check how the token works in different market conditions, we first need to estimate the average gas price for the entire year. Figure 6.5 displays the maximum

[10]Gwei is the standard unit for expressing gas prices, where 1 Gwei equals 0.000000001 ETH, or 10^{-9} ETH. Since gas prices are typically small fractions of Ether, using Gwei simplifies the representation, making it more practical by avoiding cumbersome decimal places in ETH. For example, a gas price of 20 Gwei is much easier to interpret than 0.000000020 ETH.

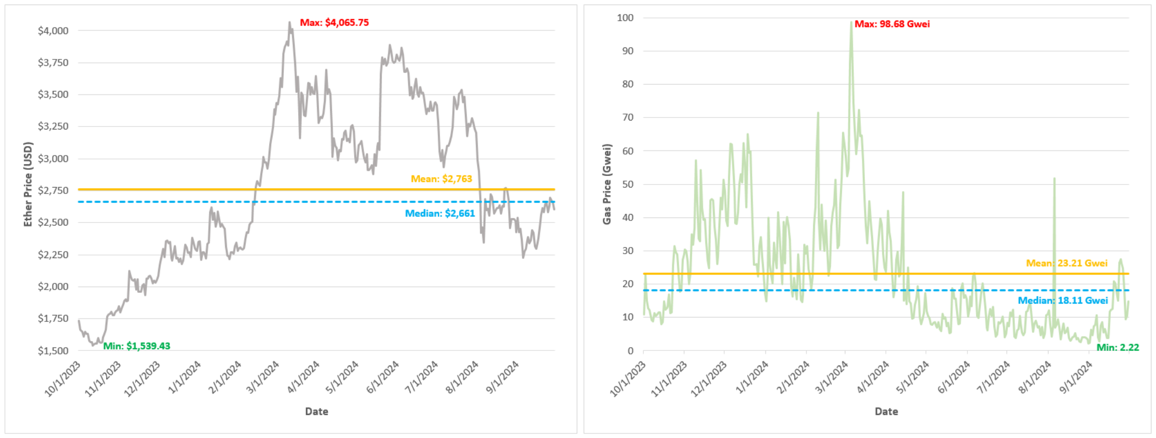


Figure 6.5: Ethereum price in USD (left) and gas prices in Gwei (right) fluctuated from Sep-2023 to Sep-2024. Gas fluctuations were influenced by periods of high congestion or increased demand for transaction throughput. The maximum gas price during this period was 98.68 Gwei, recorded on March 5, 2024, while the minimum gas price was 2.22 Gwei on August 31, 2024 (source: `etherscan.io/chart`).

and minimum Ethereum price (left chart) and gas prices (right chart) from Sep-2023 to Sep-2024. The estimated average gas fee over this period, calculated using Equation (6.6), is 23.21 Gwei (represented by the orange line in Figure 6.5).

$$\text{Average Gas Price} = \frac{\sum \text{Daily Gas Prices}}{\text{Number of Days}} = \frac{8473.40 \text{ Gwei}}{365} = 23.21 \text{ Gwei} \tag{6.6}$$

This average is sensitive to outliers (extremely high or low gas prices) and may be skewed upward. As shown in the right chart, there are noticeable spikes in gas fees due to high network congestion and major events. We can also calculate the median, which is 18.11 Gwei. It represents the middle value of the gas prices (the dashed blue line in Figure 6.5).

Unlike the average, the median is less sensitive to extreme outliers and provides a more realistic reflection of the typical gas price. Since Ethereum gas fees can be highly volatile due to sudden demand surges, the median often serves as a better measure of the typical gas fee. However, as our objective is to estimate the average cost over time (including periods of high activity), the average is more appropriate and will be used in our analysis. In future calculations, we will use 23.21 Gwei and \$2,763 as the fixed gas fee on the Mainnet and ETH price, respectively.

Since the main token processing occurs on Layer 2, there is a need to consider gas prices on Arbitrum. From Sep-2023 to Sep-2024, the average gas fee on the Arbitrum network was 0.09 Gwei. This consistently low fee is a result of Arbitrum's Layer 2 architecture, which helps maintain gas prices significantly lower than those on Ethereum's Layer 1 network.

Rebalancing Cost

To estimate daily rebalancing cost for LeverEdge, we need to break down the Ethereum gas costs based on the interaction with each application:

- *Interaction with Uniswap V3:* Swapping ETH to USDC involves a `swap` function, which typically costs around 120K to 180K gas depending on the liquidity conditions. We assume an average gas cost of 150K gas for the swap operation.
- *Interaction with GMX:* The cost of interacting with the GMX protocol to buy and sell perpetual via USDC requires executing smart contract calls to open and close positions. These operations are relatively complex and typically consume around 300K to 400K gas per transaction. For the purpose of our analysis, we assume an average of 350K gas for a typical buy/sell transaction.

- *Chainlink Keeper Network:* The keeper service triggers the rebalancing at a daily interval. The rebalancing trigger itself can vary, but Chainlink Keeper services typically use around 200K gas for basic triggers. Randomized delays add some variability as well. We assume 250K gas for each rebalancing trigger.

The total required gas is the sum of all interactions. Assuming the sequence of operations for a single rebalancing includes: (i) one Uniswap V3 swap for required conversions at 150K gas, (ii) one GMX buy/sell operation for perpetuals at 350K gas, and (iii) one Chainlink Keeper rebalancing trigger at 250K gas, the total gas consumption per day is $150K + 350K + 250K = 750K$ gas unit.

The cost of executing transactions on Ethereum Mainnet is influenced by network gas prices and the price of ETH in USD. As previously estimated, the average gas price is 23.21 and 0.09 Gwei on the Mainnet and Arbitrum, respectively. Therefore, the total cost of rebalancing for the required 750K gas, when ETH is priced at \$2,763, can be calculated as follows:

$$\begin{aligned}\text{Mainnet rebalancing cost} &= \text{Gas units} \times \text{Gas fee} \times 10^{-9} \times \text{ETH price in USD} \\ &= 750K \times 23.21 \text{ Gwei} \times 10^{-9} \times \$2,763 \\ &= 0.0174075 \text{ ETH} \times \$2,763 = \$48.10 \text{ USD}\end{aligned}$$

Thus, based on average gas price conditions, the estimated daily rebalancing cost is \$48.10 on the Mainnet. However, network congestion can lead to significant cost variations. As demonstrated in Figure 6.5, when the gas price on the Mainnet dropped to 2.22 Gwei, the daily cost decreased to just \$4.56, representing a 90% reduction.

Conversely, during congestion spikes, costs surged to as much as $204.49, a 426% increase when the gas price reached 98.68 Gwei. To mitigate these fluctuations, we have implemented specific techniques to reduce overall rebalancing expenses.

Employing MultiCall Transactions: Bundling multiple operations into a single transaction can reduce costs by minimizing redundant gas expenditures that occur when each transaction is executed separately [98]. Chainlink Keeper can trigger a single atomic bundle that executes all operations (Uniswap swap, dYdX/GMX Perps trade, and the rebalancing trigger itself) in one transaction. In this approach, all three operations share the transaction overhead costs (such as state updates and transaction initialization), further reducing the total gas consumption. Our analysis indicates that the 750K gas required for rebalancing can be reduced by an average of 14% using MultiCall transactions.

Using L2 Chains: L2 solutions reduce gas costs by enhancing transaction efficiency, batching multiple transactions, and leveraging Ethereum's Mainnet solely for security and final settlement [82]. By consolidating many transactions and posting them to Ethereum as a single entry, they distribute gas costs across a larger number of users, lowering individual fees. This allows users to benefit Ethereum's robust security while paying only a small fraction of the gas fees compared to the Mainnet.

Example 38. On Ethereum Mainnet, the average cost for a token swap is approximately $4.03, fluctuating between $3.99 and $4.54 depending on network congestion. In contrast, L2 networks like Arbitrum and Optimism can reduce these fees dramatically, often bringing the cost down to between \$0.09 and \$0.18 per swap [42], representing more than 95% savings in transaction costs.

LeverEdge implements a gateway contract on L1 to establish communication with the main contract on L2. This gateway is configurable and can connect to any L2 network where the main contract is deployed. We currently use Arbitrum and StarkEx, where active instances of GMX and dYdX already operate. dYdX offers perpetual contracts on various assets and is known for deep liquidity and zero gas fees. Comparatively, GMX provides low-cost, low-slippage trading on Arbitrum. While the zero gas fees on dYdX can fully eliminate rebalancing costs, issuers may still prefer GMX on Arbitrum due to its popularity among retail traders. In this case, the rebalancing cost would still be $1 - \frac{\$0.19}{\$48.10} = 99.6\%$ cheaper than on the Mainnet.

$$\begin{aligned}
\text{Arbitrum rebalancing cost} &= \text{Gas units} \times \text{Gas fee} \times 10^{-9} \times \text{ETH price in USD} \\
&= 750K \times 0.09 \text{ Gwei} \times 10^{-9} \times \$2,763 \\
&= 0.0000675 \text{ ETH} \times \$2,763 = \$0.19 \text{ USD}
\end{aligned}$$

6.6 Contributions

The identified deficiencies in Chapter 5 inspired us to consider possible solutions. We propose that operating on-chain mitigates most concerns (see Table 6.1) but building an on-chain LVT is non-trivial. The details of the design of LeverEdge are discussed through the following research questions:

RQ 1: What potential solutions could be proposed to address the current deficiencies of LVTs? Implementing LVT on the blockchain can enhance transparency in total supply, transactions, holders, custody, auditing, and financial backing. On-chain implementation should be adopted by LVTs to mitigate risks in centrally issued tokens. Despite proposed solutions for front-running, miners may still exploit this by including and delaying rebalancing transactions to prioritize their own, so the risk can be reduced but not fully eliminated.

To reduce tracking error: (i) increase rebalancing frequency to n blocks, potentially using L2 chains for lower fees; (ii) perform interim rebalancing at asset price thresholds (*e.g.,* $\pm 10\%$), mitigating sudden price impacts; (iii) allow leverage variation (*e.g.,* [1.95x, 2.05x] for 2x tokens) to limit frequent rebalancing. This strategy should be explained to investors as a workaround rather than a definitive fix.

Management fees in LVTs can be reduced by: (i) employing algorithms to optimize trade timing and size, minimizing slippage and costs; (ii) cross-trading between long and short funds, allowing internal matching to cut costs; (iii) leveraging L2 chains and MultiCall transactions to further reduce expenses.

RQ 2: What mitigation has been already done, and how far have these efforts been successful in eliminating the problems? Even though there have been notable attempts to decentralize LVTs by FLI, Contango, Cube, Squeeth, Toros and TLX tokens, a closer look at their functionality shows ongoing shortcomings that need to be addressed. In this regard, we propose and evaluate a new L1-L2 hybrid model, LeverEdge, and compare how well it performs against others.

RQ 3: Can a new decentralized design, LeverEdge, succeed in effectively eliminating the existing shortcomings? As shown in Table 6.3, most of the identified issues have been addressed in LeverEdge. But some inherent characteristics of the blockchain make it challenging to eliminate some issues completely, including: (i) the possibility of front-running cannot be fully addressed. Regardless of the technique used, ultimately miners have access to transactions and a malicious miner can prioritize its own transaction before the fund transaction. Therefore, in our design, the risk of front-running by other traders has been taken into account, (ii) to avoid frequent rebalancing and reduce costs, the rebalancing process can be delayed for minor deviations. For instance, the leverage of a 2x token can fluctuate in the range of [1.9x, 2.1x]. This range can be tightened, which increases the management fees of the token. Therefore, this is a trade-off between the fund's costs and the acceptable amount of deviation from the stated leverage, which is specific to the issuer.

RQ 4: How is the efficiency of LeverEdge on the Ethereum blockchain compared to the efficiency of existing decentralized LVTs? LeverEdge has addressed eight of the ten raised issues, and the other two are related to the nature of the blockchain and preference of the issuer, respectively. Our fully decentralized design is the first to deploy LVTs by utilizing decentralized perpetuals instead of debt or synthetic positions. This method addresses the shortcomings of centralized tokens and resolves issues in existing decentralized implementations. Key advantages :

- Full ownership control for token holders through blockchain-based custody.
- Full transparency of total supply and transactions, recorded on the blockchain.
- Seamless interoperability with DeFi and availability for on-chain auditing.
- Verifiable financial backing via decentralized perpetual exchanges.
- Alleviation of front-running through randomized rebalancing, triggered by a decentralized autonomous network provided by Chainlink.
- Fixed leverage, with interim rebalancing to manage market volatility.
- Lower costs achieved through MultiCall transactions and L2 deployment.

6.7 Discussion

Leveraged tokens (LVTs) aim to mirror the advantages of leveraged ETFs (LETFs) seen in traditional markets. LVTs essentially combine the functionality of futures contracts or debt positions into a single token that can be traded on the spot market. Although they have the potential to result in significant profits, they also have the capability to increase potential losses. Despite involved risks, traders find LVTs appealing due to their simplicity, which removes the requirement for constant supervision of leveraged positions. In contrast to derivatives and margin trading, LVTs provide a less risky option with relatively modest returns, making them attractive to traders looking for a balanced risk-reward ratio.

Since 2019, over 1,600 leveraged tokens have been offered by various issuers. Previous research has identified ten key deficiencies in the technical and financial performance of these tokens. In this work, we review these issues and propose practical solutions. Decentralization stands out as the primary approach for resolving these deficiencies, along with enhancements to the internal algorithms governing the tokens. We also assess the performance of six decentralized tokens, identifying areas where further optimization is still needed.

To address these shortcomings, we introduce LeverEdge as a fully decentralized LVT, integrating the proposed solutions. Utilizing a hybrid L1-L2 architecture allows for seamless interaction with higher L1 liquidity while taking advantage of the faster execution and reduced fees on L2 chains. We evaluate LeverEdge in similar situations as other LVTs, demonstrating its capability to tackle the recognized flaws. It employs perpetual futures for generating leverage and incorporates a cross-chain mechanism for compatibility with various L2 ecosystems. Its design is focused on composability and deployed on the Ethereum blockchain. The open-source code has successfully passed security audits, serving as a blueprint for developing new decentralized LVTs or transitioning current centralized versions to decentralized options.

Chapter 7

Concluding Remarks

7.1 The Broader Context

ERC-20 remains the dominant token standard in the Ethereum ecosystem, and resolving longstanding vulnerabilities within its framework is essential for protecting investors. One achievement in this research was addressing the *"Multiple Withdrawal Attack"* in 2019, an issue that has persisted since 2016. Following this, our study extended its scope in 2020 to investigate other potential security concerns within ERC-20 implementations. By systematizing 82 distinct vulnerabilities and best practices applicable to ERC-20, the research laid the groundwork for improved token safety. TokenHook, developed in 2021 as part of this work. It addresses these vulnerabilities and provides stronger compliance and security compared to the top 10 ERC-20 tokens currently in use.

Beyond technical improvements, this work focused in 2022 on the economic implications of ERC-20's use as a derivative token, particularly within the context of leveraged tokens (LVTs). The analysis identified 10 key shortcomings, ranging from transparency issues to risks associated with market volatility and rebalancing mechanisms. Solutions were proposed for each identified problem, demonstrating the potential for better investor outcomes through targeted improvements.

To understand the landscape further in 2023, the research reviewed six existing decentralized LVTs to evaluate their effectiveness in mitigating these issues. This evaluation revealed gaps in current approaches and underscored the need for a more comprehensive solution. In this respect, LeverEdge was proposed in 2024 as an hybrid L1-L2 model that (i) inherits the enhanced security properties of TokenHook and (ii) addresses most of the economic and operational deficiencies found in current LVTs. By integrating these advancements, LeverEdge ensures a more secure and reliable investment framework for users.

The summary of contributions from 2019 to 2024 is shown in Figure 7.1, highlighting that TokenHook directly addresses vulnerabilities inherent in ERC-20 tokens, enhances security, and safeguards users against potential threats such as the *"Multiple Withdrawal Attack"*. LeverEdge, on the other hand, addresses the structural and operational deficiencies found in current leveraged tokens (LVTs), providing a more secure and transparent model for users seeking leveraged exposure to crypto-assets.

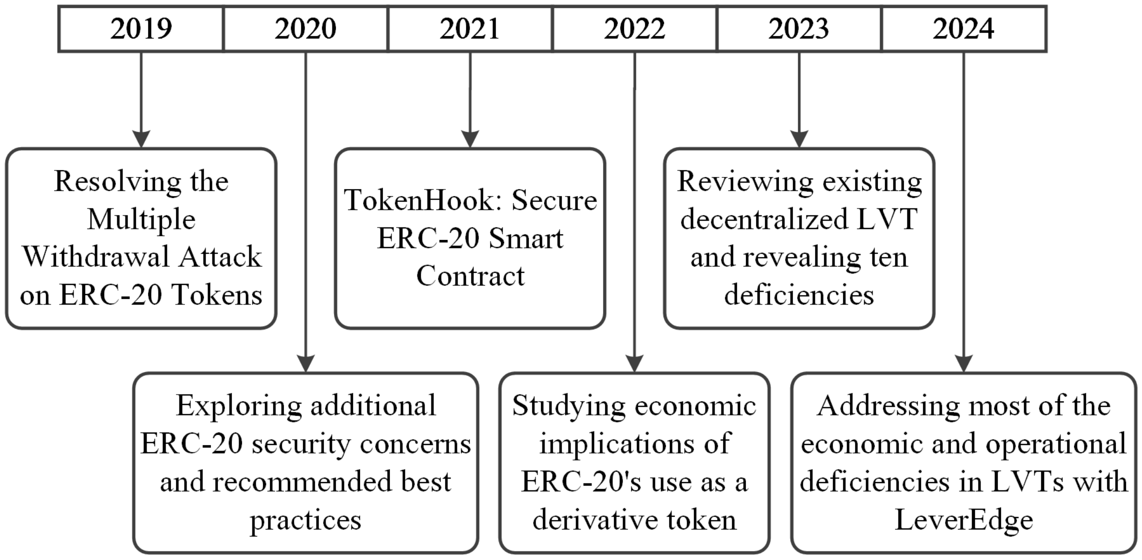

2019
2020
2021
2022
2023
2024
Resolving the Multiple Withdrawal Attack on ERC-20 Tokens
TokenHook: Secure ERC-20 Smart Contract
Reviewing existing decentralized LVT and revealing ten deficiencies
Exploring additional ERC-20 security concerns and recommended best practices
Studying economic implications of ERC-20's use as a derivative token
Addressing most of the economic and operational deficiencies in LVTs with LeverEdge

Figure 7.1: The research contributions timeline from 2019 to 2024 began with resolving the *"Multiple Withdrawal Attack"* on ERC-20 tokens in 2019, followed by exploring additional ERC-20 security issues in 2020, developing TokenHook as a secure ERC-20 smart contracts in 2021, studying ERC-20's economic implications in 2022, identifying LVT deficiencies in 2023, and concluding with addressing LVT shortcomings using LeverEdge in 2024.

7.2 Recommendations for Ethereum

This work focuses on strengthening investor protection by improving security, user experience, and economic performance within the Ethereum ecosystem. Recommended changes and lessons learned from this work are as follows.

7.2.1 Reform of Approval Mechanism

The current approval process poses risks, particularly through indefinite allowances. To enhance investor protection, the Ethereum ecosystem should consider more secure approval models such as session-based (time-limited) or single-use approvals. Session-based approvals allow users to grant token allowances for a specific period or session, automatically expiring afterward. Single-use approvals permit a one-time token transfer, after which the allowance is revoked. These mechanisms reduce the risk of unauthorized or forgotten permissions, enhancing overall security.

7.2.2 Prioritization of Withdrawal Mechanism Security

TokenHook's approach to addressing vulnerabilities such as the *"Multiple Withdrawal Attack"* highlights the importance of secure withdrawal protocols. Establishing standards for secure withdrawal practices can ensure consistent safety across ERC-20 implementations, reducing investor risk.

7.2.3 User-Friendly and Transparent Token Interactions

Simplifying how users interact with tokens can enhance protection by reducing user error. Recommendations include wallet-level support for contextual allowances and token-level permissions. Integrating clear interfaces and notifications regarding token approvals, their limits, and duration can empower users to make informed decisions and prevent common mistakes such as granting excessive or indefinite permissions. Transparent interaction mechanisms should also include detailed transaction summaries that explain the purpose and consequences of each action, ensuring users know exactly how their tokens are being utilized.

7.2.4 Continuous Focus on Cost and Performance

Leveraged tokens rely on perpetual futures and high gas fees and scalability limitations on L1, acts as an obstacle to their development. This serve as a reminder that Ethereum's long-term success depends on its ability to keep transaction costs reasonable. Future updates should aim to improve Ethereum's L1 performance while ensuring L2 solutions are sustainable and integrated tightly into the ecosystem.

7.3 Future Work

The findings of this research underscore the importance of investor protection within the Ethereum ecosystem, achieved through targeted solutions addressing specific vulnerabilities and deficiencies in ERC-20 tokens. The development of TokenHook and LeverEdge exemplifies how focused improvements can enhance the safety and reliability of DeFi interactions. But still work on the following in the future can create more resilient, transparent, and secure instruments align with investor expectations.

7.3.1 Improved Rebalancing Algorithms

One of the primary issues with LVTs is the volatility decay resulting from frequent rebalancing. Future research could explore more efficient and adaptive rebalancing algorithms that minimize value erosion, especially in high-volatility markets. Machine learning techniques could be applied to predict optimal rebalancing times, reducing the negative impact of market swings on leveraged positions.

7.3.2 Enhancing the Security of Other Fungible Tokens

Future research on ERC-20 tokens should broaden its focus to include other types of fungible tokens, such as ERC-777 and ERC-1155 tokens.

7.3.3 Cross-Chain Leveraged Tokens

Future work could investigate creating LVTs that function seamlessly across multiple blockchain ecosystems. Cross-chain capabilities could improve access to different DeFi platforms and liquidity pools, enhancing the usability and adoption of LVTs while reducing exposure to a single network's limitations or issues.

7.3.4 Leveraging Account Abstraction for Enhanced User Protection

Account abstraction (AA) has the potential to improve user protection by allowing transactions to be bundled and executed under user-defined conditions [53]. It can integrate security checks directly into wallet operations. This makes it possible for users to implement tailored permissions and rules for token interactions, adding an extra layer of protection on top of the built-in safeguards of the TokenHook and LeverEdge. While AA complements the work done on the token contract level, future research could explore how AA can be used to create safer, user-controlled mechanisms for managing LVT trades based on pre-set criteria.

Bibliography

[1] M. A. Abd Wahab, A. Mohamad, and I. Sifat. On contango, backwardation, and seasonality in index futures. *The Journal of Private Equity*, 22(2):69–82, 2019.

[2] C. S. Alliance. What is a pseudo cryptocurrency? `https://cloudsecurityalliance.org/blog/2019/11/25/what-s-a-pseudo-cryptocurrency/`, Nov 2019. [Online; accessed 16-May-2023].

[3] K. H. Ansari and U. Kulkarni. Implementation of ethereum request for comment (erc20) token. In *Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST)*, 2020.

[4] A. M. Antonopoulos and G. Wood. *Mastering Ethereum: Building Smart Contracts and DApps.* O'Reilly Media, Inc., 2018.

[5] K. Aouane. Why you should trade on leverage using money markets. `https://medium.com/contango-xyz/why-you-should-trade-on-leverage-using-money-markets-93b988fc69ec`, Feb. 2024. [Online; accessed 30-Aug-2024].

[6] AscendEX. About ascendex's leveraged token trading mechanism. `https://ascendex.com/en/support/articles/68525-about-ascendexs-leveraged-token-trading-mechanism`, May 2023. [Online; accessed 07-May-2023].

[7] C. Baum, J. Hsin-yu Chiang, B. David, T. K. Frederiksen, and L. Gentile. Sok: Mitigation of front-running in decentralized finance. In *International Conference on Financial Cryptography and Data Security*, pages 250–271. Springer, 2022.

[8] L. Bertagnoli. 22 top blockchain platforms to know, 2024. [Online; accessed 16-Oct-2024].

[9] Binance. Binance leveraged tokens agreement. `https://www.binance.com/en/leveraged-tokens/agreement`, May 2023. [Online; accessed 07-May-2023].

[10] T. Block. Value locked by blockchain. `https://www.theblock.co/data/decentralized-finance/total-value-locked-tvl/value-locked-by-blockchain`, 2024. [Online; accessed 16-Oct-2024].

[11] L. Breidenbach, P. Daian, A. Juels, and E. Gun Sirer. An in-depth look at the parity multisig bug. `https://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/`, Jul 2017. [Online; accessed 16-Oct-2024].

[12] T. N. Bulkowski. *Encyclopedia of Chart Patterns.* John Wiley & Sons, Hoboken, NJ, 2nd edition, 2011.

[13] ByBit. Bybit leveraged tokens: A complete get started guide. `https://learn.bybit.com/trading/bybit-leveraged-tokens-explained/`, May 2023. [Online; accessed 07-May-2023].

[14] ByDFi. What are leveraged tokens? `https://support.bydfi.com/hc/en-us/articles/5692456018319-What-are-Leveraged-Tokens-`, May 2023. [Online; accessed 07-May-2023].

[15] M. Charm Finance. Cube tokens. `https://github.com/charmfinance/cube-protocol`, Mar. 2021. [Online; accessed 22-Aug-2024].

[16] N. Charupat and P. Miu. The pricing and performance of leveraged exchange-traded funds. *Journal of Banking & Finance*, 35:966–977, 2011.

[17] H. Chen, M. Pendleton, L. Njilla, and S. Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *arXiv*, 1908.04507, 2019.

[18] J. Chen. Open-ended fund: Definition, example, pros and cons. `https://www.investopedia.com/terms/o/open-endfund.asp`, Dec 2021. [Online; accessed 02-Jun-2023].

[19] T. Chen, Z. Zhang, and Z. Li. Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum. `http://www4.comp.polyu.edu.hk/~csxluo/TokenScope.pdf`, Nov 2019.

[20] M. Cheng and A. Madhavan. The dynamics of leveraged and inverse exchange-traded funds. *Journal of investment management*, 16(4):43, 2009.

[21] M. Cheng and A. Madhavan. The dynamics of leveraged and inverse exchange-traded funds. *Journal of Investment Management*, 16(4):43, 2009.

[22] M. Cheng and A. Madhavan. The dynamics of leveraged and inverse exchange-traded funds. *Journal of Investment Management*, 7, 2010.

[23] J. Chiu, E. Ozdenoren, K. Yuan, and S. Zhang. On the fragility of defi lending. *SSRN Electronic Journal*, 2023.

[24] CoinDesk. The coindesk 20. `https://www.coindesk.com/coindesk20`, Jul 2020. [Online; accessed 16-Oct-2024].

[25] CoinGecko. Defi trends to watch in 2024. `https://www.coingecko.com/en/article/defi-trends-2024`, 2024. [Online; accessed 19-Oct-2024].

[26] Coinglass. Funding rates for perpetual swaps. `https://www.coinglass.com/FundingRate`, Sept. 2024. [Online; accessed 29-Sep-2024].

[27] E. Community. Resolution on the eip20 api approve / transferfrom multiple withdrawal attack. GitHub Issue Discussion, 2017. Accessed: 2024-02-11.

[28] ConsenSys. ConsenSys/Tokens. https://github.com/ConsenSys/Tokens/blob/fdf687c69d998266a95f15216b1955a4965a0a6d/contracts/eip20/EIP20.sol, Apr. 2018. [Online; accessed 24-Dec-2018].

[29] ConsenSys. Mythx swc coverage. https://mythx.io/swc-coverage/, Nov 2019.

[30] ConsenSys. Swc registry. https://swcregistry.io, May 2020.

[31] Consensys. Reentrancy. https://consensys.github.io/smart-contract-best-practices/attacks/reentrancy/, Oct. 2021. [Online; accessed 08-Jun-2023].

[32] W. Contributors. Compare-and-swap. Wikipedia, The Free Encyclopedia, 2018. Accessed: 2024-02-11.

[33] I. Coop. How composability powers fli tokens. https://indexcoop.com/blog/how-composability-powers-fli-tokens, Sep 2021. [Online; accessed 17-May-2023].

[34] I. Coop. Set protocol v2. https://docs.indexcoop.com/index-coop-community-handbook/protocol/set-protocol-v2, 2023. [Online; accessed 04-Oct-2024].

[35] T. I. Coop. Tokens restricted for restricted persons. https://indexcoop.com/tokens-restricted-for-restricted-persons, May 2024. [Online; accessed 29-Aug-2024].

[36] S. Corporation. What are leveraged tokens? https://medium.com/coinmonks/what-are-leveraged-tokens-639fb186744a, Aug 2022. [Online; accessed 09-Jun-2023].

[37] C. Coverdale. Transaction-ordering attacks. https://medium.com/coinmonks/solidity-transaction-ordering-attacks-1193a014884e, Mar 2018.

[38] L. Cremer. Missing return value bug — at least 130 tokens affected. https://medium.com/coinmonks/missing-return-value-bug-at-least-130-tokens-affected-d67bf08521ca, Jun 2018.

[39] G. T. Crypto. Etf leveraged tokens. https://www.gate.io/etf, May 2023. [Online; accessed 07-May-2023].

[40] J. Dafflon, J. Baylina, and T. Shababi. EIP 777: A New Advanced Token Standard. https://eips.ethereum.org/EIPS/eip-777, Nov. 2017. [Online; accessed 12-Jan-2019].

[41] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *IEEE Security and Privacy on the Blockchain (IEEE S&B)*, 2020.

[42] DailyCoin. Types of layer 2: Comparing different ethereum scaling solutions. *DailyCoin*, 2024. [Online; accessed 03-Oct-2024].

[43] L. DeVault, H. J. Turtle, and K. Wang. Blessing or curse? institutional investment in leveraged etfs. *Journal of Banking & Finance*, 129:106169, 2021.

[44] dHEDGE. Technical architecture. https://docs.dhedge.org/dhedge-protocol/technical-architecture, Mar. 2024. [Online; accessed 28-Sep-2024].

[45] M. di Angelo and G. Salzer. A survey of tools for analyzing ethereum smart contracts. https://publik.tuwien.ac.at/files/publik_278277.pdf, Aug 2019.

[46] C. Diligence. Mythx: Security analysis for ethereum smart contracts. https://docs.mythx.io/, 2019. [Online; accessed 08-Jun-2024].

[47] C. Diligence. Ethereum smart contract security best practices. https://consensys.github.io/smart-contract-best-practices/, Jan 2020.

[48] C. Diligence. Token implementation best practice. https://consensys.github.io/smart-contract-best-practices/tokens/, Mar 2020.

[49] B. Documentation. Introduction to binance futures funding rates. https://www.binance.com/en/support/faq/introduction-to-binance-futures-funding-rates-360033525031, Sept. 2019. [Online; accessed 31-Aug-2024].

[50] B. Documentation. Perpetual contracts guide. https://www.bitmex.com/app/perpetualContractsGuide, July 2024. [Online; accessed 8-Sep-2024].

[51] C. Documentation. What is contango? https://docs.contango.xyz, May 2021. [Online; accessed 30-Aug-2024].

[52] C. Documentation. Faq. https://docs.contango.xyz/resources/faq, Aug. 2024. [Online; accessed 27-Sep-2024].

[53] E. Documentation. Account abstraction. https://ethereum.org/en/roadmap/account-abstraction/, 2024. [Online; accessed 05-May-2024].

[54] O. Documentation. Opyn document hub. https://opyn.gitbook.io/opyn-hub, Jan. 2024. [Online; accessed 27-Sep-2024].

[55] O. Documentation. Opyn user terms of service. https://squeeth.opyn.co/terms-of-service?ct=CA, July 2024. [Online; accessed 13-Sep-2024].

[56] S. Documentation. Checks effects interactions pattern. https://docs.soliditylang.org/en/latest/security-considerations.html#use-the-checks-effects-interactions-pattern, Aug 2020.

[57] S. documentation. Security considerations. https://solidity.readthedocs.io/en/latest/security-considerations.html, Jan 2020.

[58] T. Documentation. Effects of rebalancing. https://docs.tlx.fi/basics/how-leveraged-tokens-work/effects-of-rebalancing, May 2024. [Online; accessed 27-Sep-2024].

[59] T. Documentation. Leveraged tokens overview. https://docs.toros.finance/leveraged-tokens/leveraged-tokens-overview, June 2024. [Online; accessed 10-Sep-2024].

[60] T. Documentation. Synthetix perps engine. `https://docs.tlx.fi/basics/how-leveraged-tokens-work/synthetix-perps-engine`, May 2024. [Online; accessed 27-Sep-2024].

[61] P. Durov. What was ton and why it is over. `https://telegra.ph/What-Was-TON-And-Why-It-Is-Over-05-12`, May 2020.

[62] R. D. Edwards and J. Magee. *Technical Analysis of Stock Trends.* CRC Press, Boca Raton, FL, 9th edition, 2007.

[63] S. C. Engineer. Function modifier. `https://solidity-by-example.org/function-modifier/`, Oct. 2019. [Online; accessed 08-Jun-2023].

[64] S. Eskandari, S. Moosavi, and J. Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. *International Conference on Financial Cryptography and Data Security*, 2019.

[65] Ethereum. Ethereum project repository. `https://github.com/ethereum`, May 2014. [Online; accessed 16-Oct-2024].

[66] Ethereum. Solidity documentation. `https://solidity.readthedocs.io/en/latest/`, Jan 2020.

[67] Ethereum Foundation. Maximal extractable value in proof-of-stake ethereum. `https://ethereum.org/en/developers/docs/mev/`, 2023. [Online; accessed 16-Oct-2024].

[68] D. Evans. Permissioned defi goes live with aave arc + fireblocks. `https://www.fireblocks.com/blog/permissioned-defi-goes-live-with-aave-arc-fireblocks`, Jan. 2022. [Online; accessed 27-Sep-2024].

[69] EY. Token review. `https://review-tool.blockchain.ey.com`, Sep 2019.

[70] V. B. Fabian Vogelsteller. Erc-20 token standard. `https://github.com/ethereum/ercs/blob/master/ERCS/erc-20.md`, Nov 2015. [Online; accessed 16-Oct-2024].

[71] G. Fenu, L. Marchesi, M. Marchesi, and R. Tonelli. The ico phenomenon and its relationships with ethereum smart contract environment. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 26–32. IEEE, 2018.

[72] C. Finance. Cube tokens faq. `https://learn.charm.fi/charm-finance/previous-products-1/overview-1/faq`, Mar. 2021. [Online; accessed 24-Aug-2024].

[73] K. Finley. A $50 million hack just showed that the dao was all too human. *Wired*, 2016. Accessed: 2024-02-11.

[74] K. Finley. A $50 million hack just showed that the dao was all too human — wired. `https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/`, Sep 2016. [Online; accessed 16-Oct-2024].

[75] S. Forrest, S. A. Hofmeyr, and A. Somayaji. Computer immunology. *Commun. ACM*, 40(10):88–96, Oct. 1997.

[76] S. Forrest, A. Somayaji, and D. H. Ackley. Building diverse computer systems. In *Proceedings. The Sixth Workshop on Hot Topics in Operating Systems (Cat. No.97TB100133)*, pages 67–72, 1997.

[77] E. Foundation. Eip-1884: Repricing for trie-size-dependent opcodes. `https://eips.ethereum.org/EIPS/eip-1884`, Mar. 2019. [Online; accessed 08-Jun-2023].

[78] Foundry. Anvil reference - foundry book. `https://book.getfoundry.sh/reference/anvil/`, 2024. [Online; accessed 07-Jun-2024].

[79] Foundry. Foundry book. `https://book.getfoundry.sh/`, 2024. [Online; accessed 07-Jun-2024].

[80] M. Fröwis, A. Fuchs, and R. Böhme. Detecting token systems on ethereum. *arXiv preprint arXiv:1811.11645*, 2018.

[81] A. K.-W. Fung, D. Y. Mok, and K. Lam. Intraday price reversals for index futures in the us and hong kong. *Journal of Banking and Finance*, 24:1179–1201, 2000.

[82] A. Gangwal, H. R. Gangavalli, and A. Thirupathi. A survey of layer-two blockchain protocols. *J. Netw. Comput. Appl.*, 209:103539, 2022.

[83] A. Ganti. Understanding notional value and how it works. `https://www.investopedia.com/terms/n/notionalvalue.asp`, Feb 2024. [Online; accessed 17-Mar-2024].

[84] E. Garcia. Reentrancy guard. `https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol`, June 2023. [Online; accessed 08-Jun-2023].

[85] G. Giese. On the performance of leveraged and optimally leveraged investment funds. *Available at SSRN 1510344*, 2010.

[86] M. Global. Guide to leveraged etf. `https://support.mexc.com/hc/en-001/articles/360038986011-Guide-to-Leveraged-ETF`, May 2023. [Online; accessed 07-May-2023].

[87] Grand View Research. Blockchain market size, share & trends analysis report by type (public, private, hybrid), by component, by application, by enterprise size, by end-use, by region, and segment forecasts, 2020 - 2027. `https://www.grandviewresearch.com/industry-analysis/blockchain-technology-market`, 2020. [Online; accessed 16-Oct-2024].

[88] C. Group et al. What is contango and backwardation. `https://www.cmegroup.com/education/`, 2020.

[89] GuardStrike. Contractguard knowledge-base. `https://contract.guardstrike.com/#/knowledge`, Mar 2020.

[90] I. Guedj, G. Li, and C. Mccann. Leveraged and inverse etfs, holding periods, and investment shortfalls. *The Journal of Index Investing*, 1(3):45–57, 2010.

[91] T. Hale. Resolution on the EIP20 API Approve / TransferFrom multiple withdrawal attack #738. https://github.com/ethereum/EIPs/issues/738, Oct 2017. [Online; accessed 5-Dec-2018].

[92] A. Hayes. Understanding liquidity and how to measure it. https://www.investopedia.com/terms/l/liquidity.asp, Sep 2023. [Online; accessed 18-Dec-2023].

[93] A. Hayes. Volatility: Meaning in finance and how it works with stocks. https://www.investopedia.com/terms/v/volatility.asp, Mar 2023. [Online; accessed 19-May-2023].

[94] J. Hill and G. Foster. Understanding returns of leveraged and inverse funds. *Journal of Indexes*, 12(5):40–58, 2009.

[95] J. Hill and G. Foster. Understanding returns of leveraged and inverse funds. *Journal of Indexes*, 12(5):40–58, 2009.

[96] M. Holst Swende. Repricing for trie-size-dependent opcodes. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1884.md, Mar 2019.

[97] L. N. Huashan Chen, Marcus Pendleton and S. Xu. A survey on ethereum systems security: Vulnerabilities, attacks and defenses. https://arxiv.org/pdf/1908.04507.pdf, Aug 2019. [Online; accessed 16-Oct-2024].

[98] W. Hughes, A. Russo, and G. Schneider. Multicall: A transaction-batching interpreter for ethereum. In *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, pages 1–3, 2021.

[99] Q. International. Nvivo. https://help-nv.qsrinternational.com/14/win/Content/about-nvivo/about-nvivo.htm, 2023. Accessed: 2025-02-12.

[100] F. Josselin. Slither – a solidity static analysis framework. https://blog.trailofbits.com/2018/10/19/slither-a-solidity-static-analysis-framework/, Oct 2018.

[101] F. Josselin. Slither – detector documentation. https://github.com/crytic/slither/wiki/Detector-Documentation, Mar 2020.

[102] M. Kaleem, A. Mavridou, and A. Laszka. Vyper: A security comparison with solidity based on common vulnerabilities. In *Proceedings of the 2nd Conference on Blockchain Research and Applications for Innovative Networks and Services (BRAINS 2020)*, Jun 2020.

[103] W. Kenton. S&p 500 index: What it's for and why it's important in investing. https://www.investopedia.com/terms/s/sp500.asp, Sep 2023. [Online; accessed 24-Mar-2024].

[104] M. Khomyn, T. J. Putniņš, and M. Zoican. The value of etf liquidity. *European Finance Association*, 2020.

[105] L. Kramer. Long position vs. short position: What's the difference? https://www.investopedia.com/ask/answers/100314/whats-difference-between-long-and-short-position-market.asp, Feb 2024. [Online; accessed 24-Mar-2024].

[106] KuCoin. Kucoin etf dynamic multiplier leveraged tokens. https://www.kucoin.com/support/18437931016857, Nov 2022. [Online; accessed 05-Jun-2023].

[107] KuCoin. Leveraged token trading guide. https://www.kucoin.com/support/sections/6024431447321, May 2023. [Online; accessed 07-May-2023].

[108] X. Labs. Advantages of blockchain in healthcare. https://www.xcubelabs.com/blog/how-blockchain-will-benefit-the-healthcare-industry-in-2022-and-beyond/, 2020. [Online; accessed 16-Oct-2024].

[109] G. Lando. Guy lando's knowledge list. https://github.com/guylando/KnowledgeLists/blob/master/EthereumSmartContracts.md, May 2019.

[110] M. Lettau and A. Madhavan. Exchange-traded funds 101 for economists. *Journal of Economic Perspectives*, 32(1):135–154, 2018.

[111] T. Leung and M. Santoli. Leveraged etfs: Admissible leverage and risk horizon. *SSRN Electronic Journal*, 2012.

[112] L. J. Liebi. The effect of etfs on financial markets: a literature review. *Financial Markets and Portfolio Management*, 34(2):165–178, 2020.

[113] A. Loviscek, H. Tang, and X. E. Xu. Do leveraged exchange-traded products deliver their stated multiples? *Journal of Banking & Finance*, 43:29–47, 2014.

[114] M. Lundfall. Erc-2612: Permit extension for eip-20 signed approvals. https://eips.ethereum.org/EIPS/eip-2612, 2020. [Online; accessed 05-Oct-2024].

[115] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. *CCS'16*, 2016.

[116] P. Mackintosh. Double trouble. *ETF and Indexing*, page 25–31, 2008.

[117] A. Manning. Comprehensive list of known attack vectors and common anti-patterns. https://github.com/sigp/solidity-security-blog, Nov 2019.

[118] MarketsandMarkets. Blockchain market worth $39.7 billion by 2025. *PR Newswire*, 2020. [Online; accessed 16-Oct-2024].

[119] C. Mitchell. Front-running definition, example, and legality. https://www.investopedia.com/terms/f/frontrunning.asp, May 2022. [Online; accessed 28-May-2023].

[120] T. of Bits. Slither: Solidity static analysis framework. https://github.com/crytic/slither, 2018. [Online; accessed 08-Jun-2024].

[121] T. of Bits. Watch your language: Our first vyper audit. https://blog.trailofbits.com/2019/10/24/watch-your-language-our-first-vyper-audit/, Oct 2019.

[122] T. of Bits. Slither, the solidity source analyzer. https://github.com/crytic/slither, Jan. 2023. [Online; accessed 08-Jun-2023].

[123] OpenZeppelin. openzeppelin-solidity. https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/token/ERC20/ERC20.sol, Dec. 2018. [Online; accessed 23-Dec-2018].

[124] OpenZeppelin. Erc-20 token standard implementation, 2020. Accessed: 2024-02-11.

[125] OpenZeppelin. Safemath library for solidity, 2020. Accessed: 2024-02-11.

[126] Opyn. Historical daily premium. https://squeeth.opyn.co/squeeth, Sept. 2024. [Online; accessed 29-Sep-2024].

[127] S. Palladino. The parity wallet hack explained. https://blog.openzeppelin.com/on-the-parity-wallet-multisig-hack-405a8c12e8f7/, July 2017. [Online; accessed 16-Oct-2024].

[128] PeckShield. Alert: New batchoverflow bug in multiple erc20 smart contracts. https://peckshield.medium.com/alert-new-batchoverflow-bug-in-multiple-erc20-smart-contracts-cve-2018-10299-511067db6536, Apr 2018. [Online; accessed 16-Oct-2024].

[129] PeckShield. Alert: New batchoverflow bug in multiple erc20 smart contracts, 2018. Accessed: 2024-02-11.

[130] D. Perez and B. Livshits. Smart contract vulnerabilities: Vulnerable does not imply exploited. https://www.usenix.org/system/files/sec21summer_perez.pdf, Aug 2018.

[131] Pionex. Leveraged tokens. https://www.pionex.com/blog/what-arepionex-leveraged-tokens/, May 2023. [Online; accessed 03-May-2023].

[132] Pionex. Pionex will terminate futures manual trading since 2021/6/30. https://pionex.zendesk.com/hc/en-us/articles/900007639683-Pionex-will-terminate-Futures-manual-trading-since-2021-6-30, May 2023. [Online; accessed 15-May-2023].

[133] Pionex. What are pionex leveraged tokens. https://www.pionex.com/blog/what-arepionex-leveraged-tokens/, May 2023. [Online; accessed 07-May-2023].

[134] S. Pittman. Volatility drag: What is it? what is it doing to my wealth? and how do i curb it? https://seekingalpha.com/instablog/659612-sampittman/73533-volatility-drag-what-is-it-what-s-it-doing-to-my-wealth-and-how-do-i-curb-it, May 2010. [Online; accessed 19-May-2023].

[135] Poloniex. What is a funding rate? https://medium.com/poloniex/what-is-a-funding-rate-257e4cbeefe0, Nov 2022. [Online; accessed 20-May-2023].

[136] Poloniex. Ftx leveraged tokens faq. https://support.poloniex.com/hc/en-us/articles/360045644653-FTX-Leveraged-Tokens-FAQ, May 2023. [Online; accessed 07-May-2023].

[137] W. Prospere. Squeeth primer: a guide to understanding opyn's implementation of squeeth. https://medium.com/opyn/squeeth-primer-a-guide-to-understanding-opyns-implementation-of-squeeth-a0f5e8b95684, Jan. 2022. [Online; accessed 13-Sep-2024].

[138] A. Punia. Price alert: Eth2x-fli premium due to supply shortage. https://indexcoop.com/blog/eth2x-fli-price-alert-240521, May 2021. [Online; accessed 29-Aug-2024].

[139] J. Quintal. Building robust smart contracts with openzeppelin. https://www.trufflesuite.com/tutorials/robust-smart-contracts-with-openzeppelin, Aug 2017.

[140] H. Qureshi. A hacker stole $31m of ether — how it happened, and what it means. https://www.freecodecamp.org/news/a-hacker-stole-31m-of-ether-how-it-happened-and-what-it-means-for-ethereum-9e5dc29e33ce/, Jul 2017. [Online; accessed 16-Oct-2024].

[141] R. Rahimian and J. Clark. Tokenhook: Secure erc-20 smart contract. *arXiv preprint arXiv:2107.02997*, 2021. [Online; accessed 16-Oct-2024].

[142] R. Rahimian and J. Clark. A Shortfall in Investor Expectations of Leveraged Tokens. In R. Böhme and L. Kiffer, editors, *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:24, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[143] R. Rahimian, S. Eskandari, and J. Clark. Resolving the multiple withdrawal attack on erc20 tokens. https://arxiv.org/abs/1907.00903, Jul 2019. [Online; accessed 16-Oct-2024].

[144] R. Rahimian, S. Eskandari, and J. Clark. Resolving the multiple withdrawal attack on erc20 tokens. *arXiv*, 1907.00903, 2019.

[145] A. Reporting. Generate meaningful knowledge from ethereum. https://reports.aleth.io/, Jul 2020. [Online; accessed 25-Feb-2020].

[146] Reportlinker. Blockchain technology market worldwide is projected to grow by us $14.3 billion. https://www.globenewswire.com/news-release/2020/05/20/2036593/0/en/Blockchain-Technology-market-worldwide-is-projected-to-grow-by-US-14-3-Billion.html, May 2020. [Online; accessed 16-Oct-2024].

[147] B. Research. Key trends in crypto – october 2024. https://www.binance.com/en/blog/crypto/key-trends-in-crypto-october-2024, 2024. [Online; accessed 19-Oct-2024].

[148] A. Roan. 7 openzeppelin contracts you should always use. `https://medium.com/better-programming/7-openzeppelin-contracts-you-should-always-use-5ba2e7953cc4`, Apr 2020.

[149] G. Rompotis. Return and volatility of emerging markets leveraged etfs. *Journal of Asset Management*, 17:165–194, 2016.

[150] U. Securities and E. Commission. Leveraged and inverse etfs: Specialized products with extra risks for buy-and-hold investors. `https://www.sec.gov/investor/pubs/leveragedetfs-alert`, Feb 2023. [Online; accessed 02-May-2023].

[151] U. Securities and E. Commission. Updated investor bulletin: Leveraged and inverse etfs. `https://www.sec.gov/resources-for-investors/investor-alerts-bulletins/updated-investor-bulletin-leveraged-inverse-etfs`, Aug. 2023. [Online; accessed 17-Aug-2024].

[152] K. Sedgwick. Myetherwallet servers are hijacked. `https://news.bitcoin.com/myetherwallet-servers-are-hijacked-in-dns-attack/`, Apr 2018. [Online; accessed 16-Oct-2024].

[153] B. Seyhan and E. Sefer. Nft primary sale price and secondary sale prediction via deep learning. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, 2023.

[154] M. Shirole, M. Darisi, and S. Bhirud. Cryptocurrency token: An overview. In *IC-BCT 2019: Proceedings of the International Conference on Blockchain Technology*, pages 133–140. Springer, 2020.

[155] P. Shum, W. Hejazi, E. Haryanto, and A. Rodier. Intraday share price volatility and leveraged etf rebalancing. *Review of Finance*, 20(6):2379–2409, 2016.

[156] P. Shum, W. Hejazi, E. Haryanto, and A. Rodier. Intraday share price volatility and leveraged etf rebalancing. *Review of Finance*, 20(6):2379–2409, 2016.

[157] D. Siegel. Understanding the dao attack. `https://www.coindesk.com/understanding-dao-hack-journalists`, Jun 2016. [Online; accessed 16-Oct-2024].

[158] SmartDec. Knowledge-base. `https://tool.smartdec.net/knowledge`, Sep 2018.

[159] J. Songthammakul. Alpha incubate introduces contango, the first dex to bring expirable futures to defi. `https://blog.alphaventuredao.io/alpha-incubate-introduces-contango-the-first-dex-to-bring-expirable-futures-to-defi`, Feb. 2023. [Online; accessed 27-Sep-2024].

[160] Sooho. Verify a smart contract. `https://odin.sooho.io/`, Mar 2020.

[161] A. Sristy. Blockchain in the food supply chain - what does the future look like? `https://tech.walmart.com/content/walmart-global-tech/en_us/blog/post/blockchain-in-the-food-supply-chain.html`, 2021. [Online; accessed 18-Oct-2024].

[162] C. STAMFORD. Gartner predicts 90% of current enterprise blockchain platform implementations will require replacement by 2021. https://www.gartner.com/en/newsroom/press-releases/2019-07-03-gartner-predicts-90--of-current-enterprise-blockchain, Jun 2019. [Online; accessed 16-Oct-2024].

[163] Statista. Market share of largest providers of exchange traded funds (etfs) in the united states as of september 2022. https://www.statista.com/statistics/294411/market-share-etf-providers-in-the-us/, May 2023. [Online; accessed 13-May-2023].

[164] R. Sullivan. The pitfalls of leveraged and inverse etfs. *CFA Institute Magazine*, 20:12, 2009.

[165] F. R. Systems. Leveraged etf list. https://etfdb.com/etfs/leveraged/, Apr 2024. [Online; accessed 20-Apri-2024].

[166] L. Szpruch, J. Xu, M. Sabate-Vidales, and K. Aouane. Leveraged trading via lending platforms. *Available at SSRN*, 2024.

[167] F. Times. Investors pump record sums into leveraged etfs. https://www.usa.gov/agencies/securities-and-exchange-commission, Nov 2022. [Online; accessed 19-Feb-2024].

[168] T. Tracker. Erc-20 tokens. https://etherscan.io/tokens, Jul 2020. [Online; accessed 16-Oct-2024].

[169] W. Trainor. Do leveraged etfs increase volatility? *Technology and Investment*, 1:215–220, 2010.

[170] W. Trainor Jr. Daily vs. monthly rebalanced leveraged funds. *Journal of Finance and Accountancy*, 6:1, 2011.

[171] G. Tsalikis and S. Papadopoulos. Can shorting leveraged exchange-traded fund pairs be a profitable trade? *Journal of Investment Strategies*, 2019.

[172] P. Tsankov. Securify v2.0. https://github.com/eth-sri/securify2, Jan 2020.

[173] P. Tsankov, A. Dan, D. Drachsler-Cohen, and A. Gervais. Securify: Practical security analysis of smart contracts. https://arxiv.org/pdf/1806.01143.pdf, Aug 2018.

[174] P. Vessenes. Monolithdao token contract. GitHub repository, 2017. Accessed: 2024-02-11.

[175] F. Victor and B. K. Lüders. Measuring ethereum-based erc20 token networks. In *International Conference on Financial Cryptography and Data Security*, 2019.

[176] M. Vladimirov. Attack vector on ERC20 API (approve/transferFrom methods) and suggested improvements. https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729, Nov. 2016. [Online; accessed 18-Dec-2018].

[177] Vyper. Pythonic smart contract language for the evm. https://github.com/vyperlang/vyper, Apr 2021.

[178] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. Knottenbelt. Sok: Decentralized finance (defi). *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, page 30–46, 2022.

[179] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. Knottenbelt. Sok: Decentralized finance (defi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 30–46, 2022.

[180] K. Wided. On the properties of leveraged etfs. *idea*, 2:2, 2019.

[181] K. Wided. On the properties of leveraged etfs. *IDEA*, 2(2), 2019.

[182] Wikipedia. Incremental research. `https://en.wikipedia.org/wiki/Incremental_research`. [Online; accessed 02-Oct-2023].

[183] Wikipedia. Compare-and-swap. `https://en.wikipedia.org/wiki/Compare-and-swap`, July 2018. [Online; accessed 10-Dec-2018].

[184] wikipedia. Mutual exclusion. `https://en.wikipedia.org/wiki/Mutual_exclusion`, Jan 2019.

[185] Wikipedia. Liberty reserve. `https://en.wikipedia.org/wiki/Liberty_Reserve`, Jun 2020.

[186] G. Wood. Ethereum: a secure decentralised generalised transaction ledger. `https://ethereum.github.io/yellowpaper/paper.pdf`, Mar 2016. [Online; accessed 16-Oct-2024].

[187] yAudit. Tlx review. `https://tlx.fi/tlx-audit.pdf`, Mar. 2024. [Online; accessed 29-Sep-2024].

[188] M. Zalecki. Create and distribute your erc20 token with openzeppelin. `https://www.tooploox.com/blog/create-and-distribute-your-erc20-token-with-openzeppelin`, Aug 2018.

[189] L. Zhang, Y. Chen, and Z. Xu. A survey of smart contract vulnerabilities and their mitigation in ethereum. In *Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 114–121. IEEE, 2020.

[190] J. Zhou, R. Johnson, and V. Vasudevan. High-frequency trading on decentralized on-chain exchanges. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 517–530, 2020.

[191] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais. High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445. IEEE, 2021.