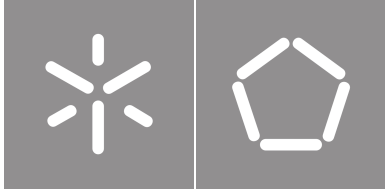
Universidade do Minho
Escola de Engenharia

Vítor Hugo da Silva Ribeiro

Towards Secure Layer-2 Blockchain Solution using TEEs

Dezembro, 2022



Universidade do Minho
Escola de Engenharia

Vítor Hugo da Silva Ribeiro

Towards Secure Layer-2 Blockchain Solution using TEEs

Dissertação de Mestrado
Mestrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação de:
Doutor Sandro Pinto

Dezembro, 2022

Agradecimentos

Depois de um ano desafiante, com muitos altos e baixos, mas com muito esforço e dedicação, chegou finalmente o momento de fazer os devidos agradecimentos a todos aqueles que tiveram um papel importante nesta jornada.

Começo por agradecer aos grandes impulsionadores deste trabalho, ao David Cerdeira por todo o acompanhamento e ajuda constante e ao Prof. Dr. Sandro Pinto pelo suporte e orientação e sobretudo pela amizade e confiança depositada em mim para realizar este trabalho numa área tão disruptiva e ainda pouco explorada.

Tenho a agradecer também aos meus companheiros de curso, Diogo Matias, Francisco Dias, Gonçalo Freitas e Heitor Silva, pelo companheirismo e momentos partilhados ao longo destes 5 anos que ficarão marcados para sempre.

Aos meus amigos do RENT: Quim, Fraga, Tala, Tinga, Tó, Varela, Edinho, David, FF, CC, Zé, Gingla, Vieira, Serginho e Relenha, pelos momentos de alegria e descontração, principalmente nas fases de maior dúvida e frustração.

Um agradecimento especial aos meus pais e irmã, pelo apoio e conforto e por me proporcionarem todas as condições necessárias para que me dedicasse sempre a 100% não só a esta dissertação, mas também a todo o curso, sem limitações e complicações. Não há palavras para vocês!

Por fim, agradecer à minha namorada, Ana Ribeiro, o meu pilar fundamental. Por me apoiar sempre, nos bons e nos maus momentos, agradecer por todo o amor e por toda a compreensão, sem dúvida que foi essencial para tornar todo este percurso mais fácil. Um obrigado é pouco!

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Resumo

Segurança para Layer-2 de Blockchain utilizando TEEs

As tecnologias *Web3* têm sido alvo de especial atenção no contexto académico e empresarial. A *blockchain* é uma das tecnologias mais disruptiva utilizada em *Web3* e tem sido amplamente adotada. Uma *blockchain* pode ser descrita como uma base de dados ou um *ledger* que permite o armazenamento de dados de forma descentralizada e distribuída. Esta tecnologia tem vantagens em termos de segurança, imutabilidade e privacidade de dados, tornando-a assim adequada para uma variedade de casos de uso. O conceito *blockchain* foi inicialmente introduzido pela moeda criptográfica Bitcoin, mas hoje em dia é utilizado em diferentes áreas.

Após o aparecimento de Bitcoin, outro marco importante na tecnologia foi o aparecimento de Ethereum que abriu um enorme leque de oportunidades na área. Ethereum é uma plataforma descentralizada que permite a criação de *smart contracts* e aplicações descentralizadas alavancadas pela tecnologia *blockchain*. Com isto, surgiram muitas aplicações descentralizadas para os mais variados casos de uso.

Apesar da constante e contínua adoção de *blockchain* e principalmente de Ethereum, existem ainda inúmeros desafios. O trilema da escalabilidade sugere que das três principais propriedades da *blockchain*, isto é, escalabilidade, descentralização e segurança, não é possível escolher duas sem colocar a restante em causa. De forma a colmatar este problema, têm surgido várias soluções, como *layer-1* e *layer-2*. Soluções de *layer-2* movem as computações complexas da *blockchain* para fora da *chain*, de forma a aliviar o processamento na *blockchain*. O excesso de computações na *blockchain* tem um custo significativo, por isso, movendo-as para *off-chain*, as soluções de *layer-2* aumentam a escalabilidade das aplicações. Tendo em conta que os cálculos são realizados fora da *blockchain*, muitas das garantias de segurança são perdidas, uma vez que a componente *off-chain* pode ser comprometida.

Esta dissertação apresenta a análise de segurança de uma solução *layer-2* de *blockchain* chamada Cartesi. Este trabalho mostra que a componente *off-chain* de Cartesi é vulnerável não só por falhas associadas à arquitetura, mas também devido a ameaças que podem surgir por parte de um *host* comprometido. Foi montado um ataque a uma aplicação de Cartesi que explora as vulnerabilidades evidenciadas, pondo em causa a segurança do sistema e dos seus utilizadores. De forma a mitigar estas vulnerabilidades, propomos e discutimos como aproveitar os TEEs, uma tecnologia bem estabelecida em aplicações móveis e na *cloud*, para fornecer garantias de segurança adicionais a Cartesi.

Palavras-chave: Web3, Blockchain, Layer-2, Smart Contracts, DApp, TEE, Cartesi, Segurança, Confidential Computing, Off-chain, Criptografia

Abstract

Towards Secure Layer-2 Blockchain Solution using TEEs

Web3 technologies have attracted huge attention from both academia and industry. Blockchain is one of the most disruptive technologies used in Web3 and has been widely adopted. A blockchain can be described as a database or ledger that allows data to be stored in a decentralized and distributed way. This technology has advantages in terms of security, immutability, and data privacy, thus making it fitted for a variety of use cases. The concept of blockchain was initially introduced by the Bitcoin cryptocurrency, but nowadays, it is used in wide spectrum of domains and applications.

Following the rise of Bitcoin, another major milestone in technology was the emergence of Ethereum, which unlocked numberless opportunities in the field. Ethereum is a decentralized platform that allows the creation of smart contracts and decentralized applications leveraging blockchain technology. With this, many decentralized applications have emerged for a variety of use cases.

Despite the constant and continued adoption of blockchain and especially Ethereum, there are still numerous challenges. The scalability trilemma suggests that of the three main properties of blockchain, i.e., scalability, decentralization, and security, it is not possible to prioritize one without jeopardizing others. To address this problem, various solutions have emerged, such as layer-1 and layer-2. Layer-2 solutions move the complex computations of the blockchain to off-chain, to alleviate the overload from the blockchain. The excess of computations in the blockchain has a significant cost and therefore, by moving them off-chain, layer-2 solutions increase the scalability of applications. Given that the computations are performed outside the blockchain, many of the security guarantees are lost as the off-chain side can be compromised.

This thesis presents the security analysis of a layer-2 blockchain solution called Cartesi. This work shows that the *off-chain* component of Cartesi is vulnerable not only due to flaws associated with the architecture, but also due to threats that can arise from a compromised *host*. An attack was mounted on a Cartesi application that exploits the disclosed vulnerabilities, jeopardizing the security of the system and its users. In order to mitigate these vulnerabilities, we propose and discuss how to leverage TEE, a well-established technology in mobile and the cloud applications, to provide additional security guarantees to Cartesi.

Keywords: Web3, Blockchain, Layer-2, Smart Contracts, DApp, TEE, Cartesi, Security, Confidential Computing, Off-chain, Cryptography

Contents

List of Figures

List of Tables

List of Listings

Acronyms

API Application Programming Interface 15, 18, 22, 28

CLINT Core Local Interruptor 24

DAO Decentralized Autonomous Organization 5, 12
DApp Decentralized Application xi, 1, 2, 9, 10, 11, 13, 15, 19, 20, 21, 22, 23, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 48, 50, 52, 53, 54, 56
DeFi Decentralized Finance 5
DoS Denial of Service 12, 39
DPoS Delegated-Proof-of-Stake 6, 20
DSL Domain-specific Language 10

EVM Ethereum Virtual Machine 39, 52

GPL General-purpose Language 10

HTIF Host-Target Interface 24

I/O Input/Output 24
IoT Internet of Things 1, 5, 16, 18
ISA Instruction Set Architecture 15, 23

NFT Non-fungible Token 5, 10

OS Operating System 2, 13, 14, 15, 17, 21, 22, 23, 24, 40, 45, 46, 48

P2P Peer-to-Peer 10
PoS Proof-of-Stake 5, 6, 19

PoW Proof-of-Work 5, 6, 19

SDK Software Development Kit 22, 29, 39
SGX Intel's Software Guard Extensions 15, 16, 17, 18, 19, 20, 47, 48

TA Trusted Application 15
TEE Trusted Execution Environment 2, 3, 4, 15, 16, 17, 18, 19, 20, 45, 46, 47, 48, 53, 55, 56, 57
TPS transactions per second 6

UI User Interface 11, 27

VM Virtual Machine 14, 16, 17, 24

ZK Zero-Knowledge 8, 9

Chapter 1

Introduction

This chapter contextualizes the present work in the scope of security of a layer-2 blockchain solution, defines the objectives, and presents the structure of the thesis.

1.1 Contextualization

Web3 technologies have attracted huge attention from both academia and industry. Blockchain is the main web3 pipeline and is a technology that has been widely adopted. It works as an immutable decentralized database, with several advantages in terms of tracking and ownership. This technology was first widely adopted in cryptocurrency, thanks to Bitcoin [1], but its utility has now expanded to various areas. Areas such as Internet of Things (IoT) [2], real estate and smart cities [3], healthcare [4], energy and sustainability [5], and several others [6], already use blockchain technology to manage and store their data.

Followed by the advent of Bitcoin, another major milestone in technology was the emergence of Ethereum [7] that allowed several other opportunities in the industry [8–10]. Ethereum is a decentralized platform that allows the creation of smart contracts and DApps leveraged by the blockchain. Smart contracts have enabled the development of DApps that are similar to traditional applications, but run on a blockchain and gather all its advantages. Several DApps have emerged with the most diverse use cases, considering the advantages they offer not only for the developers but also for the users.

However, due to it being new technology, it faces some challenges that are heavily linked to widespread adoption. The three key properties of blockchain are scalability, decentralization, and security. According to the scalability trilemma [11], it is not yet possible to increase two of these properties without jeopardizing the third. Therefore, various protocols have emerged to try to address this challenge, such as layer-1 [12, 13] and layer-2 [14, 15] solutions. Layer-1 solutions are upgrades of the main architecture of the

blockchain itself in order to optimize it. The purpose of layer-2 solutions is to move computations from on-chain to off-chain in order to increase the scalability of the blockchain by removing the overload, which leads to high operation costs. Therefore, several layer-2 proposals have emerged that implement different ways of doing this offload, processing the computations and returning the results to the blockchain. Although security is one of the great properties of the blockchain, this offload leads to losses of the blockchain security guarantees, since the off-chain side can be compromised.

Therefore, this thesis aims to conduct a security analysis of a layer-2 solution, called Cartesi [16], in order to identify flaws in the architecture by which an attacker could attack and jeopardize an entire DApp. Cartesi is a layer-2 platform for development and deployment of DApps. The goal of the founders is to allow DApps to be developed using conventional programming languages by offering a Linux Operating System (OS) coupled with a blockchain infrastructure. These advantages reduce the entry barrier for developing DApps and free the developers from the limitations and specificities imposed by each blockchain. Being a layer-2 solution, the scalability improvement is based on running the off-chain computations. In Cartesi these computations are done inside a virtual machine, the so-called Cartesi Machine, which leverages a Linux OS. Although the authors assure that off-chain there are the same security guarantees as on the blockchain, and assuming that blockchain is completely secure, being off-chain it is likely to always be subject to vulnerabilities arising from external sources. In this work, we perform a security analysis and investigate and categorize threats according to their impact on the security of the system. With the vulnerabilities identified, we will mount a proof of concept attack on a Cartesi application and propose mitigation solutions based on the use of Trusted Execution Environments (TEEs).

1.2 Goals

This thesis aims to provide a security analysis of a layer-2 blockchain protocol. To conduct this analysis, we will have to fulfill some goals to reach a potential solution to the flaws found in this platform. The first goal is to perform an in-depth security analysis of the Cartesi layer-2 solution in order to identify vulnerabilities and threats to the system. In order to successfully perform the security analysis, it will also be necessary to perform a detailed analysis of the Cartesi protocol, be able to run it and perform some actions to understand every aspect of every subsystem involved. This includes understanding the off- and on-chain side of the system, as well as the communication between them.

After the study is completed, the next goal will be to create a threat model with all the potential sources of security threat in the system. To complement this, it will be necessary to do a conceptual study of how to categorize and address the problems identified in the threat model. Threats must be ranked since not all threats have the same level of severity, they depend on the impact that potential attacks can cause on a given component and thus jeopardize the entire system's security.

The final goal will be to develop threat mitigation solutions, based on leveraging TEEs, according to the impact of the threats found. Once again, the techniques used for mitigation will take into consideration

their impact on the security of the system. This solution should be supported by a proof-of-concept that effectively shows the results of TEE application in the off-chain part of a layer-2 blockchain solution.

1.3 Responsible Disclosure

We responsibly disclose all our work and findings to the respective Cartesi team. We provide information about the vulnerabilities found in the architecture and also about the attack performed. We also provide detailed solution proposals to mitigate the mentioned threats. We hope to contribute and work together towards a solution that adds value not only to the Cartesi product, but to the entire web3 area by introducing the use of TEEs in the context of layer-2 solutions.

1.4 Structure

The structure of this document is as follows. Chapter 2 presents the theoretical background of the concepts covered in the thesis, namely concepts associated with blockchain and layer-2 and also those related to security and TEEs. In addition to the theoretical concepts, this chapter also presents related work. Chapter 3 presents an in-depth analysis of the layer-2 solution protocol, Cartesi, studied in this thesis. We introduce an overview of the protocol, focusing more on the first version of the platform, Descartes. We also provide a detailed analysis of each of the on- and off-chain components of Descartes, mainly of its core technology, the Cartesi Machine. This chapter covers the entire architecture of the protocol, as well as the interaction between the various components in the timeline of an application's execution. Chapter 4 includes the security analysis of the layer-2 protocol studied in the previous chapter. This analysis encompasses all the flaws and vulnerabilities found in the system, in order to create the threat model and properly categorize the threats. Besides the description of the threat model, this chapter presents an attack conducted on an application that uses the protocol in order to exploit and prove the vulnerabilities mentioned above. Chapter 5 describes several mitigation proposals for the threats found above. These proposals are based on the use of TEEs to increase the security of the protocol. This chapter includes an analysis of several solutions using different TEE technologies that are easily adapted to the platform under study. Finally, we conclude with an implemented solution, supported by a proof-of-concept that solves several of the protocol's problems, mainly the one exploited in the attack scenario. This thesis ends with chapter 6 which presents the conclusions drawn from the work done, concluding with several proposals for future work that emerged from the analyses carried out, with a view to expanding the work developed.

Chapter 2

Background and Related Work

This chapter presents a theoretical background of the main concepts covered in this thesis. Given that the objective of this thesis is to perform the security analysis of a layer-2 blockchain solution based on the use of TEEs for threat mitigation, it is essential to clarify the fundamental concepts. The project under study is built on blockchain technology, thus, we provide an overview of the technology and its use cases. In addition, it is necessary to understand the concepts of layer-1 and layer-2, since they are directly correlated with the case study. In this chapter we present an overview of the protocol under analysis, Cartesi, and also briefly compare it to its competitors. Finally, we will provide an overview of the concept of confidential computing and TEEs, these being the basis of the mitigation solution that this thesis proposes.

2.1 Blockchain

A blockchain is a distributed ledger for recording transactions or assets that is cryptographically secure, append-only and transparent. It can be seen as a chain of blocks that store information in a decentralized and distributed network. This chain grows when new information is stored, with new blocks being generated and appended. Blocks record several types of information, such as their origin, their creator, and the timestamp of their creation. The update or addition of the blocks on the chain can only occur by agreement among network participants, a concept known as consensus [17] which will be further explained below. Blockchain is built upon digital signatures, cryptographic hashes, and consensus algorithms [18]. These principles provide the technology with key characteristics such as security, anonymity, and immutability.

This technology allows the transfer of digital assets in a peer-to-peer way, thus requiring no intermediaries. This concept was introduced by Bitcoin and now blockchain is the underlying and key technology of digital cryptocurrencies. A major innovation in the world of cryptocurrencies, after Bitcoin, was Ethereum.

Ethereum has become a foundation for a large number of new cryptocurrencies and applications that were previously not possible. New concepts have emerged mainly in the finance domain such as Decentralized Finance (DeFi) [19], Decentralized Autonomous Organizations (DAOs) [20], Non-fungible Tokens (NFTs) [21], stable coins [22], and many others [23, 24].

Besides the implementation in cryptocurrency, blockchain characteristics make the technology well fitted for a variety of applications, including financial services [25], Internet of Things (IoT), healthcare, supply chain [26], among others [27]. In the finance industry, blockchain technology allows the transfer of assets with confidence that the transaction is secure and reliable. In the supply chain [28], for example, the consumers lack information about the origin of the products. By introducing blockchain in this field, the data is decentralized from the intermediaries (wholesalers) and each customer can transparently access all important data. This removal of control of data from any authority and the ability to overwrite existing data records is also suitable for the IoT industry, adding security to systems. Areas that require data records will benefit from using blockchain as the underlying technology, as is also the case in healthcare. In general, the use of this technology can help solve several problems in a variety of use cases and traditional applications.

One key aspect of blockchain is that it is a decentralized system, thus there is no need for a third-party trusted authority. Instead, to guarantee the security, availability, and consistency of the data and transactions, blockchain adopts the consensus mechanisms. Consensus ensures that every new block that is added to the blockchain is the one and only version of the truth that is accepted upon by all the nodes in the blockchain. There are different types of consensus mechanism algorithms [29], each of which operates on a different set of principles. Among the various consensus methods, the most important are: Proof-of-Work (PoW) [30] and Proof-of-Stake (PoS) [31].

PoW is based on the network user's capacity to prove that a computational task is accomplished. This mechanism requires computational power to solve complex calculations in order to validate a new block in the chain, a process known as mining. The devices that perform this process are called miners and receive a reward for each validated block, which encourages them to keep a honest participation in the network. The PoW mechanism is computationally expensive as it involves hash-based encryption and digital signatures. Despite the disadvantage of the computational effort inherent to cryptography, energy consumption is the price to pay for network security. PoS is an approach to address the inefficiencies of PoW, mainly to reduce its dependency on energy consumption originated by complex computations. PoS is based on participation in the network, known as staking. In cryptocurrency terms, in the PoS mechanism, participation in validation is determined by the ownership of a coin or a cryptocurrency and not by mining. A node, which stakes a certain amount of the blockchain native cryptocurrency, is elected, and its job is to check the authenticity of the transactions in the block, sign it and submit it to the network for validation. As in PoW, the validator node, in this case named staker, is also rewarded with a percentage of its stake.

There is an ever-growing number of blockchain applications, and naturally, the networks leveraging this technology are constantly growing, thus increasing the pressure on the requirements for the underlying

infrastructure. Since the consensus mechanisms currently deployed are mainly based on redundancy, increasing the number of transactions creates significant scalability challenges [32].

2.1.1 Layer-1

According to the blockchain scalability trilemma, there are three properties, i.e., scalability, decentralization, and security. Scalability refers to the increase in the utilization, with the blockchain being able to process more transactions per second (TPS). Decentralization means the network runs without intermediaries and without dependence on centralized actors. Finally, security is the protection of data and the network against various types of attacks that will be mentioned later. However, blockchain scalability is a quandary, because each of these properties pushes in different directions.

Scaling a blockchain ecosystem can be done through two ways: layer-1 and layer-2 scaling. Layer-1 and layer-2 solutions emerge as possible solutions to the scalability problem. Layer-1 solutions provide protocol variations and optimizations to the operation of the blockchain itself or for the "main chain", mainly in terms of greater transaction capacity and lower operating costs. The blockchain requires each authenticating computer or node, to record all the data in the chain because it is part of the consensus process, but as more transactions occur, more computations are required. Additionally, having too many blocks to process causes congestion in the network. Since transactions are executed sequentially, executing them in parallel [33] may be one of the solutions. One solution implementing this idea is *Sharding* [1], which splits the network into multiple sets of blocks (shards) and then this approach handles the validation responsibility to only a random set of validators. This way, when a validator verifies a shard, it publishes the digital signature that proves it, and the other validators in the network, instead of analyzing all the blocks in the network, only verify the signatures generated, greatly reducing the computational effort.

There are other solutions to solve the problems associated with the scalability trilemma, such as PoS or Delegated-Proof-of-Stake (DPoS) algorithms [34]. Even though they do not require time and energy-consuming like PoW, layer-1 solutions are still not enough for high adoption scenarios, considering that Bitcoin, for example, can only perform a maximum of 7 TPS, as compared to a traditional system like Visa that can handle around 20000 TPS [35]. Due to the shortcomings of layer-1 solutions, other approaches have been explored, including layer-2 solutions.

2.1.2 Layer-2

Layer-2 is a secondary protocol built on top of an existing blockchain, where transactions can take place independently of layer-1 (main chain), i.e., off-chain. Instead of modifying the main chain to improve scalability, the objective is to offload some processing outside the main underlying chain, perform processing there and retrieving it back to the blockchain as an integrity guarantee. This approach reduces the load on the blockchain itself, allowing for increased throughput while trying to ensure the same level

[1] Why sharding is great: demystifying the technical properties - https://vitalik.ca/general/2021/04/07/sharding.html

of network security, since the computation takes place off-chain and the main chain only receives the result. All off-chain computation is abstracted and detached from the whole complex process of on-chain transactions.

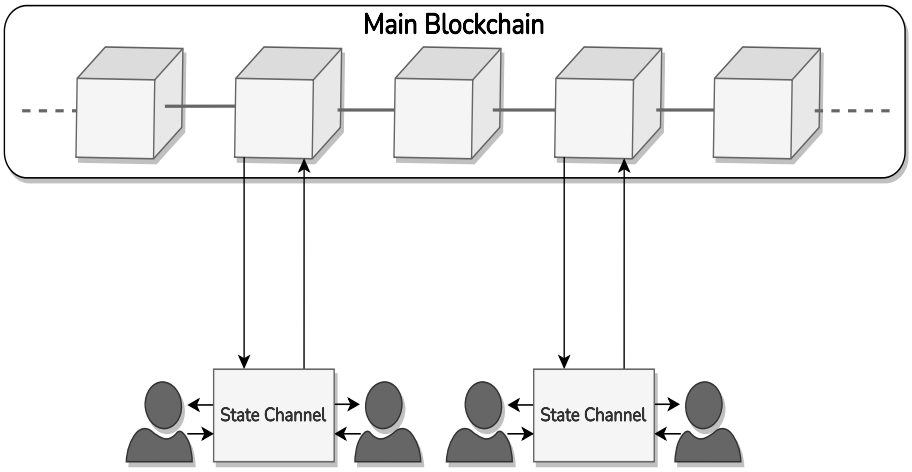


Figure 1: State Channels.

There are several types of layer-2 solutions, among which, the most important are State channels [36], Sidechains [37], and Rollups. State channels, shown in Figure 1, are two-way communication channels between nodes that occur off-chain. These channels are open, the whole process takes place between two interested participants and only the final result is sent to the main chain. At the end of the process, the channel is closed. Lightning Network [38] is an example of this implementation, allowing Bitcoin to improve latency and throughput as the blockchain is not involved in every transaction and the number of transactions that can be processed is not limited by it.

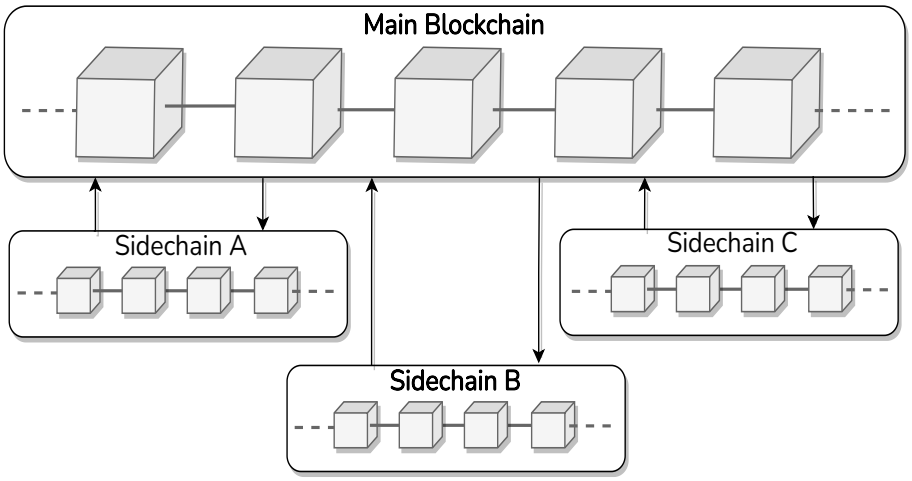


Figure 2: Sidechains.

Sidechains are small independent blockchains, as depicted in Figure 2, that employ their own consensus mechanism and work in parallel to the main chain, as happens with Plasma sidechains [39] on

the Ethereum blockchain. Transactions are moved to the sidechains, processed, and then, in the end, a confirmation is communicated across the chains.

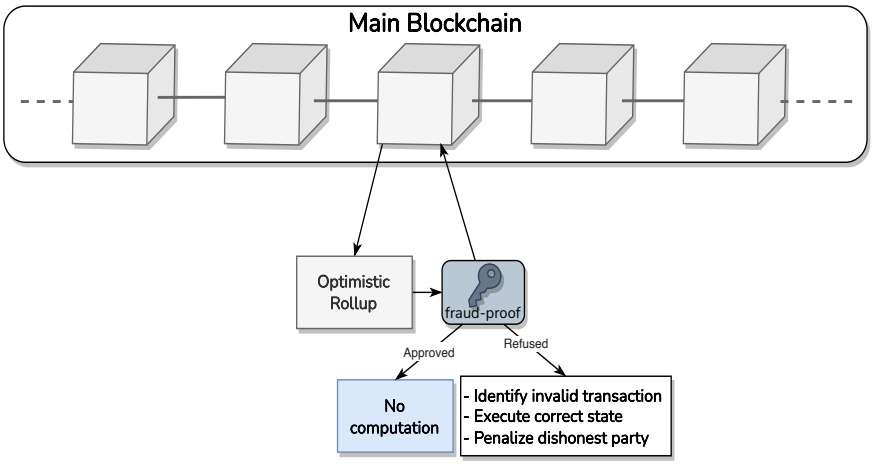


Figure 3: Optimistic Rollups.

Finally, rollups are solutions that move computations off-chain while keeping data on-chain. In a rollup, transaction data is higly-compressed to save on blockchain data transfer costs. The state of each rollup contains this compressed data and the root of the Merkle representing the previous state. After the rollup is accepted, the root of the Merkle tree is updated. In order to verify the truthfulness of the state roots, rollups can be divided into two flavors: Optimistic [40] and Zero-Knowledge (ZK) [41]. Optimistic rollups always assume that transactions are valid by default, approving them without any computation, as depicted in Figure 3. It only performs the calculation if, after the period nodes have to contest, some validator detects that the transaction data was fraudulent, using a fraud-proof security method. By validating all transactions, it accelerates the speed of transactions and also of the network, being only delayed by the time of contestation.

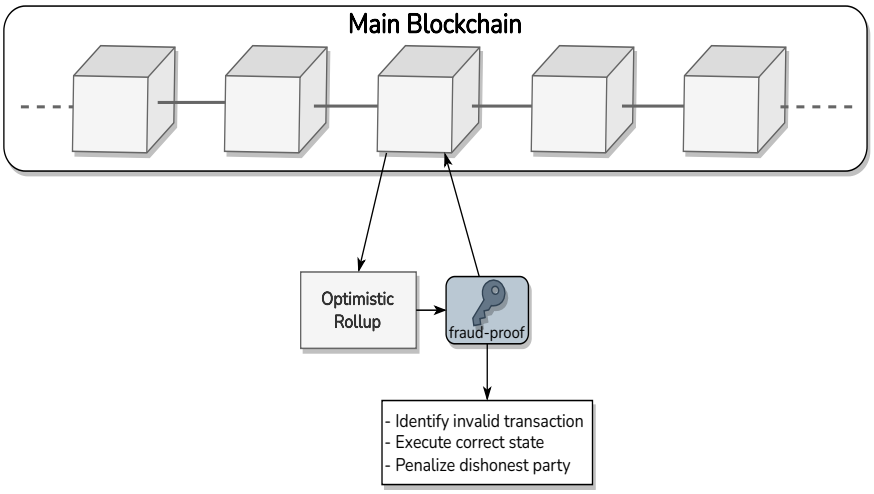


Figure 4: Zero-knowledge Rollups.

In contrast, as can be observed in Figure 4, ZK Rollups, performs the calculation normally, only submitting a cryptographic key that proves to the main chain the validity of the transaction. Although it is a bit more time-consuming when calculating validity proofs, it significantly decreases the amount of data. On the Ethereum blockchain, the implementation of rollups can scale up to about 30 times more transactions [42].

2.1.3 Smart Contracts and DApps

Ethereum's blockchain has enabled the introduction of two new concepts in the field: smart contracts and Decentralized Applications (DApps). Smart contracts are programs or transactions protocols that run on the blockchain, and DApps are decentralized blockchain-based applications that allow users to interact with smart contracts. The development of infrastructure to support smart contracts and DApps is an ongoing intense area of research.

Smart Contracts

The first smart contract was implemented in the 1990s [43] before the invention of Bitcoin and the upcoming blockchain platforms, such as Ethereum. Initially, smart contracts were defined as electronic transaction protocols that execute the terms of a paper contract, to satisfy contractual terms and conditions, minimizing fraud or accidents, without the need for intermediaries. With Bitcoin, smart contracts started to be used again [44], although it is not referred to as such, mainly to transfer Bitcoins between users.

Nowadays, smart contracts are considered to be secure programs, running on the blockchain, representing an agreement that is automatically executable and enforceable. Leveraging Solidity [2], a Turing-complete and JavaScript-like programming language, developers are able to deploy a set of smart contracts that will be written in blocks of the blockchain. In Solidity, there are contracts (such as classes), functions, and events, similar to other programming languages, so any developer can develop a smart contract. From this programming language, it is possible to develop programs that enforce the terms and conditions of a given agreement or contract, working on the principle that "code is the law". This principle concerns the fact that there is no need for a third party to enforce or control the execution. These programs are stored on the blockchain and will run automatically when predetermined conditions are met.

Smart Contracts are immutable, fault-tolerant, and cryptographically verified to ensure the trustworthiness of the program. In addition, they must be immune to any kind of external issue, either in the environment or coming from other programs, by aborting their execution or reacting properly, considering that typically involve financial operations.

There are several emerging smart contract platforms with various features to suit specific applications, such as financial services [45], insurance systems [46] and Hyperledger Fabric [47], healthcare management [48], car-sharing systems [49], energy trading [50], election systems [51] and many monitoring

[2]Solidity Documentation `https://docs.soliditylng.org/en/v0.8.11/`

applications [52, 53]. Additionally, smart contracts have enabled the creation of several concepts mentioned earlier, such as NFT and the creation of new cryptocurrencies based on the Ethereum blockchain, known as ERC-20 tokens.

Besides Solidity, research has been proposed to extend the development of smart contracts into various programming languages, both Domain-specific Languages (DSLs) [54] and General-purpose Languages (GPLs). DSLs can improve productivity by simplifying the complex code of the contract, promoting better communication between domain stakeholders, and eliminating development or transaction bottlenecks. There is also a growing interest in using GPLs like Java [55] and C++ [56] to develop smart contracts, thus a developer who is familiar with these languages can easily use his skills to write the contracts. Rust has also been one of the most adopted languages for writing smart contracts [57] because of its features such as memory safety, small runtime, etc. It allows writing smart contracts with fewer bugs and with low storage consumption, which is important considering the size limitations of blockchain.

DApps

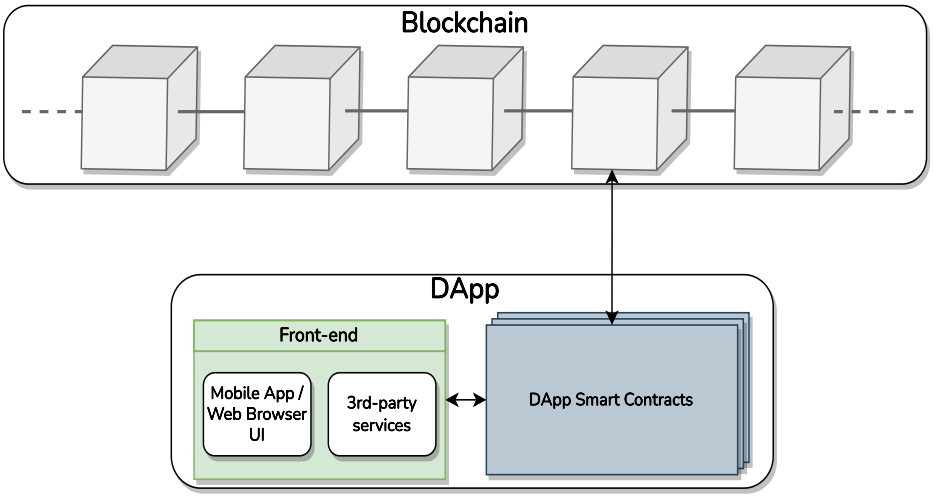


Figure 5: DApp Architecture.

Blockchain applications are considered DApps. A DApp is defined as an end-to-end decentralized blockchain application, including a user interface, smart contract(s), and the host blockchain. Typically, these applications run on a blockchain or Peer-to-Peer (P2P) computer network, rather than a single computer or from a traditional server, therefore, information from users is not subject to the control of any corporate entity. Some of well-known DApps running on a P2P network have been developed in the past, for example, BitTorrent[3] for distributing files and Tor[4] for private browsing. Recently, several different DApps have emerged, running on various platforms (blockchains), in categories such as smart building systems [58], education [59], games and gaming platforms [60], and authentication security systems [61].

[3] BitTorent https://www.bittorrent.com/pt/
[4] Tor Project https://www.torproject.org/

In the cryptocurrency domain, after the first DApp was published in 2016, the pace of DApps creation has increased exponentially and Ethereum is the leading platform capturing new DApps. Cryptokitties[5], a game built on Ethereum, was one of the most popular DApps, even overwhelming the network due to rampant adoption. Ethereum blockchain is currently the best platform for developing DApps [62] and has the most applications in the Gaming industry [63].

Contrary to a typical client-server application where a user interface, a server, and a backend database typically exist, DApps have a different architecture, as can be observed in Figure 5. DApps have smart contracts running on the blockchain, with all the contract logic. As front-end, traditional applications and DApps can have a variety of interfaces such as web or mobile User Interface (UI) developed in JavaScript or React. Since DApps are built on top of the blockchain, the information is public and securely accessible, so there is no need to rely on the application owner, who can manipulate the data or make it disappear. Every DApp requires a provider to talk or interact with the blockchain. These providers take JSON-RPC requests and return the response. This can be done by running its own node or by using the services of a node provider.

These applications must follow some criteria to be considered decentralized. A DApp must be open source and consensus-driven, based on the community or network activity. All information, transactions or operations, must be cryptographically signed, in order to maintain transparency with proper security. To operate on the network, a cryptographic token must be used, which will serve as the environment's native currency, for all kinds of transactions or rewards for users.

2.1.4 Blockchain Operation Costs

Blockchain is a transaction-based system. As already explained, in Bitcoin blockchain, for example, a new block is created and added to the main chain periodically by the miners. Each block contains valid and confirmed transactions sent from some accounts to others. The miners who process these transactions and blocks are rewarded, and this reward can vary depending on a transaction fee paid by users.

On Ethereum, the gas mechanism has been introduced to handle smart contracts operations costs. Gas refers to the measurement unit to the amount of computational effort required to execute specific operations on the Ethereum blockchain. The users need to set a Gas price to get a transaction processed and approved by miners, similar to Bitcoin. However, smart contracts have a particularity as it executes a given code. In this case, the fee paid to the miner is based on the gas consumed by the smart contract's code. Each transaction has a gas limit for consumption defined by the user requesting the transaction, typically overestimated, and a gas price that is the amount of ether (Ethereum network native token) the user is willing to pay for every unit of gas used by the transaction. This gas price depends on the lines of code in the contract and increases according to the operations required (arithmetic, bitwise, memory/storage, etc). This gas mechanism has the advantage of incentivizing miners and protects the network from attackers, considering that they will have to pay large fees to attack, but it makes some

[5]Cryptokitties - https://www.cryptokitties.co/

applications unviable due to costs. In periods of high adoption and usage, gas fees increase significantly, becoming impractical for some users and use cases. This is why several solutions have emerged to mitigate this problem [64, 65], mainly the layer-2 mentioned above in subsection 2.1.2.

2.1.5 Vulnerabilities

Despite all the advantages, as with any disruptive technology, there are some challenges that need to be addressed to make it robust and accessible to all. Due to the transparent and public nature of the blockchain, these ecosystems are highly sensitive to attacks. In the cryptocurrencies domain especially, attacks are attractive given the potentially profitable benefits to be gained from carrying out a successful attack.

The major blockchain challenges are mainly based on scalability, security, and decentralization, the scalability trilemma. At the top of the list of concerns is scalability, because blockchain does not meet the performance levels expected by users on a large scale. Also related to this is the problem of security and privacy, especially if the application is in privacy-demanding industries with specific confidentiality requirements such as finance, law, and health [66].

In some blockchains, such as Ethereum, there are some reasonable problems associated with smart contracts such as *Denial of Service (DoS)* [67] and *reentrancy* attacks [68]. In 2017, one of the most famous hacks in the blockchain occurred [69], where a DAO named "The DAO" lost around $50 million due to a reentrancy attack. Since smart contracts run on the blockchain, they inherit some of its properties that end up being undesirable for some applications. Although consensus mechanisms ensure the security of the network, to validate transactions and smart contracts, the data is public to miners, multiplying the attack surface. Most smart contracts are not confidential and private and therefore cannot handle sensitive data. Many of the advantages of smart contracts also become vulnerabilities and as with any software, bugs happen and can jeopardize systems and their users. In the cryptocurrency domain, there have been proposals for the review and inspection of smart contracts such as CertiK [70]. CertiK is a security-centric platform for analyzing and monitoring blockchain protocols.

Most of these issues arise from the immaturity of the technology, compared to traditional systems, and also its decentralized nature being a barrier for consumers that are used to rely on institutions. Also in the security domain, there are some cases of attacks such as double-spending [71], where the attackers manage to duplicate transactions, and in cases of cryptocurrency, spend the amount more than once. Another type of attack is *51% attack* [72] in which one or more attackers dominate more than 50% of the network's mining hash rate. This latter attack can decrease the market price of a cryptocurrency by as much as 15% [73]. Attackers with majority control of the network can, for example, disrupt the registration of new blocks, preventing other miners from completing blocks. Although it is a concern, this type of attack usually affects smaller networks.

Given the aforementioned problems, various solutions have emerged to mitigate them, mainly the previously mentioned layer-1 and layer-2 solutions. Layer-1 solutions emerged as a proposal based on the

idea of changing the fundamentals of blockchain protocols. Layer-2 solutions use alternative networks or resources to the main blockchain protocol.

2.2 Cartesi

Cartesi is the platform under study in this thesis and is a layer-2 solution for blockchains that allows developers to write and deploy scalable DApps or smart contracts. Cartesi provides features to smart contracts and DApps and aims to provide the convenience and scalability of the mainstream software world to DApp developers and users. Cartesi's whitepaper is outdated but updated information can be found in the website documentation[6].

DApps built for the Cartesi platform aim to achieve high computational scalability over large amounts of data, apparently without compromising decentralization or security, according to the authors. In terms of privacy guarantees, the data is not disclosed on-chain, it can remain private to the parties involved in the application. Moreover, the platform for building DApps is blockchain-agnostic, meaning that the way in which computations are formulated facilitates the development of applications and makes them independent of the blockchain used. It is built using a hybrid model, since Cartesi incorporates both blockchain (on-chain) and off-chain components to give flexibility to DApps and developers. Being a layer-2 solution, Cartesi moves DApp computations to off-chain, allowing to handle larger amounts of data without concerns about operation costs. Cartesi relies on its virtual machine with Linux Operating System (OS), the Cartesi Machine, to offer several advantages for DApps development. This machine is the core component that enables all off-chain computing.

In the on-chain component, a Cartesi DApp can specify the off-chain computations to be performed over large amounts of data, and these specifications are automatically followed by the Cartesi Nodes, in order to perform these computations, as explained below. In Cartesi, the blockchain is only used as a "supreme court", in case of a dispute over the result of the computation. When the parties involved disagree, a dispute resolution mechanism is triggered on-chain. The authors argue it is supposed to happen rarely, since the model provides an economic incentive for honest behavior, by penalizing dishonest ones. Even then, a dispute is resolved at negligible cost compared to related platforms, according to the authors.

This thesis will focus on all these aspects, with an in-depth analysis of all the components of this protocol, in order to reach the proposed goal.

2.2.1 Cartesi Competitors

Other works similar to Cartesi are TrueBit [74] and Arbitrum [75]. These technologies move computations off-chain, in order to improve the scalability of the blockchain. Despite this, these approaches differ in some aspects at the core of their design and architecture, as presented in Table 1. Truebit is a technology to help Ethereum perform heavy or complex off-chain computations but unlike other layer-2

[6]Cartesi Documentation - `https://cartesi.io/docs`

solutions, it does not allow for increased throughput. Arbitrum is one of the most widely used layer-2 as a solution to congestion and high transaction costs in the Ethereum network. As already mentioned, transaction fees increase when the number of network users increases and more transactions are requested. Arbitrum intends to lower network congestion and transaction costs by offloading as much work and data storage as possible from Ethereum's mainnet or layer-1.

Cartesi is based on a RISC-V Virtual Machine (VM) while TrueBit is based on a WebAssembly VM and Arbitrum on a Arbitrum VM. WebAssembly and Arbitrum are implemented between the applications and the underlying OS, providing an "OS" as a software layer to perform all the computation. Cartesi with the RISC-V VM supports the Linux OS, which allows it to provide developers access to all the tools or programming languages they need and are used to. Furthermore, Cartesi's architecture also allows the platform to be blockchain-agnostic, i.e. portable across other blockchains.

Regarding the consensus mechanism, Arbitrum is consensus-agnostic, so its implementation works equally well with any type of consensus. Cartesi is designed to achieve local consensus and only affected parties are required to perform the transaction process, thus allowing intensive off-chain computations. On the other hand, in TrueBit, the consensus is unanimous, meaning that anyone can object to faulty solutions.

	Cartesi	TrueBit	Arbitrum
Virtual Machine	RISC-V	WebAssembly	Arbitrum VM
Operating System	Linux	TrueBit OS	ArbOS
Consensus Mechanism	Local Consensus	Unanimous Consensus	Consensus-Agnostic
Blockchain-agnostic	X		
Incentive Layer		X	X
Large Storage Capacity	X		X

Table 1: Layer-2 Competitive Solutions.

In TrueBit and Arbitrum there is an Incentive Layer that helps prevent fraudulent or malicious processes by rewarding players for disputing incorrect results. On the other hand, Cartesi offers the responsibility of evaluation to the parties involved in the transaction process, and they are the ones who resolve disputes if necessary. Although further verification by a dispute resolution mechanism may be required, there is no built-in incentive layer.

Finally, real-world applications have very large storage requirements, and this is a challenge that TrueBit's design doesn't address, unlike the other solutions. In Cartesi's case, it is possible to handle large amounts of data through the Logger service, representing it cryptographically on-chain.

2.3 Confidential Computing

Confidential computing [76] is causing a paradigm shift in security area. The use of confidential computing, secure enclaves, and advanced cryptography techniques is increasingly being used to reduce the risk of attack for various applications. The majority of significant Instruction Set Architectures (ISAs) now include support for third-party confidential compute, such as AMD Secure Encrypted Virtualization [77], Arm TrustZone [78] and Arm Confidential Compute Architecture[7], Intel's Software Guard Extensions (SGX)[8], and Intel Trust Domain Extensions [79]. These technologies provide strong guarantees of confidentiality and integrity to the code and data against privileged attackers by properly isolating the sensitive hardware or firmware.

Blockchain technologies leveraged by confidential computing can use hardware-based privacy to secure computations. The solution may involve either encrypting the entire blockchain ledger to ensure data confidentiality or, for example, run just a node in an enclave. Following all the blockchain vulnerabilities mentioned in subsection 2.1.5, it can be observed that security is still an issue, mainly due to the transparency and pseudoanonymity provided by the architecture. A smart contract running on a blockchain is clearly visible to the public, since its data is replicated to all nodes on the network. As a solution to address these problems, blockchains can be paired with Trusted Execution Environments (TEEs) to run applications that require privacy. The data and computations of a smart contract or DApp can be protected inside a TEE, isolating it from untrusted parties. Confidential computing allows to conduct computations on the data without having access to that data, thus being ideal for privacy-preserving applications.

2.3.1 Trusted Execution Environment

With TEEs, applications, typically known as Trusted Applications (TAs), can be securely executed isolated. This isolation is achieved through a combination of hardware mechanisms, such as hardware memory protection. Thus, it is possible to run security-critical applications in a protected manner, unable to be accessed by the OS. A TEE must isolate the TAs within the TEE, and from the TEE itself. Changes to these applications can only be made by authenticated entities. Access to peripherals is securely provided to Application Programming Interfaces (APIs), under the control of the TEE. TEEs have random number generation, cryptography, and timestamps as key aspects to add security to TAs.

Several TEE technologies have been implemented that achieve the aforementioned security features. The most prominent technologies are Arm TrustZone, which runs on Cortex-A processors and, more recently, microcontrollers [80] with Cortex-M, and SGX, which runs on Intel processors and servers. Arm Cortex-A is mostly used in the mobile segment, and other high performance applications such as smart-TVs and infotainment systems. Arm TrustZone [81] is a technology that provides hardware-enhanced

[7] Arm Confidential Compute Architecture https://developer.arm.com/documentation/den0125/0100/Arm-CCA-extensions

[8] Intel Software Guard Extensions https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

separation. TrustZone has been widely embraced to protect critical operations on mobile [82] and embedded devices [83] due to the implementation of the necessary mechanisms to implement a TEE. SGX is another TEE technology that helps protect data in use via unique application isolation. This technology allows developers to divide their applications into a trusted and untrusted module, to help increase application security. Other prominent TEE examples include MultiZone[9] and Keystone [84] on RISC-V. MultiZone Security is the first TEE for RISC-V and has been used to develop secure IoT systems [85, 86] on RISC-V processors. Keystone is an open-source framework for developing customizable TEEs based on the RISC-V architecture.

In addition to specific hardware features that establish a TEE, virtualization can also be used to a similar effect. Virtualization is used in multiple fields of computing, with hypervisors focusing on different use cases from cloud computing [87], to mixed critically systems [88–90]. Leveraging the flexibility enabled by hardware supported virtualization, the research community has proposed multiple virtualization based solutions that protect the execution of security critical applications from the OS [91–96]. These solutions are supported by various architectures, including x86, Arm and, recently, RISC-V [97], since they feature hardware support for virtualization.

It should be noted that TEEs are not a panacea. Intel SGX and Arm TrustZone, for example, have been vulnerable to many vulnerabilities over the years [98, 99]. Although solutions exist to mitigate some of the problems [99–101], vulnerable trusted computing bases, and side-channel and hardware attacks remain significant challenges for TEEs. Notwithstanding, TEEs can improve on the weakest aspects of layer-2 solutions, by raising the bar on the difficulty of mounting attacks. For layer-2 solutions like Cartesi that run off-chain computations, it is important to ensure that these computations keep the same security provided on-chain. Thus, future applications may combine both technologies by running part of their native computations, inside hardware TEEs, or by running an entire blockchain node within this trusted environment.

Besides the security provided by TEEs, the question may arise about how to know if an application is running securely in a TEE. For this, the concept of attestation is introduced. In the context of confidential computing, attestation is about proving the trustworthiness of the TEE. It proves which set of software layers are running on the TEE. An application running on the TEE can perform the attestation, establish a secure channel, and retrieve the secrets using the tools available for specific TEEs. Intel SGX features attestation execution, which makes it possible to prove the correct execution of a program in the enclave, by issuing a remote attestation. Remote attestation can be considered a digital signature, using a private-key only known to the hardware [102].

2.3.2 Trusted Execution Environment - Containers

Cloud computing is emerging at a significant pace, leveraging lightweight solutions to outsource workloads to the cloud. More than just using VMs, the cloud computing approach is now shifting to container

[9] MultiZone Security for RISC-V https://hex-five.com/multizone-security-sdk/

technologies. Containers are standard units of software that package code and its dependencies into an image that allows to reliably run applications across different computing environments. Both containers and VMs are forms of virtualization. While VMs allow one piece of hardware to host multiple OSs as software, containers virtualize the OS by dividing it into compartments to run container applications. With containers, applications and their dependencies are distributed in virtualized environments and can run anywhere the corresponding OS is running. Figure 6 presents the Docker[10] container architecture, the most popular container ecosystem, alongside the typical VMs architecture.

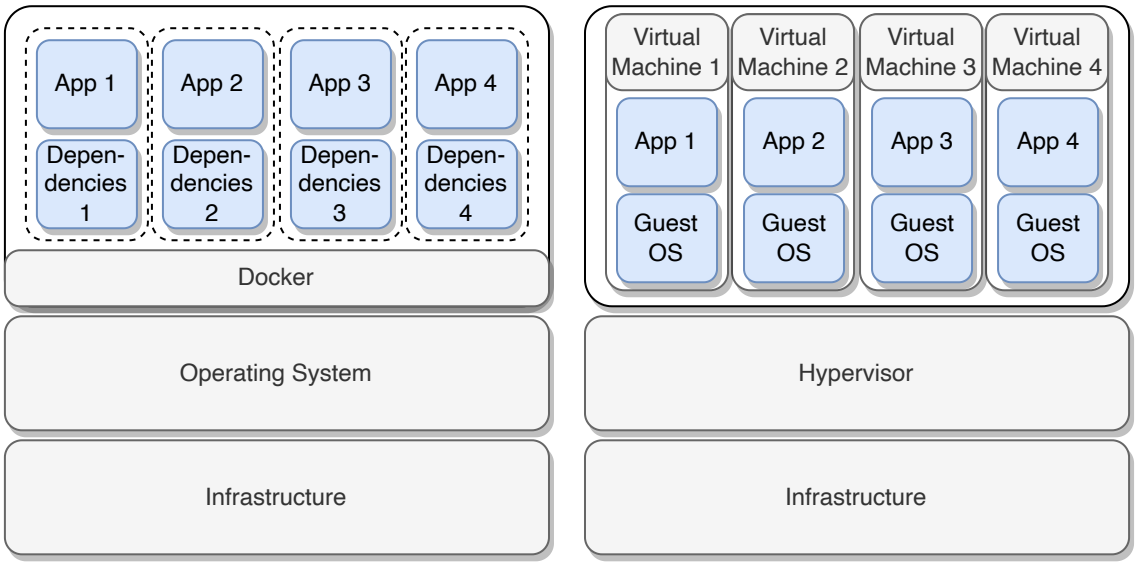


Figure 6: Docker Containers and Virtual Machines Architecture.

Cloud services and containers have been widely adopted due to their advantages, but there are security concerns [103]. Containers rely on the OS to enforce their security guarantees and this brings several security risk. Due to the kernel being shared between all container instances, there is a wide attack surface, allowing attackers to introduce vulnerabilities or malware into the images [104]. Considering the vulnerabilities, a new security mechanism is emerging, so-called Confidential Containers, which enable cloud Confidential Computing. Confidential computing protects sensitive data, rendering it opaque to the cloud provider and also to the malicious users, creating a trusted environment [105]. Features listed in subsection 2.3.1 allows for memory and data isolation and other security features that vary by hardware vendor. There are novel architecture proposals [106, 107] which combine the properties of TEEs with standard containers to increase security for container workloads, based on AMD SEV and Intel SGX TEEs. Furthermore, Kata Containers [108] are lightweight VMs acting like containers, providing isolation and security benefits from both technologies.

[10]Docker - Accelerated, Containerized Application Development `https://www.docker.com/`

2.4 Related Work

In the following section, related work based on TEE implementation in blockchain will be analyzed and discussed, mainly in terms of security. As mentioned in section 2.3, pairing blockchains with TEEs could be a solution for running applications that require privacy. Some proposals that already combine these technologies to fix blockchain vulnerabilities will be analyzed.

Decentralized IoT Data Management using Blockchain and TEE

As the adoption of IoT devices increases, proper verification of data access, transparency, and privacy are critical due to the vast amount of data generated by these devices. This work proposes a decentralized data management system [109] for IoT devices in which all data access permissions are enforced using smart contracts and all access records are stored on the blockchain. The smart contracts run on Etheurem's blockchain and Intel SGX is used as the TEE technology.

This system is leveraged by a framework that only stores a cryptographic hash of the data on the blockchain and then stores the data on secure storage using TEEs. TEE is used to ensure the security and privacy of the sensitive part of the application (code and data). By leveraging Intel SGX-based TEEs, this framework provides data protection against non-authorized access from adversaries. Third-party users will need to request permission to access the blockchain data using the smart contract API. The hash of the data is returned and used to retrieve the data from the SGX platform.

Teechain

Teechain [110], has been proposed as an off-chain payment protocol that performs secure, efficient, and scalable transfers on top of a blockchain. The fundamental idea behind Teechain is to establish bidirectional payment channels between pairs of participants and exchange funds directly and securely, rather than placing transactions on the blockchain for each unique payment. To secure these payment channels and to prevent fraud by network participants, Teechain makes use of the confidentiality and integrity provided by TEEs.

Teechain uses TEEs to force the proper operation of distrusted parties during off-chain exchanges. In the Teechain system, each participant operates its own TEE that runs the protocol, and to ensure that transactions are executed correctly across payment channels, participants attest each other, thus proving the veracity of the code within the TEE. Teechain allow users to dynamically move funds between different payment channels. Since TEEs protect the internal channel state and release it only at the end of the channel, they ensure that users cannot attack using the stale state, i.e., the state stored in the TEE that no longer reflects the underlying persistent state.

Teechain's implementation uses SGX as the TEE technology and Bitcoin as the operational blockchain. Despite this, it can be implemented on other blockchains and SGX can be replaced with alternative TEE implementations, according to the authors.

PoTS - Proof of TEE-Stake

A lot of research has been done on PoS protocols, as an alternative to the PoW that is one of the most applied consensus mechanisms in blockchain projects. PoS is a good alternative mainly due to its energy efficiency, but PoS is still not widely deployed. PoS-powered blockchains have some vulnerabilities to security threats such as *nothing at stake* and *long range* attacks [111]. *Nothing at stake* attacks occurs when a validator creates a block without "spending anything", i.e., without a stake. A *long range* attack is when a validator has no risk of loss due to misbehavior on the network.

Given these security shortcomings, PoTS emerged as a secure PoS protocol to address these problems. PoTS [112] leverages TEEs to protect against the mentioned attacks on PoS protocols, according to the authors, without affecting performance. The core idea of PoTS is the use of TEEs to enforce the honest behavior of validators. The solution is to run the protocol inside trusted applications, not allowing the validators to deviate from the protocol instructions. Nevertheless, only validators need to be equipped with TEEs, which makes PoTS viable for simple users who do not wish to mine. Since the leader election operation is performed within the TEE, it prevents any adversary from compromising future leaders as well as malicious validators. This protocol tolerates compromised TEEs, in case the attacker does not control a significant number of TEE enabled nodes. In combination with TEEs, PoTS also uses cryptographic techniques to prevent attacks that corrupt the network by updating signature keys and deleting old keys.

Scaling Blockchain Systems via Sharding and TEEs

As noted in section 2.1, scalability is an issue with existing blockchains, largely because of their consensus mechanisms. As seen also in subsection subsection 2.1.1, there are solutions to these problems, such as layer-1 solutions based on *Sharding*, a mechanism that splits the blockchain into several shards, in order to improve their transaction throughput and consequently, scalability. To improve upon the *Sharding* technique, there is a solution that uses *Sharding* to scale blockchain systems [113], with the help of TEE technology.

The first challenge of this approach focuses on solving the existing consensus problems, leveraging SGX to eliminate the problems in consensus decisions. The second challenge is to achieve a secure and efficient shard formation. To solve this, they leverage SGX to design a secure shard, implementing trusted randomness, i.e., producing random outputs to make it difficult for attackers to decrypt. This solution uses Intel SGX to provision the TEE, but its design is also compatible with other TEE instantiations such as TrustZone.

Secret Network: A Privacy-Preserving Secret Contract & Decentralized Application Platform

Due to the properties of smart contracts, several blockchains are hosting diverse DApps and with growing adoption, it has become increasingly difficult to maintain network security and privacy mainly

because of the transparency feature of blockchain. In this sense, Secret Network [114] arose with the purpose of achieve smart contracts data privacy for DApp development.

Secret Network is a DPoS blockchain network, built using Cosmos development tools, that introduce data privacy by creating the Secret Contracts. These contracts leverage cryptographic key management, encryption, and decryption protocols and TEEs. New nodes registered on the Secret Network are able to check the validity of the hardware and TEE through remote attestation, proving the authenticity of the node. Nodes also cannot access and tamper each other's computation data as it runs within the trusted environment. The consensus hashes are securely stored inside the TEE of each node, thus allowing for decryption of inputs to be used and computed in a safe environment. The Secret Network leverages SGX as TEE but it works with other TEEs implementations.

Oasis Protocol

Oasis Foundation built the Oasis Blockchain Platform [115], a layer-1 protocol to achieve scalability and privacy in smart contracts. The platform design aggregate multiple smart contract running independent and in parallel, named "ParaTimes". Each Paratime can employ different verifiable and confidential computing mechanisms such as discrepancy detection, multiparty computation [116], fully homomorphic encryption [117] and zero-knowledge proofs [118].

TEEs are used in ParaTimes to allow private and confidential execution of smart contracts, not requiring the use of specific technique, each one defines its own smart contract execution environment and chooses the mechanism to verify the results. The architecture is modular in the way that it separates the Consensus Layer and the Paratime Layer, thus increasing security and efficiency on the network, since smart contracts are isolated from external effects of consensus operations. The modular architecture also allows a ParaTime to be unique and to use, or not use, any type of confidential computing, and still coexisting with others that do. These type of features and mechanisms result in the design resembling layer-2 solutions.

Chapter 3

Cartesi Architecture

In this chapter, we will present the layer-2 platform under analysis, Cartesi, identifying the multiple components of the system, as well as the interaction between them. Since it is a layer-2 platform, the computations are moved off-chain. The off-chain operations leverage a virtual machine, whose role in the framework will be explained in detail. Having a clear distinction between on-chain and off-chain is crucial to understand the logic of the applications developed on the platform, mainly due to the security implications. We will present the mechanisms used to build the DApps on top of the blockchain. This will explain the available choices for the selection of node topology and the consensus model. These concepts are relevant for a developer to build a DApp.

3.1 Descartes

The Cartesi project started in 2018 and later, in 2020, they launched Descartes. The team has continuously been developing this version, with several use cases and proofs of concept, and is currently working on a new version, the Cartesi Rollups. This thesis will address only on Descartes.

3.1.1 Overview

Cartesi is a multichain layer-2 infrastructure for the development and deployment of DApps that can leverage a Linux OS. Cartesi uses a hybrid system, where it is possible to run DApps both on the blockchain and off-chain. The main goal of the platform is to bring scalability to the Ethereum ecosystem and also to developers by providing them with tools to facilitate their adaptation to DApps development, without having to master Solidity and blockchain related concepts. Cartesi's core technology relies on its virtual

machine called the Cartesi Machine. This machine is the basis for two Cartesi's frameworks: the Descartes Software Development Kit (SDK) and the latest implementation, the Cartesi Rollups.

Descartes SDK is a framework for developing and running DApps using a decentralized computational oracle for verifiable computations. In blockchain, when off-chain data (any kind of data or services) and on-chain code (smart contracts) are combined, allowing to connect DApps with traditional systems, it is called an oracle. These computations are performed off-chain on a Linux environment supported by the Cartesi Machine. In contrast to a normal Ethereum DApp, in a Descartes DApp, the participating nodes try to reach consensus locally and off-chain, moving to on-chain only in case of disagreement. Blockchain is used only for dispute resolution, by means of an interactive verification algorithm, accompanied by economic incentives to punish dishonest parties and benefit honest ones. These mechanisms are abstracted from the user who only interacts with the client software and to the developer, whereby an API is provided to perform complex off-chain calculations with automatic dispute resolution.

Descartes Components

A DApp that uses the Descartes SDK, needs both the Descartes on-chain and off-chain components. The on-chain components are a set of smart contracts developed by Cartesi that encompass all the on-chain mechanisms, with the core contract named "Descartes". The main function to interact with Descartes is the `instantiate` function, since it is the function that triggers execution by requesting a given computation to be performed off-chain. This function will be described in detail in DApp Execution Flow (subsection 3.1.4). The off-chain core component is called Descartes Node, and it is the component that allows clients to interact with Descartes DApps. All the parties involved in running computations must run a Descartes Node, which consist of a variety of internal services intended to interact with the blockchain, and also the Cartesi Machine, whose details will be presented in subsection 3.1.2. Moreover, the Descartes Node can also be composed of additional off-chain services that the developer wants to include to interact with the client or to get external data. Descartes Nodes are made available as a set of Docker containers to be run on a computer under the owner's responsibility. These nodes are in charge of automatically performing computations, submitting results, and verifying claims made by other nodes. For each DApp, one node is selected as *claimer*, responsible for submitting the result of the computation, while the remaining nodes are *challengers* that verify the claim and can agree or challenge this result.

3.1.2 Cartesi Machine

Cartesi Machine, depicted in Figure 7, is the solution for verifiable and reproducible computing. It is a virtual machine that emulates a RISC-V microprocessor and can run a full-fledged Linux OS. The emulator off-chain implementation is written in C/C++ following POSIX standards. DApps running inside Cartesi Machines can process larger amounts of data than running in smart contracts because it runs off-chain, unburdened by the costs associated with the blockchain's consensus mechanism. The main features of the Cartesi Machine are transparency, since it exposes its state in the blockchain via Merkle

tree hashes, and reproducibility, since identical machines performing the same computation get exactly the same results.

This machine can be analyzed from different perspectives: i) host perspective, ii) target perspective and iii) blockchain perspective. The host perspective is a view of the Cartesi Machine emulator external environment. It is relevant for developers to configure and manipulate the machine from the interfaces: C++, Lua, gRPC and command line interface. The target perspective refers to the environment inside the Cartesi Machine, namely the RISC-V architecture and the Linux OS. Finally, there is the smart contracts' view of the machine, the blockchain perspective. The blockchain has access to the current state and all state changes of the machine, represented by Merkle tree hashes. An overview of these perspectives is fundamental for a developer of DApps.

The machine architecture can be divided into two parts, so-called processor and board. The processor performs the DApps computations. The board shapes the environment with a variety of devices and memories (ROM, RAM, and flash drives).

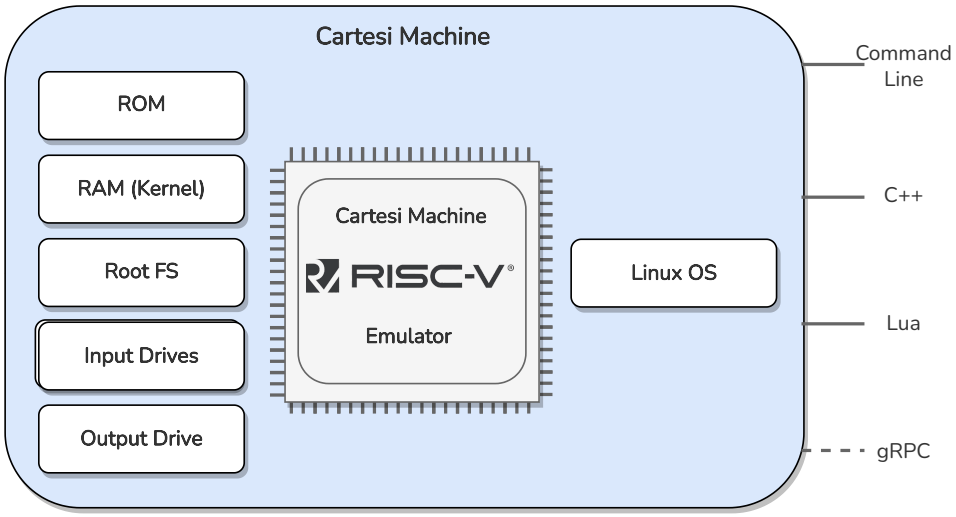


Figure 7: Cartesi Machine architecture.

Processor

The emulator implements a RISC-V's RV64IMASU ISA. This specification corresponds to a 64-bit machine with Integer arithmetic, Multiplication and division instructions, Atomic operations and optional Supervisor and User privilege levels. This also include support to Sv48 mode of address translation and memory protection.

The whole state of the processor fits inside 512 bytes, divided into 64 registers, each of them with 64-bits and mapped to the lowest 512 bytes in physical memory. Memory-mapping is presented in Table 2. The RISC-V ISA defines the majority of these registers, except those beginning with *i*, which are Cartesi-specific. The `mcycle` register is one of the most important registers, which is incremented at each CPU cycle. This register can be used to find an instant of a computation, which will be useful to identify the

divergence point in the Verification Game partition stage that will be explained later. In addition, it can be used to limit the number of cycles of a Cartesi Machine computation.

Offset	Register	Offset	Register	Offset	Register
0x000	x0	0x138	mtvec	0x190	stvec
0x008	x1	0x140	mscratch	0x198	sscratch
...	...	0x148	mepc	0x1a0	sepc
0x0f8	x31	0x150	mcause	0x1a8	scause
0x100	pc	0x158	mtval	0x1b0	stval
0x108	mvendorid	0x160	misa	0x1b8	satp
0x110	marchid	0x168	mie	0x1c0	scounteren
0x118	mimplid	0x170	mip	0x1c8	ilrsc*
0x120	mcycle	0x178	medeleg	0x1d0	iflags*
0x128	minstret	0x180	mideleg		
0x130	mstatus	0x188	mcounteren		

*Cartesi-specific

Table 2: Processor Memory Mapping.

Board

The board refers to the mapping of the memories and devices and the interaction with the processor can be shown in the Table 3. ROM starts at address 0x1000 and is in charge of holding the devicetree and a bootstrap program. This program sets register 0x10 to 0, 0x11 to point to the devicetree, and then jumps to RAM start address 0x80000000 where it is expected to contain the boot image. The board also maps two non-memory devices to the physical address space, the Core Local Interruptor (CLINT) and the Host-Target Interface (HTIF). The CLINT is responsible for maintaining control and status registers which are associated with the software and timer interrupts, and is mapped into register *mtime* and *mtimecmp*. This interrupt management is crucial to the machine reproducibility. The HTIF mediates communication between host and target. It is mainly used as a communication port during machine interactive sessions to halt the machine or to inform when it has been halted, for example. Memory ranges are reserved for flash drives, which will typically be loaded with file system images or raw data corresponding to the Input/Output (I/O) drives of the computations.

Unlike traditional VMs, a Cartesi Machine is not intended to be interactive. When a Cartesi Machine is instantiated, it is supposed to boot, bring up the OS, execute desired computations and then halt. The Cartesi Machine specification includes metadata describing its input drives, which can be more than one, and its single output drive. After the requested execution, the obtained result is written to the output drive.

Physical address	Mapping
0x00000000 – 0x000003ff	Processor shadow
0x00000800 – 0x00000Bff	Board shadow
0x00001000 – 0x00010fff	ROM
0x02000000 – 0x020bffff	CLINT
0x40000000 – 0x40007fff	HTIF
0x80000000 – *	RAM
* – *	Flash 0
...	...
* – *	Flash 7

Table 3: Physical Memory Mapping.

Machine On-chain

The entire state of a machine, as well as the computations performed, should not be stored on-chain. Due to the size limitations and costs associated with blockchain, storage is an expensive operation. Thus, as already mentioned in blockchain perspective, the state of the machine, with all the regions represented in the Table 3, is submitted in the blockchain in the form of Merkle tree. A cryptographic hash of an entire machine's state can be generated and stored right before execution starts. This hash is called the Cartesi Machine template and will be used to represent on-chain the machine at the instantiation of a computation in Descartes. In the template, the flash drives are filled with zeros, which makes the template a representation of a virgin machine. When instantiating a computation, the empty drives are replaced with the provided drives, from Merkle tree operations, resulting in a new hash of the machine with all its state.

Cartesi machines allow the development of two types of applications. In the first type, considered stateless, the machine is initialized, it runs the necessary computations and then halt, thus being able to inspect the outputs generated. The second type, statefull, waits for inputs and once it receives them, it performs the requested computations and produces the response. The host checks the outputs, prepares the next request and takes over the machine to run again.

Initialization

Since Cartesi Machines development happens in a host platform and only after will the applications run in a target system, there is a development environment to initialize and test Cartesi Machines without having to deploy a full blockchain system. The Cartesi team provides a Docker image, called *playground*, which comes with a pre-built emulator, ROM, RAM and root file-system images and the RISC-V cross-compiler, for testing purposes. Once inside playground, it is possible to initialize a Cartesi Machine to perform a given computation. The following command Listing 3.1 builds a Cartesi Machine with the respective parameters.

```
cartesi-machine \
  --rom-image="/opt/cartesi/share/images/rom.bin" \
  --ram-length=64Mi \
  --ram-image="/opt/cartesi/share/images/linux.bin" \
  --flash-drive="label:root,filename:rootfs.ext2" \
  --flash-drive="label:input,length:1<<12,filename:input.raw" \
  --flash-drive="label:output,length:1<<12,filename:output.raw,shared" \
  -- $'dd status=none if=$(flashdrive input) | lua -e \'print((string.
     unpack("z", io.read("a"))))\' | dd status=none of=$(flashdrive output
     )'
```

Listing 3.1: Cartesi Machine initialization.

For simple tests, the default `--rom-image`, `--ram-image`, `--ram-length`, and some `--flash-drive` can be used, omitting the parameters, and just simple drives must be defined. For a more complex simulation, the drives needed to perform the desired computation can be specifically created. The desired root file-system can be created using the Buildroot[1] tool, where necessary packages can be added for use inside the Cartesi Machine. The remaining input and output drives can be created as raw flash drives containing data that will be read in the case of input and written in the case of output. Finally, a command can be passed to the machine to run during its execution. As explained previously, the flash drives are mapped into the machine's 64-bit address space, so the start and length of each of them must be specified. By default, the start of the first flash drive, typically the root file-system, is set to the beginning of the second half of the address space and the others are mapped consecutively, spaced by 2^{60} bytes. The length parameter must exactly match the size of the image file referred to by the filename parameter.

After running the above command, the Cartesi Machine will be initialized, with the file `rootfs.ext2` mapped to the `'root'` drive, and will read the contents of the input drive from `input.raw` file, use a Lua script to ensure data is read as a null-terminated string and write them to the output drive. Finally, the machine halts and the contents of the `output.raw` file can be inspected to verify the result.

3.1.3 DApp Architecture

Typically DApps are composed of a blockchain node and a client software. The blockchain node can be a local Ethereum node like geth[2], parity[3] and Ganache[4] or a remote one such as Infura[5]. The DApp logic runs in the Client Software, which has a front-end that is linked to Ethereum, for example, via *web3.js* JavaScript Library, which is bundled with the front-end resources and served to a browser by a web server. In Descartes DApps the architecture is similar, the Descartes Node plays an identical role as the blockchain node and the Client Software runs the logic implemented by Descartes smart contracts.

[1] Buildroot - https://buildroot.org/
[2] Official Go implementation of the Ethereum protocol - https://geth.ethereum.org/
[3] Parity Ethereum Client - https://www.parity.io/technologies/ethereum/
[4] Ganache Truffle Suite - https://trufflesuite.com/ganache/
[5] Infura - https://infura.io/

Figure 8 shows a possible approach for the architecture of a Descartes DApp, since different DApps may have a slightly different design depending on the use case. All the components presented will be analyzed individually, as well as the interaction between them and the respective on-chain and off-chain distinction.

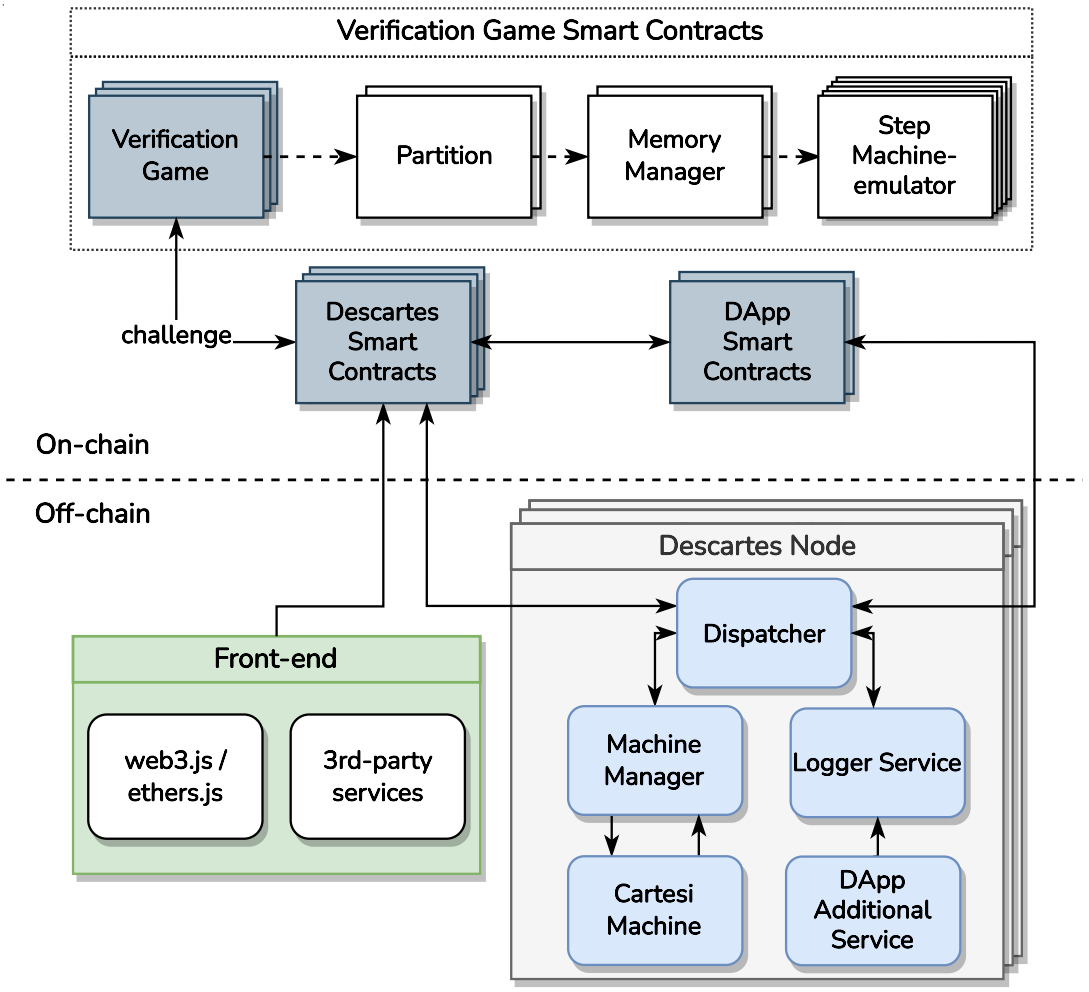


Figure 8: Cartesi DApp Architecture.

Off-chain

Front-end: As with any application or DApp, the user interacts with the front-end that defines the UI logic, which can be a web browser or even a command line interface (e.g. Hardhat or Python) that can be developed, for example, in JavaScript or TypeScript, with the desired design for the applications needs. By using Hardhat, the front-end can interact directly with the Descartes smart contracts or the DApp smart contracts to mainly perform compilation, deployment and the testing of them. The front-end can also interact directly with the Dispatcher to observe the state of the blockchain. This allows it to show that there is a disputed computation, know if a claim has already been submitted by the claimer, get some information from other components, given its connection to most of them.

Descartes Node: As mentioned in section 3.1, Descartes Nodes are composed of several components and these are in Docker containers, started via Docker compose to start multiple containers simultaneously. All DApps need to have Descartes Nodes running and these can be run by DApp users themselves or by a third-party organization or institution they trust, thus undermining decentralization, a concept that will be addressed in section 3.2. Within each Descartes Node there are the off-chain components that interact with the blockchain, validate and process the computations, including the Cartesi Machine.

Cartesi Machine: Detailed explanation in subsection 3.1.2. This component has the Cartesi Machine Emulator which is the reference off-chain implementation of the Cartesi Machine specification. The machine is instantiated by the Machine Manager, receives the input drives provided by the user, performs the requested computation, and returns the generated output. For each DApp, a specific template machine hash is stored in a directory accessible by Machine Manager to bootstrap the propper Cartesi Machine. This stored template hash contains all the components needed to perform the computations required for the DApp.

Machine Manager: Machine Manager contains a program responsible for managing different sessions of Cartesi Machines. It contains a high level gRPC API to create and interact with machine emulator sessions. This component is combined with volumes that are later used to provide the flash drives and support files needed for instantiation of Cartesi Machines.

Logger Service: The input drives of the Cartesi Machine have size limitations. These limitations are not inherent to the machine itself, but to the maximum transaction size imposed by the blockchain, in this case, by the Ethereum network. The Logger service allows to overcome this, allowing much larger drives to be submitted to the machine, to perform Cartesi computations without large transaction costs. Logger Service is the combination of the on-chain and the off-chain Logger modules that together allows publishing and retrieving data more efficiently to and from the blockchain. This service splits the contents of the input drives into chunks that are stored as call data, in the case of Ethereum.

Dispatcher: The Dispatcher is the central component of the Descartes Node since it communicates to various other components and is the bridge between the off-chain and the on-chain. This component is stateless and acts primarily as transaction and state manager during the entire DApp execution flow. It observes the state of the blockchain and broadcasts it to the requesting component, for example, to inform a user of the current state of the DApp at a specific instant. The Dispatcher is also responsible for collecting the inputs and requesting the proper emulations to be performed. Once the results are available, it handles the posting of transactions considering the blockchain and transaction pool indiocrasies.

On-chain

DApp Smart Contracts: The on-chain portion of the application logic is written in a set of DApp Smart Contracts. The DApp logic encompasses the application actions and events that should trigger some computation on the Cartesi Machine, especially those that require verification. Besides smart contracts

for the logic, there can be contracts for the handling of the application's utility token, as well as a Faucet contract for distribution of those tokens to the users. Typically, the main DApp smart contract is the one capable of using the Descartes smart contracts which are already deployed in the network, when the Docker compose starts up the services. The constructor of this main contract receives as argument the effective address of deployed Descartes smart contracts, which enable code to issue transactions and query results from it. This contract implements the `instantiate` method which is one of the endpoints to interact with Descartes. The `instantiate` method will be detailed in subsection 3.1.4, with all the parameters that must be passed from the DApp smart contracts to Descartes. The most important point to mention is that in this method the input drives are passed to the machine to perform the intended computation. Once the computation is performed, the results are available via another endpoint, the `getResult` method.

Descartes Smart Contracts: This set of smart contracts are the on-chain component of Descartes SDK. All interactions with the Descartes infrastructure are done through a single smart contract, called Descartes. The Descartes smart contract is in charge of call and interact with all of the contracts provided by Cartesi, especially the ones that trigger the Verification Game. As mentioned above, the most important endpoints are the `instantiate` and `getResult` functions. After setting all the parameters to `instantiate`, the contract instantiates a Descartes SDK instance and controls the computation states. In addition, the Descartes smart contract imports several other Cartesi smart contracts to handle Merkle tree operations, indexed instantiation of each computation, and the on-chain part of the Logger service. This set of contracts is also interfaced with the Verification Game contracts that will be explained next.

Verification Game: This is the mechanism that settle disputes regarding the results to the Descartes computations. A dispute begins whenever a node challenges the claimer result. The Verification Game works together with Arbitration D-Lib to resolve any disputes that might occur during the execution of a DApp. **Arbitration D-Lib** is the amalgamation of the on-chain and off-chain protocol to handle disputes in case of a challenge after the claimer submits the result of a computation. The off-chain service is responsible for watching the blockchain events to trigger the Verification Game and react according to the game's actions. The on-chain smart contracts encompasses the whole on-chain part of the Verification Game. After a computation is instantiated, the initial state is *WaitingClaim*. Once in this state, there is a deadline for the claimer node to submit a result. When the claimer submits the final hash to the computation result, there is a period where challengers can either accept or challenge this result. If the result is challenged, this will trigger the Verification Game, following three stages, **Partition**, **Memory Manager** and **Step**. In the Partition stage, the parties involved look for the first execution cycle in which they diverge. They disagreement in the instruction on which there was agreement immediately before but disagree after it is executed. The next stage is when the claimer has to fill the Memory Manager with his entire activity log, that is, his off-chain state referent to the disagreement point. This information is consumed, in the next stage, by the **RISC-V on-chain emulator**. The Step stage has this name, since it is a state transition function that takes the machine from state s_i to s_{i+1}, which means, the transition state

at the divergence point, using the log provided by Machine Manager. The instruction transition is then executed by the on-chain machine step implementation, which is a RISC-V emulated in smart contracts written in Solidity. Once executed, the on-chain machine should reach a consistent state s_{i+1}, and with this result, identify the winner.

3.1.4 DApp Execution Flow

The execution of a Descartes DApp starts with a request made by the DApp's smart contract to Descartes' smart contracts. This request is made by calling the `instantiate` function (Listing 3.2).

```
function instantiate(
    uint256 _finalTime,
    bytes32 _templateHash,
    uint64 _outputPosition,
    uint8 _outputLog2Size,
    uint256 _roundDuration,
    address[] memory _parties,
    Drive[] memory _inputDrives) external returns (uint256);
```

Listing 3.2: Instantiate Function to Interact with Descartes Smart Contracts.

The parameters of this function are `_finalTime`, `_templateHash`, `_outputPosition`, `_outputLog2Size`, `_roundDuration`, `_parties`, and `_inputDrives`. The `_finalTime` represents the maximum number of cycles the Cartesi Machine should run to execute the requested computation. The `_templateHash` is the Merkle tree root hash that represents the entire content of the template machine, which has already been explained in subsection 3.1.2. The `_outputPosition` and `_outputLog2Size` stand for the position of the output drive inside the address space of the Cartesi Machine, and the log_2 of the drive's size, given in bytes, respectively. The `_roundDuration` is the time each participant has to submit results to the blockchain, whether it is a claim or a challenge to dispute. The array of structs `_parties` lists the addresses of accounts that are participating as validator nodes in the computation. Finally, `_inputDrives` that describes an array of Drives representing each input that the machine should receive to calculate the output.

All the states of a DApp execution flow are represented in Figure 9. After instantiate, the first state is *WaitingProviders*, in which, if the inputs are not filled at the function call, there is a time period to submit the contents of these drives to effectively start the machine execution. In these cases, external users, called providers, are needed to supply these input drives. The data can be provided directly via `provideDirectDrive` function, in the `directValue` variable of the input Drives. In case of large amounts of data, it will be handled by the Logger service and sent via `provideLoggerDrive` function. Once the inputs are populated, the computation start automatically by the Descartes Nodes running by the parties involved and the state changes to *WaitingClaim*. At this state, there is a deadline for the claimer node to submit the result of the proposed computation. Therefore, a new session of Cartesi Machine is initialized and the required computation is performed in order to fulfill this deadline. At the end of the

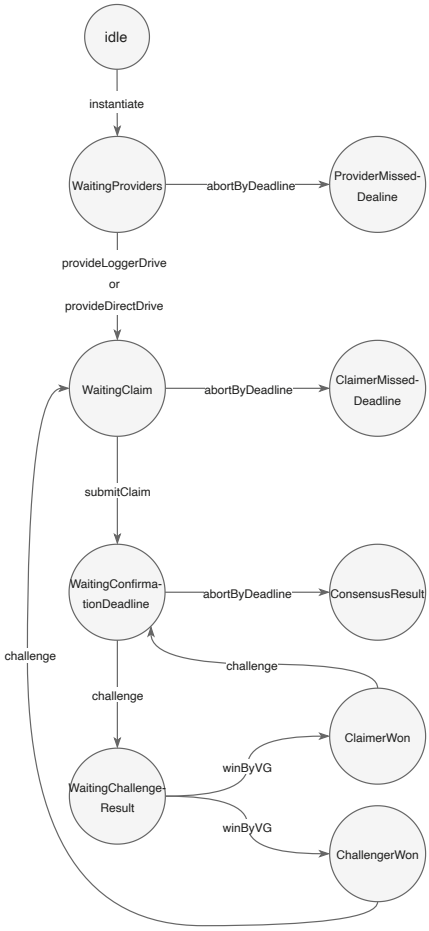
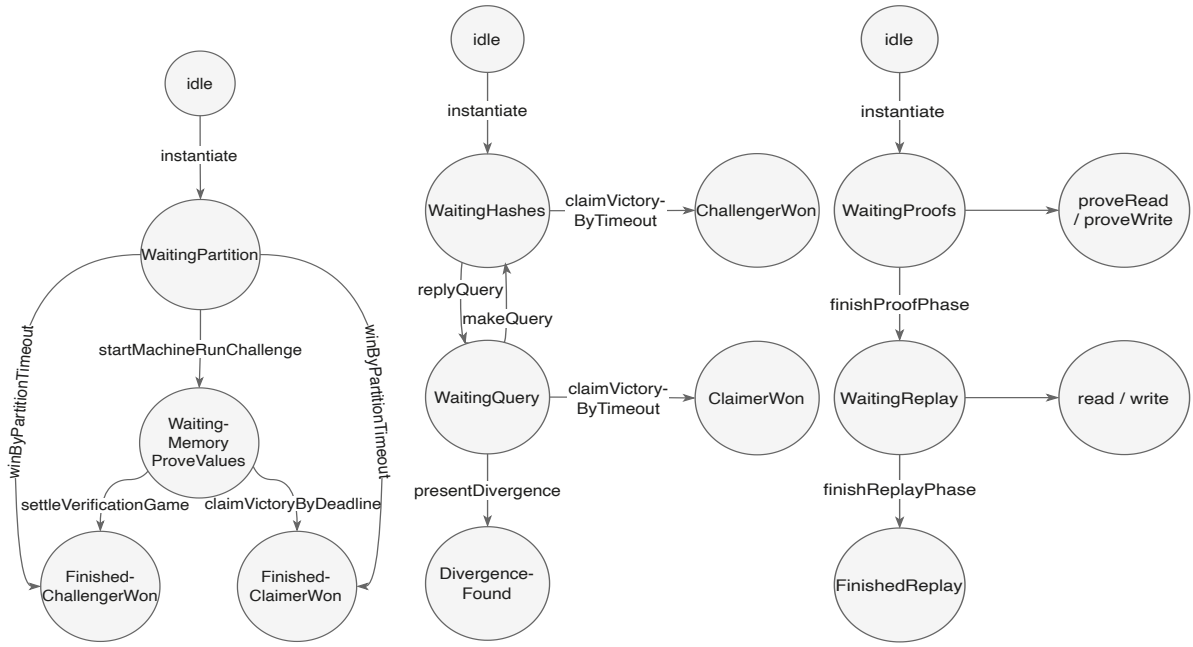


Figure 9: Descartes State Machine.

calculation, the claimer claims the final hash of the machine and the content of the output drive, through `submitClaim` function. The next state is *WaitingConfirmationDeadline* where the challengers nodes have a time window to either accept or challenge the claim. In case the challenger accepts the given result, the final state is *ConsensusResult* and the computation is finished since there was consensus.

In case of challenge, Figure 10, where a challenger disputes the claim, a Verification Game is triggered. When there is a Verification Game, the state is *WaitingChallengeResult*, waiting for the result of the entire dispute resolution mechanism. For each phase of the Verification Game, there are different states until the winner of the dispute is reached. As detailed in subsection 3.1.3 (Verification Game), the Partition contract finds the divergence point between claimer and challengers nodes, the exact step in which they agree with the initial hash but disagree with the final one, as presented in Figure 10(a). Partition contract starts with the `startMachineRunChallenge` function call. In order to find the disagreement point, the challenger makes queries and the claimer replies to them with the hashes of different computation stages, as can be seen in Figure 10(b). If the queries were successfully provided, the next phase is the Memory Manager, presented in Figure 10(c), in which the claimer sends the data related to all the activities performed in order to `instantiate` the on-chain emulator. This is the last stage of the Verification Game, the Step, in which the data from the Memory Manager stage is provided to the RISC-V emulator to execute the

computation at the divergence point. By checking the final hash obtained by the emulator, the winner is found, the DApp is notified of the final result, and the dishonest party is punished. The process of selecting nodes in charge of managing claims and challenges will be explained in the following section section 3.2.



(a) States of first stage of Partition. (b) States of second stage of Partition. (c) States of Memory Manager.

Figure 10: Verification Game States.

3.2 Blockchain Aspects for Cartesi

Despite the presented architecture, each DApp can present a different architecture, depending on the developers' requirements. In case of Cartesi DApps, each DApp can be developed with distinct trust models and this may, for example, imply more or less decentralization. The topology of a DApp is chosen by the developers according to the need for decentralization, the way users interact with it and the anonymity of validators nodes. The consensus is another blockchain idiosyncrasy that is connected with the decentralization of the application. On the blockchain, the consensus mechanism assures that all the smart contracts' instructions are performed by the nodes in the network to determine which transactions are valid. Computationally-intensive DApps are highly expensive due to the transaction fees and then, as already explained in subsection 2.1.4, layer-2 kind of solutions have emerged.

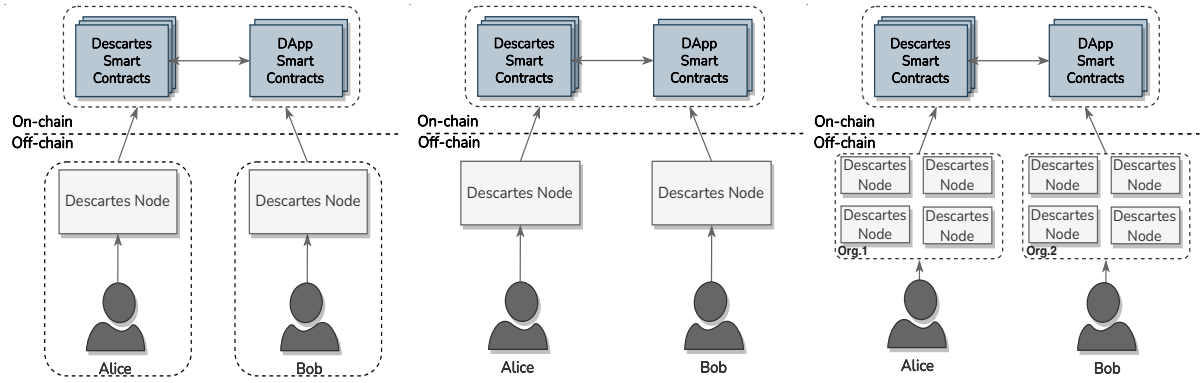
With Cartesi, instead of Global Consensus, DApps can achieve consensus on the results locally without direct contact with the blockchain. In Cartesi DApps, Local Consensus is used, and only one honest party is needed to enforce the correct outcome. In situations when a dispute arises, blockchain is used as a "supreme court" to get the correct result. Hence, a developer must be careful to ensure the proper Local Consensus structure, given the idiosyncrasies mentioned above, mainly in terms of decentralization. A

DApp developer is empowered to choose the nodes of a given DApp and the three possible approaches are: i) the involved users running their own nodes, ii) the users choose nodes they trust, iii) the users trust a committee of nodes run by trustable organizations or institutions, all those presented in Figure 11.

The first trust model, Figure 11(c), is fully decentralized but inconvenient for the users since they have to maintain a Descartes Node themselves by running the Docker containers and financing the node wallet to have funds for the transaction fees. In this type, an example of two players, Alice and Bob, playing a game, will also have to play the role of validators. It is a decentralized model because only the two involved in the game are responsible for its verification and truthfulness. If there is a claim by one of the players, the other will be responsible for verifying it and challenging it if a wrong claim is found.

The second, Figure 11(b), depends on the user's choice and the amount of decentralization depends on this same choice. The user can either specifically choose nodes that he/she knows or else select nodes from organizations or nodes with a good reputation. One of the optional factors in the selection is also the number of nodes.

The last one Figure 11(a), is the most centralized as it entrusts the verification of DApps to trusted entities. In this case, security and/or trust is chosen over decentralization. The choice of nodes is based on reputation, availability or any other criteria relevant to the DApp context. Despite being more centralized, this option is consequently simpler and more efficient in terms of computational capabilities and execution time.



(a) Users running their own nodes. (b) Users choose nodes they trust. (c) Users trust a committee of nodes.

Figure 11: Topology of Nodes.

Whatever the developer's choice, it is important to determine the right trade-offs specific to each application considering not only these idiosyncrasies but also limitations, mostly in terms of security. In either setup, a poor choice of node topology can undermine the security of a system. Cartesi DApps aren't scalable to large numbers of validator nodes, as they can suffer from multiple disagreements to exhaust the network resources or in case of a committee setup, where nodes are running on centralized server such as AWS or Google, face attacks on host system vulnerabilities. These attack vectors will be explored in the following chapter.

Chapter 4

Cartesi Security Analysis

In this chapter, we will present an analysis on the vulnerabilities and potential security threats of the Cartesi system. Initially, we will present the Cartesi threat model, with all trusted entities in the system being covered. We will discuss a set of flaws in the Cartesi protocol, which lead to the unveiling of undisclosed vulnerabilities in the DApp. We also present threats arising from a malicious host that can exploit the flaws in the protocol to manipulate a DApp. All these vulnerabilities will be presented according to the components they threaten in terms of confidentiality, integrity, and availability. Finally, this chapter ends with the presentation of a real attack mounted on a Cartesi DApp which exploits the identified vulnerabilities.

4.1 Cartesi Threat Model

In the light of DApps architecture, presented in subsection 3.1.3, we will analyze the security of each Descartes component. As can be observed in green in Figure 12, all off-chain components: the Descartes Nodes components, the front-end services, and the host are potential sources of vulnerabilities and are thus a threat. We will consider only the off-chain components, assuming that the on-chain components are secured by the blockchain.

Besides the components themselves, the communication between them can also be intercepted for malicious purposes. Therefore, we will disclose below the threats found in the system. These threats can arise due to flaws in the Cartesi protocol itself, or from a malicious host running a Descartes Node. Depending on the severity of the threats, they can have a greater impact on the security of the system and especially on the risk to the user.

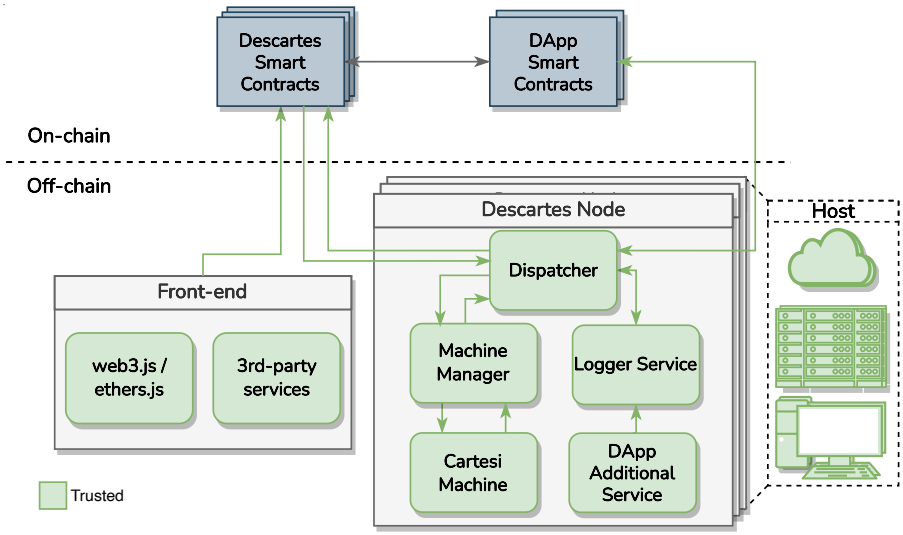


Figure 12: Cartesi DApp Threat Model.

4.2 Cartesi Protocol Flaws

Despite having a blockchain providing a layer of security to DApps, Cartesi's architecture features a large attack surface. There are several flaws in Cartesi protocol. Starting with the topology of nodes, as mentioned in section 3.2, Cartesi is not scalable in terms of number of nodes. This problem arises due to the mechanism of computation validation and dispute resolution.

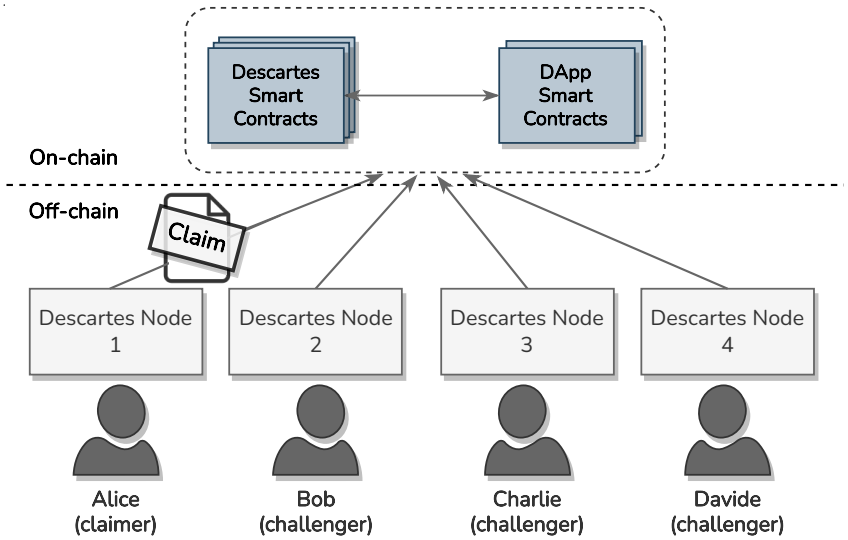
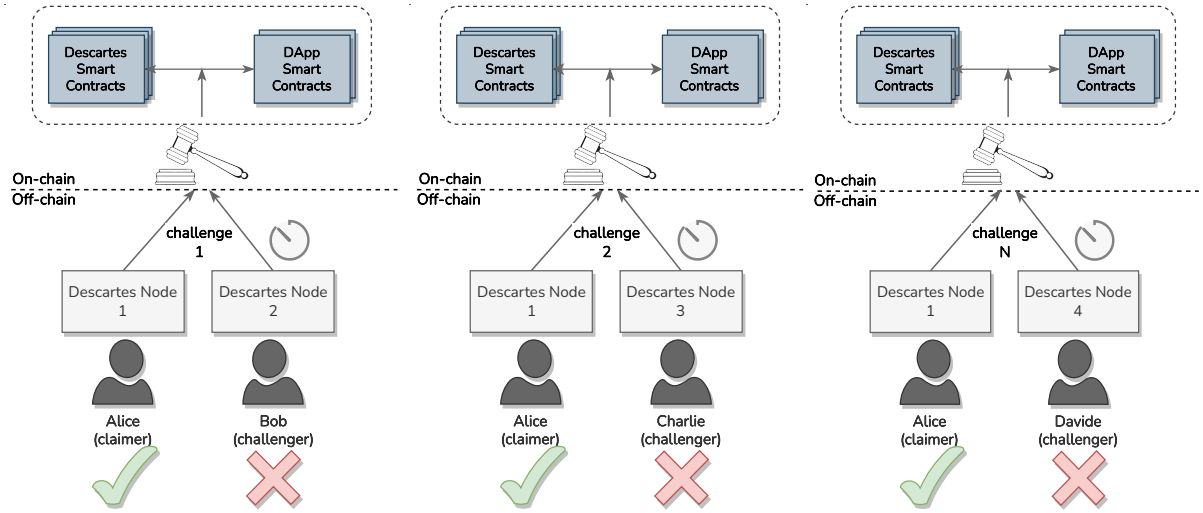


Figure 13: Claimer Node Claiming a DApp Computation Result.

Verification Game triggered continuously

At the end of a Descartes computation, it is expected to have a single node be the claimer and all the remaining ones will be challengers that will guarantee the veracity of the result, as depicted in Figure 13.

In case of a dispute, the challenger node will trigger a Verification Game against the claimer, as presented in Figure 14(a). Each Verification Game has a duration for the whole process to take place, to submit all the necessary data, and to find out the truth result. Regardless of the decision, the result will be a new claim, with a new challenging period. This may generate a new challenge, Figure 14(b), if other challenger node disagree with the current claim. Then, it will take another Verification Game period, and at the end there could be yet another challenging period, Figure 14(c), and this can be repeated infinite times. With this, the total time to finish a computation will be equal to the challenging periods and Verification Game period multiplied by the eventual number of challenging nodes, thus possibly delaying the computations.



(a) First challenge. (b) Second challenge. (c) N challenge.

Figure 14: Multiple Challenges for a Given Claim.

Descartes Nodes Availability

Another threat is related to Local Consensus, although it only requires at least one node to validate a result, it is also necessary to ensure that there is at least one interested party participating in the validation. In case an attacker compromises the nodes involved, Figure 15, they can disrupt them or make them disappear and undermine the availability of the system. Even if a DApp has two or more nodes involved in the validation, it may happen that not all nodes are correctly connected with a stable internet connection, or they may be engaged in other tasks that do not allow them to validate a given computation. Thus, the claimer node submits its result, which may or not be the correct one, but the other nodes are busy or disconnected and therefore do not dispute, and the claimer wins by reaching the deadline. This can be used by attackers who compromise the availability of nodes so that they do not check a fraudulent claim and enable it to be published on-chain.

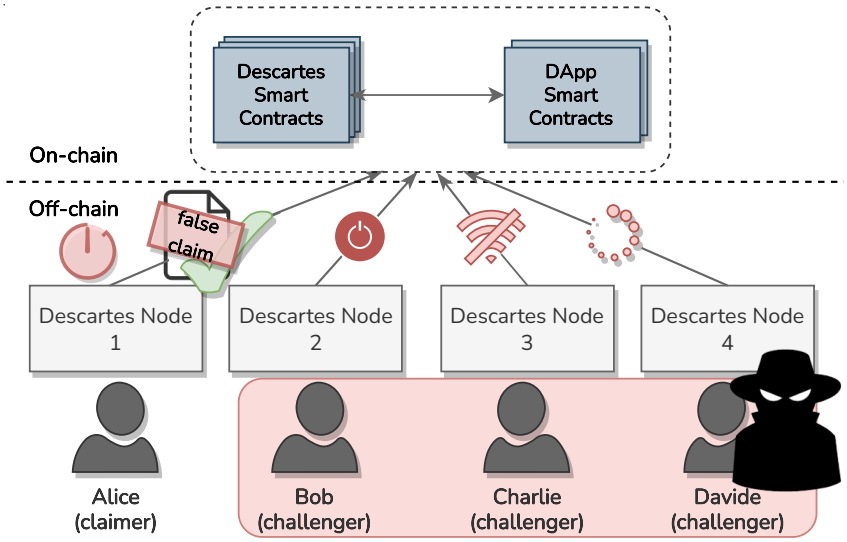


Figure 15: Compromised Availability of Nodes.

Dispute Resolution as Security Mechanism

Verification Game is used to settle disputes, since only one node is needed to validate an honest computation, but it is not tamperproof in all situations, and can be a threat to the integrity of the system. An adversary can compromise all nodes of a given DApp, as shown in Figure 16, and thereby enforce a false result on the blockchain, since none of the nodes will contest the claimer's result. If all nodes suffer from the attack, they will all be injected with the same wrong result, therefore there will be no divergence between them, so it will not start a Verification Game. All Merkle tree cryptographic proofs of the Cartesi Machine state will normally be submitted, and the claim will be accepted with no challenges between validators.

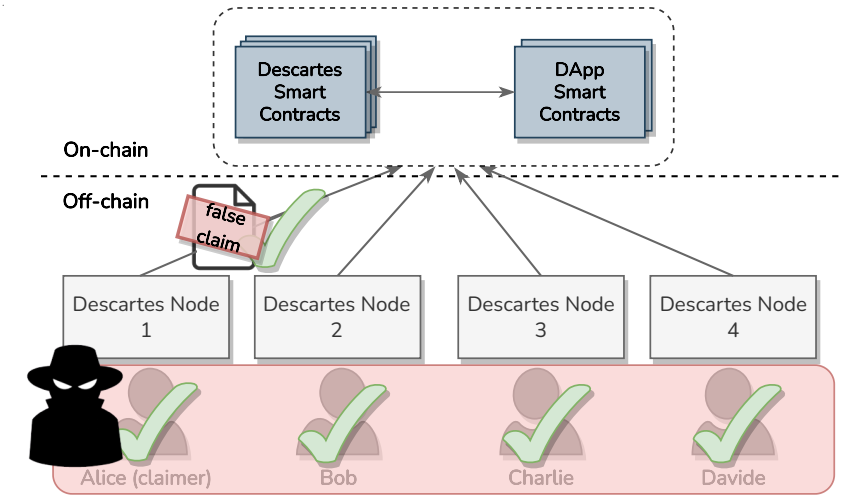
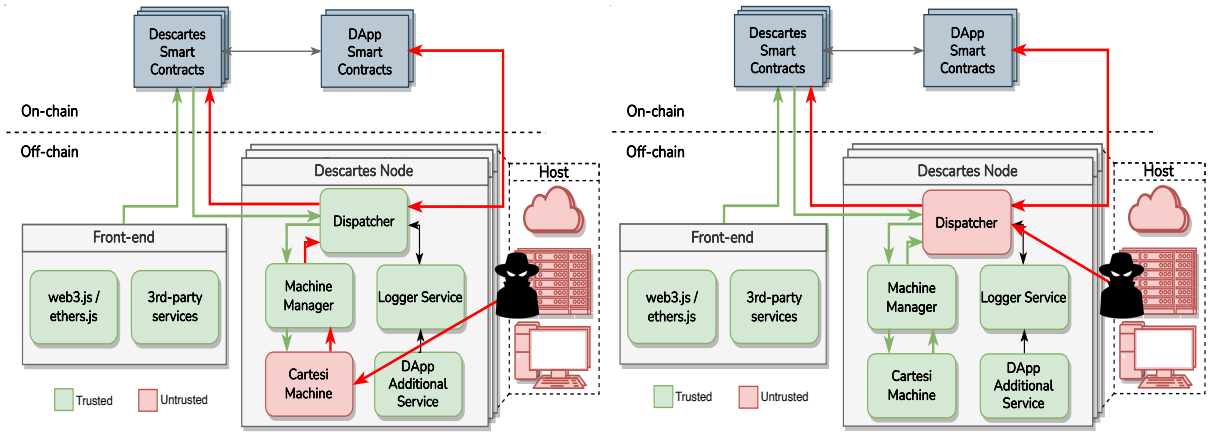


Figure 16: An Adversary Compromising All Nodes.

4.3 Host Security Threats

In addition to protocol flaws, there are problems associated with Descartes Nodes running on a host system. A malicious host can compromise the system by several attack vectors. The node infrastructure can allow attackers to gain control of the host and manipulate Descartes Node services. An infrastructure running in the cloud could be vulnerable and allow attackers to gain control over the nodes. An attack on a single node can spread its effects to other nodes sharing the same infrastructure. Since the configuration and settings of the services are entrusted to cloud providers, it could lead to attacks on the confidentiality and/or integrity of the system provided by the cloud, affecting nodes without the parties involved being aware of it. In contrast, nodes running on the same physical hardware can also be a serious threat. Although each Descartes Node has its own address and images for each component, they are not properly isolated, enabling hardware attacks that causes, for example, a data breach. These host security threats will allow attackers to exploit the protocol flaws mentioned above.

As already explained in the previous chapter, Descartes Node consists of several Docker images that contain all the executables and dependencies for the different Descartes components.These images are read-only, as a kind of snapshot of the service, which cannot be changed. Therefore, at the Descartes Node's boot, all the images are used as a template base to build a Docker container for each DApp component. The containers run the images and add a writable layer over them to make them interactive and thus modifiable. Despite this, the images continue to exist separately and are immutable, only a read-write copy of the image is created in the container. This interaction is a way to allow the host to manipulate the Descartes components and can be used as intended or maliciously. Since the containers are running on each node's host, it makes it possible for each node to tamper with the content or code that is crucial to the normal execution of a DApp. Descartes Nodes components run in different containers, and therefore it is possible to compromise any of them, affecting the entire DApp system, since the components are not securely isolated from the host and communicate with each other.



(a) Cartesi Machine container attack.

(b) Dispatcher container attack.

Figure 17: Possible Attacks on a Cartesi DApp.

A host adversary manipulating the Cartesi Machine container, Figure 17(a), could modify the contents of the drives associated with a given computation and thus return a false output to the blockchain. An attack on Dispatcher, Figure 17(b), the core component of the Descartes Node, could also affect the entire system, given that it is the component in charge of communication between all services on the node and also with the blockchain. Even though the Cartesi Machine performs the computation correctly, the Dispatcher may be hacked to modify the output received from the Machine Manager.

As mentioned in section 3.2, the choice of the trust model will have a bearing on these security threats. The damage from a malicious host is dependent on the choice of node topology. By choosing an institutional committee model explained in section 3.2, an attack on the institution could affect all nodes, making it impossible to detect a fraudulent claim by Cartesi security mechanisms. Attackers can also perform a DoS attack on the host, congesting the node or even disconnecting it, causing a malicious claim to win by time out. Cartesi assures that off-chain DApps have the same security guarantees on-chain, since the blockchain acts as supreme court. However, this requires nodes to trigger a Verification Game when disagreeing with a claim. When a host compromises the entire infrastructure of the nodes, the Verification Game becomes useless, because this is simply a dispute resolution mechanism and not a security one. Therefore, in cases where host attacks all nodes, where all are injected with the same fraudulent result, or where any of the nodes are overloaded and become unavailable, no node will trigger a dispute. The nodes may reach consensus for the wrong result, or they may be congested and not trigger the dispute, causing the malicious claimer to win by missing the deadline, and in these cases fraudulent results are published on the blockchain and can have large monetary implications for the victims. The following section will present a real attack scenario that exploits several vectors mentioned above.

4.4 Cartesi Attack Demo

With the knowledge of the main threats to Cartesi we mounted a proof of concept attack on a Cartesi DApp, presented in Figure 18. This attack was performed in a Calculator DApp example and using the Descartes SDK 1.0 and Descartes SDK Environment 1.1.0 with all the on-chain and off-chain components of a Descartes DApp. Descartes' smart contracts use Solidity 0.7.0 and will not work with lower version compilers. The Descartes environment also provides a local testnet blockchain using Hardhat 2.1.2. Hardhat comes built-in with Hardhat Network, a local Ethereum network designed for development and testing. It is backed by the Ethereum Virtual Machine (EVM) implementation and works in the same way as the mainnet. Furthermore, Hardhat provides 20 accounts with 10000 ETH test tokens each. The first two nodes of the 20 are used to conduct the computations and verifications, named Alice and Bob, with Alice as the claimer. Although it involves only 2 nodes, this attack would work the same way regardless of the number of nodes. The goal of this attack is to force a wrong result of an operation requested by a user to be written to the blockchain. To achieve this goal, it was required to compromise the nodes so that they would not dispute the (fraudulent) claim submitted by the claimer.

In a normal process of a Cartesi DApp the nodes are supposed to detect any cheating in the computations and dispute the result, automatically initiating a Verification Game to resolve the challenge. However, the enumerated flaws in the architecture and threats associated with the host allow an attacker to manipulate the data without being detected by Cartesi's security mechanisms.

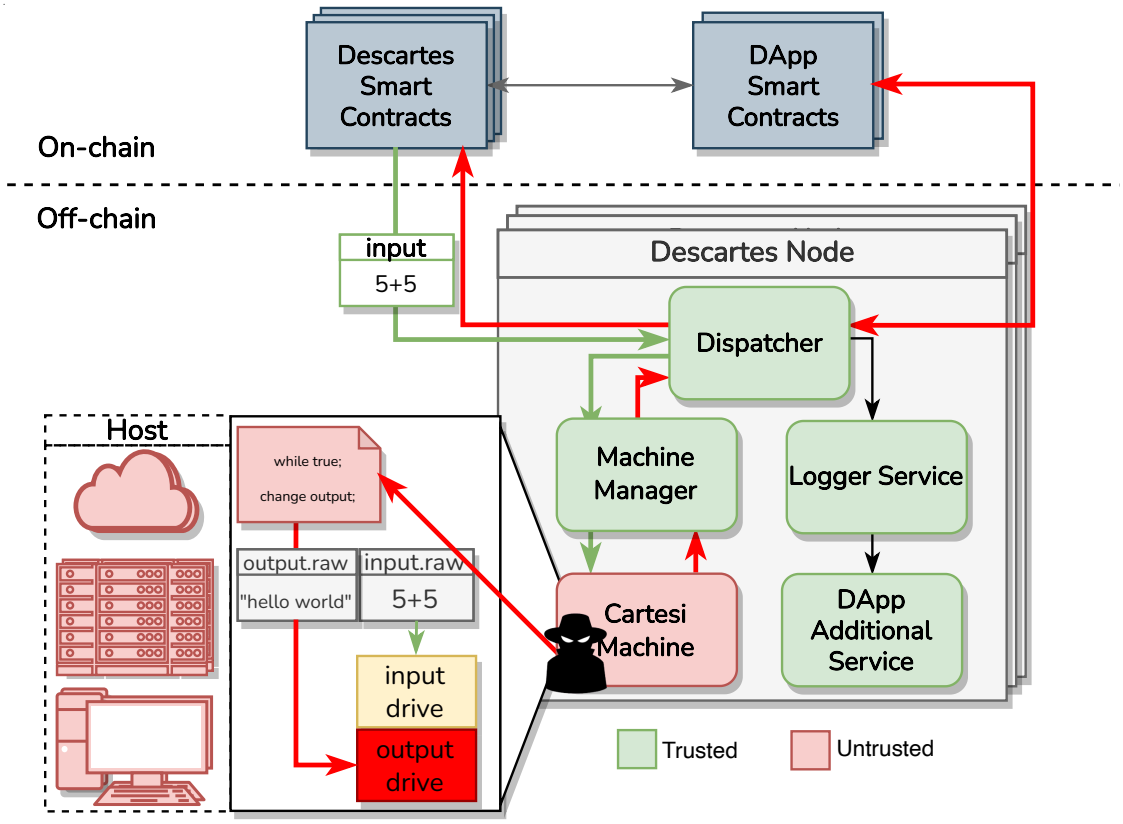


Figure 18: Cartesi DApp Real Attack Scenario.

This attack exploits the host security threats as well as the topology of nodes. Once the nodes have been compromised, they will accept a fraudulent output as the final result. In addition, it overrides the Verification Game which, by not being triggered, jeopardizes the DApp security. In a typical scenario and with honest actors, the user indicates the calculation he wants to perform, and this calculation is passed to the `instantiate` function of Descartes smart contract, to be inserted into the Cartesi Machine's input drives through the Machine Manager.

For this DApp, a Cartesi Machine template was specifically built to perform the off-chain computation of the received operation. Since the Cartesi Machine leverages a Linux OS, the bc tool is used to compute the result of a mathematical expression in string format. Given an input, the machine is programmed to read the drive using the dd tool, pipe it through a Lua script to ensure it is a null-terminated string, pass the string to the bc tool, and finally write the result to the output drive. These template machine configurations are presented below in Listing 4.1.

```
1 docker run \
2   -e USER=$(id -u -n) \
```

```
  -e GROUP=$(id -g -n) \
  -e UID=$(id -u) \
  -e GID=$(id -g) \
  -v `pwd`:/home/$(id -u -n) \
  -w /home/$(id -u -n) \
  --rm $CARTESI_PLAYGROUND_DOCKER cartesi-machine \
    --max-mcycle=0 \
    --initial-hash \
    --store="$MACHINE_TEMP_DIR" \
    --flash-drive="label:input,length:1<<12" \
    --flash-drive="label:output,length:1<<12" \
    -- $'dd status=none if=$(flashdrive input) | lua -e \'print((string.
        unpack("z", io.read("a"))))\' | bc | dd status=none of=$(flashdrive
         output)'
```

Listing 4.1: Cartesi Machine Template for Calculator DApp.

This output value is then forwarded by the Machine Manager to the Dispatcher that will process the information to be submitted to the blockchain as a transaction, with gas price optimization and published in a timely manner. The output result is claimed via `submitClaim` function and the state changes to *WaitingConfirmationDeadline*, waiting for confirmation from the other nodes, that can either be a challenge or consensus. Since the scenario under analysis relies on honest players, in a short time the result will be validated and the final state will be *ConsensusResult*, with the correct result being stored in the chain.

```
alice_machine_manager:
  image: cartesi/machine-manager:0.5.0
  volumes:
    - ./machines:/opt/cartesi/srv/descartes/cartesi-machine
    - ./alice_data:/opt/cartesi/srv/descartes/flashdrive
  networks:
    ethereum: {}
    alice:
      aliases:
        - machine-manager

bob_machine_manager:
  image: cartesi/machine-manager:0.5.0
  volumes:
    - ./machines:/opt/cartesi/srv/descartes/cartesi-machine
    - ./bob_data:/opt/cartesi/srv/descartes/flashdrive
  networks:
    ethereum: {}
    bob:
      aliases:
        - machine-manager
```

Listing 4.2: Machine Manager Configuration in the compose.yml File.

Before presenting the attack, it is important to understand what node configurations used in the attack scenario. The node configuration is established using the docker-compose.yml file. This file defines the set of services that will run simultaneously to create the Descartes Node environment, the network, and MNEMONIC of the address used are initially defined, and then, the profiles that will represent the services related to each node, in this case Alice and Bob.

As can be seen in Listing 4.2, the configuration of the Machine Manager service for each node includes the Cartesi Machine Manager images, 2 volumes are mounted: machines and bob_data, and finally the operation network which is set by default. The volumes property denotes directories of the host where the node runs, which contain the stored template for the Cartesi Machine, as well as the raw input and output files of the machine. After setting up, the node can be booted and is ready to receive computations from DApp. The process will occur in a similar way to the honest scenario mentioned above.

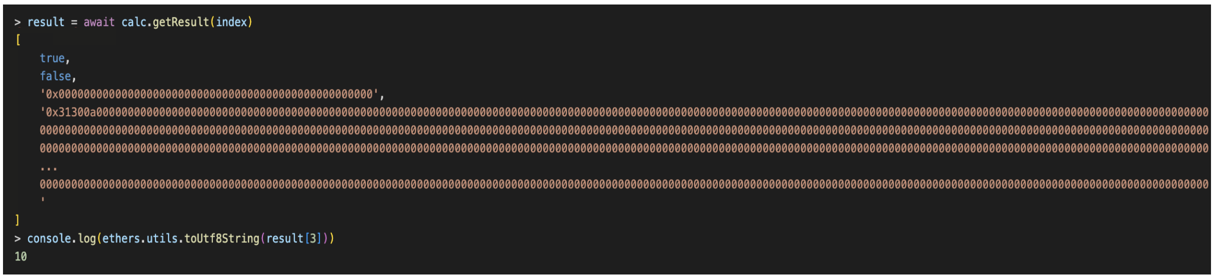


Figure 19: Hardhat Console with Results of the Honest Computation.

The computation is requested by using the Descartes' `instantiate` function, the input of the operation will be provided to the machine, the example used being "5+5" and the result computed by the Cartesi Machine will be "10", as expected and proven in Figure 19. Position 3 of the result array has the result, where "0x3130" is the hexadecimal representation of the string "10".

```
while true;
# 'hello world' string in ASCII
do echo $'\x68\x65\x6C\x6C\x6F\x20\x77\x6F\x72\x6C\x64'|
# output drive memory page
dd of=a000000000000000-1000.bin |
truncate -s 4K a000000000000000-1000.bin;
done
```

Listing 4.3: Attack Script.

However, in the context of this attack, a script was run on the host that automatically changes the contents of the output raw file to a different value than the calculated one, in this example "hello world". This script, presented in Listing 4.3, enters a string "hello world" into the file `a000000000000000000-1000-.bin` that represents the memory page where the output drive is located and pads the file with zeros with truncate tool, to fulfill the machine's requirements. Whatever computation is requested, this script will tamper with the output and thus send to the blockchain a different result than expected. Once

the time is up for the remaining nodes to challenge the claim, the result will be accepted by consensus and the computation will be terminated, since the attack affects the entire infrastructure of Descartes Nodes.

Figure 20: DApp Dispatcher Log File.

In an honest computation, the claim is sent via `submitClaim` function, the state changes from *WaitingClaim* to *WaitingConfirmationDeadline*, without a challenge submitted before the deadline, the waiting period is cancelled by using `abortByDeadline` function that leads to the final state of *ConsensusResult*. In the attack scenario, everything happens the same way, but the output read from memory has been changed. The Figure 20 shows these state changes, as well as the content of the Cartesi Machine memory and the claimed output. Contrary to expectations ("10"), the claimed output present in Figure 21, contains the fraudulent result ("hello world"). The content of the claimed output has the decimal representation of the string "hello world".

Figure 21: Content of Fraudulent Claimed Output.

By starting a Hardhat console, it is possible to interact with the smart contract via JavaScript to query the DApp for the final result via the `getResult` method of Descartes, Figure 22. The first boolean value indicates that the result is already available, the second shows that the computation has finished, the third

value indicates that no address (node) has disputed the result and finally presents the final result. Decoding the hexadecimal result "0x68656c6c6f20776f726c64" retrieves the string "hello world". This result is wrong, it should be "10", but as can be verified, this result was accepted and returned by `getResult` as a "true" result.

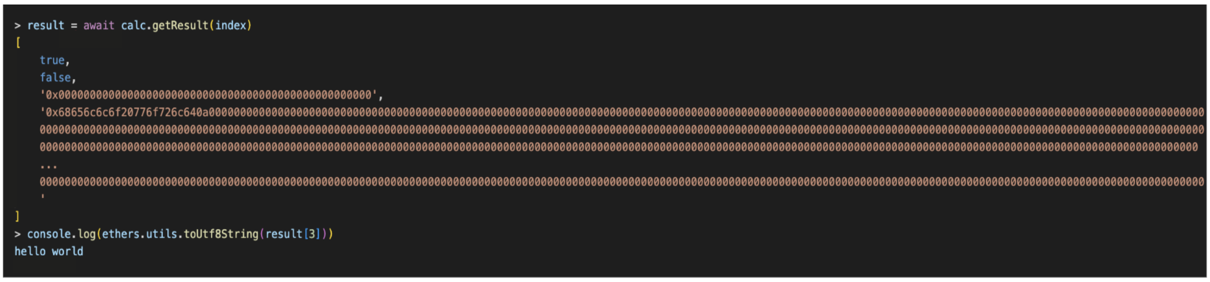


Figure 22: Hardhat Console for the Malicious Computation.

This attack could seriously affect a real blockchain DApp, involving large amounts of money or private operations concerning a user. Since one of the great advantages of the blockchain is to keep data stored and immutable, layer-2 protocols should ensure the same security considering all the external factors it is subject to off-chain. However, as discussed in chapter 2, the scalability trilemma remains a challenge for the Web3 area, with protocols that aim to increase scalability threatening the security and/or decentralization of the applications.

Chapter 5

Cartesi's Threat Mitigation

Given the threats and vulnerabilities presented in the previous chapter, we next discuss possible mitigation solutions based on TEEs. Afterwards, we also present various solutions using TEEs in containers. Finally, we will present a solution as a proof of concept that adds an authentication layer to Cartesi based on digital signatures, which solves several attacks like the one mentioned in chapter 4.

5.1 Threat Mitigation - Overview

The threats presented in the previous chapter, as well as the attack performed, show that most of the problems are associated with a malicious host, e.g., a DApp can be fully compromised and controlled mainly due to host threats. This was proven in the attack scenario, section 4.4, which showed that a malicious host accessed the contents of the drives and was able to change them. The threats and flaws of Cartesi DApps architecture allow an attacker to accomplish this, mainly due to Docker's threat model assuming the host is trusted. Cartesi introduces Docker containers as a lightweight solution for facilitating the boot and maintenance of a Descartes Node. Each Descartes component, presented in Figure 23, fits into a Docker container and communicates with the others in order to run all the processes needed to run a DApp.

However, although containers speed up deployments of applications and ease the distribution and delivery of software, they are not isolated environments from the host. They have a large attack surface and are often only lightly protected by software isolation mechanisms. These mechanisms protect only containers from being accessed by untrusted containers, but not from being accessed by high privilege system components. In addition, containers do not run their own OS, they use the host OS to operate, thus introducing vulnerabilities that an attacker can exploit to penetrate the applications. This can lead to mounting sensitive host directories into containers, vulnerabilities in the image distribution or inside

the image, as it can enable the attacker to cause Bash to execute arbitrary commands and also lead to vulnerabilities of the Linux kernel.

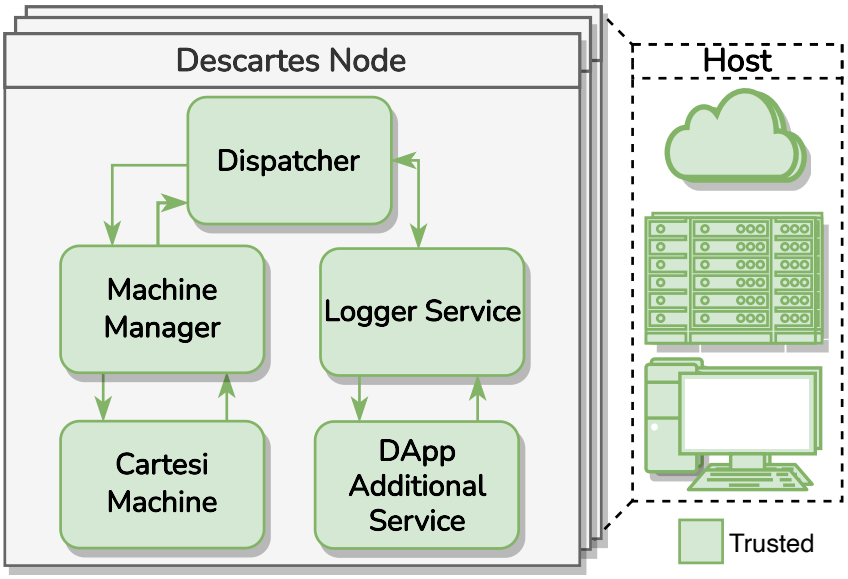


Figure 23: Descartes Node Docker Containers.

Cartesi DApps are dependent on containers to run the Descartes Nodes environment and are therefore subject to all of its vulnerabilities. The Dispatcher container communicates with all Descartes containers and therefore, if compromised, could compromise the whole system, mainly by maliciously manipulating off-chain and on-chain interactions and by requesting computations from the Cartesi Machine. An attack on the Machine Manager or Cartesi Machine containers can affect the integrity of a DApp, as already shown in the attack in section 4.4. In this case, mounting host directories on the container makes them easily accessible by a malicious host. From the Logger container it is also possible to modify computation data, which in large quantity is split by the Logger service to be lightly presented on-chain. The DApp Additional Service can also bring threats to the system, depending on the type of external service used by the developer.

Due to these problems, this thesis suggests using TEEs to run the containers in a secure environment. Confidential containers are emerging with capabilities to secure container workloads to achieve greater security. Containers leveraged by TEEs provide strong assurances of data confidentiality, code, and data integrity and helps isolate containers not only from other containers but also from untrusted parties and applications. Containers can run within enclave-based TEEs achieving high-isolation and memory encryption through hardware security guarantees. As can be observed in Figure 24, containers running in TEEs (3 and 4) are protected from vulnerabilities of malicious hosts and from privilege attacks on host OSs, in contrast with the normal containers (1 and 2).

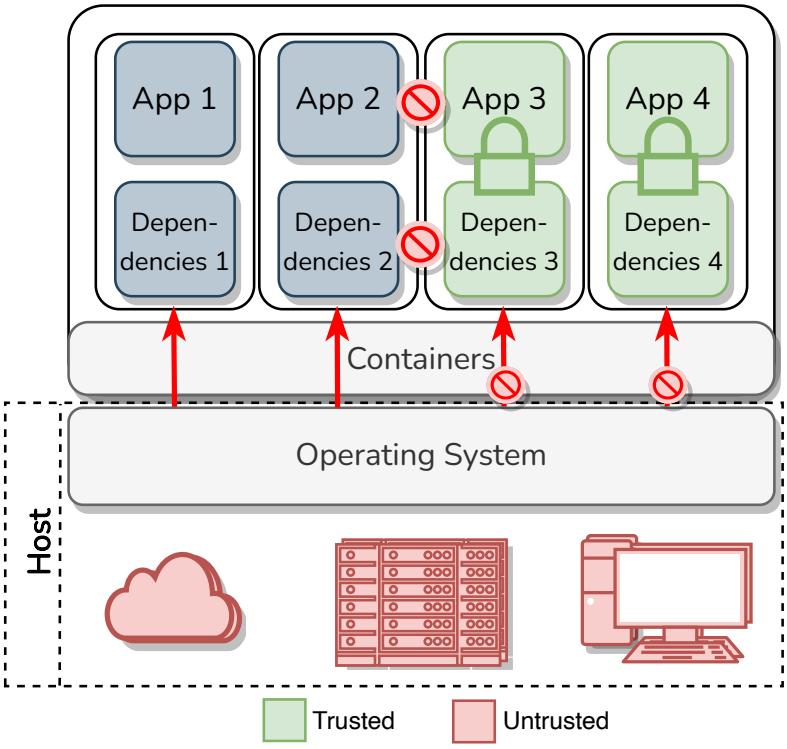


Figure 24: Security of Normal Containers Versus TEE Containers.

5.2 Analysis of TEE Containers Solutions

There are several implementations for running containers on different TEEs such as SCONE [107], KubeTEE [119] and Graphene-SGX [120] for running on Intel SGX and TZ-containers [121] leveraging Arm TrustZone. Cloud Native Computing Foundation [122] is working on a confidential container solution that supports various TEE technologies such as AMD Secure Encrypted Virtualization and Intel SGX.

SCONE presents a secure mechanism for Docker containers that uses Intel SGX to protect containers processes against outside attacks by running the user-level part in an enclave. This secure containers are able to compile unmodified source code into an enclave application binary using an SGX-aware musl-libc and/or run unmodified Alpine Linux binary. SCONE has the advantage of provides encryption not only for files, but also for input parameters and environment variables. KubeTEE is a collection of TEE development, deployment, maintenance middleware frameworks, and services. Leveraging SGX, its goal is to allow developers to implement TEE-based workloads of Docker containers, Kubernetes orchestration, and other cloud-native technologies. Graphene-SGX also leverage Intel SGX to protect containers providing features such as full SGX Attestation support, protected files support, multi-process support with encrypted IPC, and support for the upstreamed SGX driver for Linux. TZ-Container leverages Arm TrustZone to protect containers in an isolated execution environment. These containers bring security for all interactions between processes and the kernel and manage the integrity of user access.

Figure 25 presents the architecture of SCONE as an example of a Docker-compatible solution for Cartesi, where it can be observed that each Descartes container has its trusted part in an enclave, isolated

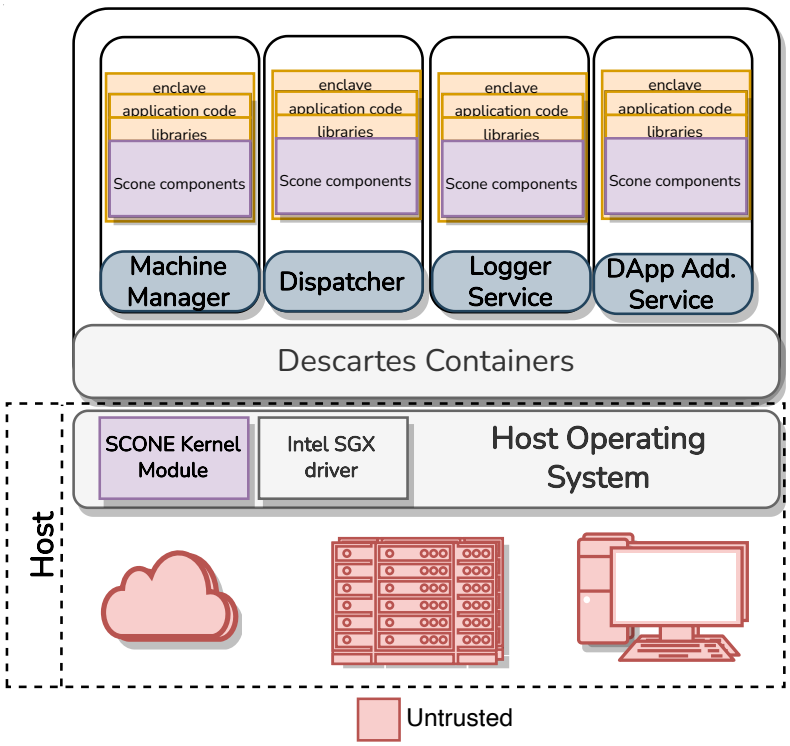


Figure 25: SCONE Architecture. Adapted from [107]

from all others. The host OS must include an SGX Linux driver and a SCONE kernel module to increase performance. It only uses the Linux SGX driver from the Intel Linux SDK. This architecture of SCONE provides secure containers on top of an untrusted OS, protected from various threats from adversaries with superuser access to either the system or physical hardware. Furthermore, secure containers fit transparently into existing Docker containers, making it possible to build secure container images with the help of Docker in a trusted environment and run secure containers in an untrusted environment. These solutions enable increasing Cartesi security by mitigating the vulnerabilities associated with Descartes Nodes components based on Docker containers.

5.3 Solution based on Digital Signatures - Proof of Concept

As mitigation of the problems discussed throughout this thesis, mainly the threats exploited in the attack scenario, we propose a solution based on digital signatures for Cartesi's DApps. This solution adds an authentication layer to DApps, increasing system security against attacks from malicious hosts that modify the contents of the computations requested from Descartes and Cartesi Machine. In the Figure 26 we represented the model for a DApp with 2 nodes involved, a claimer, and a challenger. In this proof of concept, no TEE mechanisms are used, the modifications are made only at the level of the Cartesi Machine image and the smart contracts. As already explained in the course of the chapters, after the instantiation of a computation, the node defined as claimer executes the computation and claims the

result. The inputs of the requested computation are passed to the input drive of the Cartesi Machine in order to perform all the necessary actions to reach the final result. In the case of the attack performed, the input was a computation of "5+5" which would be calculated by the Cartesi Machine and would result in "10". However, apart from the remaining validator nodes, there is no guarantee that the calculation was correctly performed. If a result other than "10" is presented and no node disputes that result, it will be considered valid.

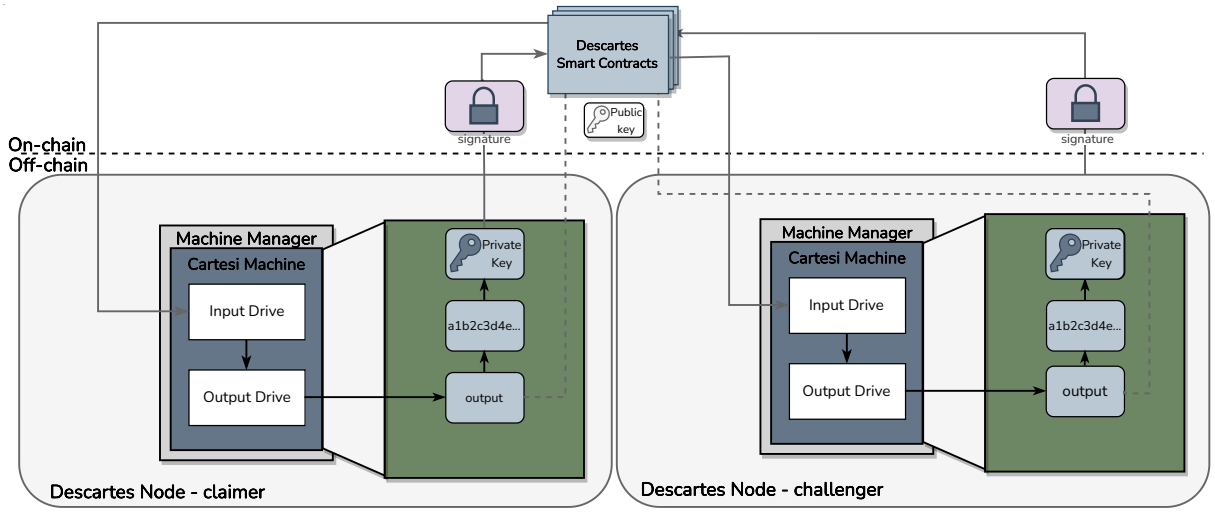


Figure 26: Cartesi DApp Threat Mitigation.

Therefore, the proposed solution proposes the use of digital signatures, for signing these outputs on the Cartesi Machine. The purpose of digital signatures is to sign the result to authenticate that it was made by a trusted party. It requires the use of keypairs: a public- and a private-key, to validate the authenticity and integrity of data. In this mechanism, the keypairs used have a different meaning from other encryption mechanisms, since the private-key is used as the signature key to sign the data and the public-key as the verification key. Digital signatures can provide data integrity and non-repudiation, as it is possible to ensure that received data is the same as originally sent and that it has not been tampered with or intercepted by a third party [123]. In addition, it provides authentication by enabling the identification of the sender of the data. A digest is a fixed size representation of the data, calculated by a hash function. An encoded digest forms a digital signature. The signatures are cryptographic values calculated from the data to be verified, and the private-key used should only be managed by the signer. The private-key is a cryptographic string, from which two values are extracted to create the public-key: exponent and modulus. Although the public-key is extracted from the private-key, the private-key cannot be obtained from the public-key.

To digitally sign data, first a hash of the original data is computed, and then it is encrypted using the sender's private-key. After that, the original data and the signature are sent to the receiver to decrypt the signature. For the receiver to verify the signature, a hash of the received data is once again calculated and the signature is decrypted with the public-key. If both digests match, the signature is valid, meaning that the result was sent from a trusted party.

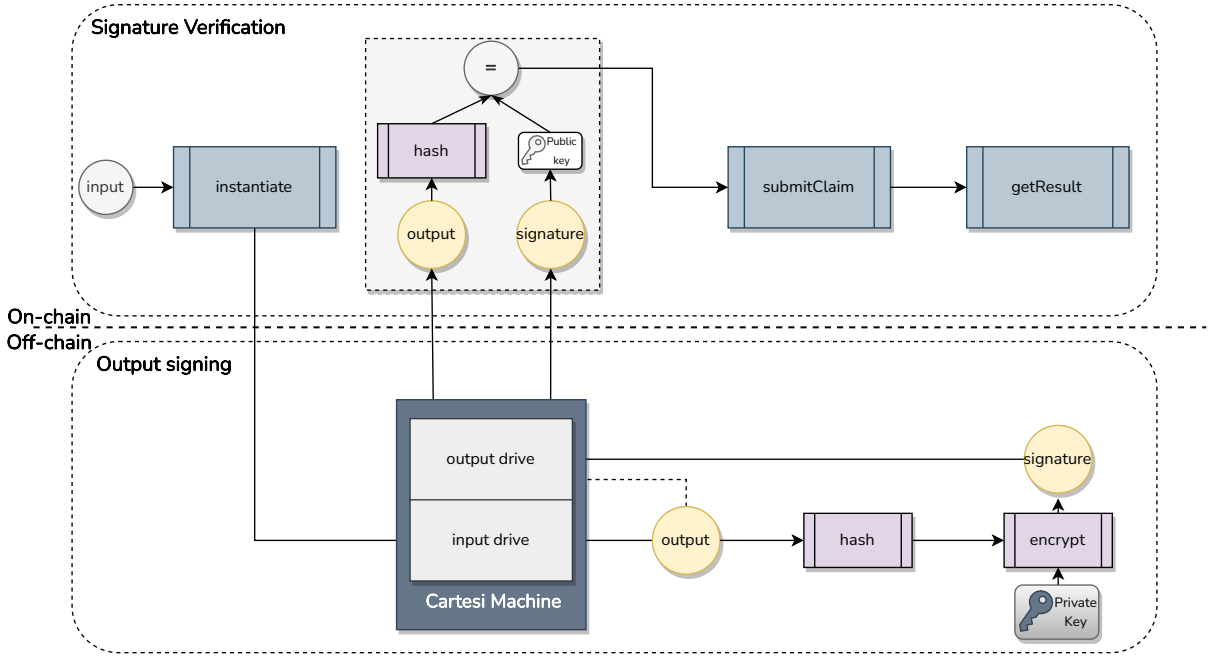


Figure 27: Output Signing and Verification Flow.

As shown in Figure 27, in a Cartesi DApp, an input is passed as a parameter in the `instantiate` method of the Descartes smart contract when initializing a computation. At the boot of an off-chain Cartesi Machine session, this input will populate the machine's input drive and the respective output will be calculated. In the case of the proposed solution, the resulting output will be passed through a hashing algorithm to obtain a hash or digest of the original output. Then, this hash will be encrypted by the private-key and will originate the digital signature of the output. The obtained signature and the original output will be sent to the blockchain to be verified by the digital signature verification algorithm. The original output will once again be transformed into a hash, and the public-key will decrypt the received signature. Finally, the hash obtained from the output and the hash of the signature will be compared to verify that the signature matches the received output. If it is valid, it will be submitted as the claim of the computation and returned to the user via the `getResult` method.

To generate the private-key, the *openssl* tool was used. The *genrsa* command was used to generate an RSA key with a size of 4096 bits. Subsequently, from the private-key the public-key is extracted also with an *openssl* command, as depicted in Listing 5.1. These keys are placed in the host directory that will be mounted on the Cartesi Machine so that they are already created at the moment of executing the computation. The distribution of public-keys is a developer trade-off between development flexibility and decentralization, according to the DApp trust model.

```
1 openssl genrsa -out key.pem 4096
2 openssl rsa -in key.pem -pubout > key.pub
```

Listing 5.1: Private- and Public-key Generation.

At that moment, as presented in Listing 5.2, the computation input is read and written to a file

`input.raw`, it is then truncated to comply with the machine's sizes and the computation is executed, in this case the calculator operation. Recurring again to the *openssl* tool, the computation result, `result.raw`, is hashed and signed with the private-key `key.pem` to generate the digital signature, `result.raw.sign`. Finally, the output and the signature are concatenated to be sent together to on-chain, through machine output drive.

```
dd status=none if=$(flashdrive input) > input.raw
truncate -s 4K input.raw
cat input.raw | bc > result.raw
openssl dgst -sha256 -sign /mnt/key.pem -out result.raw.sign result.raw
cat result.raw.sign | xxd -p | tr -d \\n > resultado.raw
cat result.raw resultado.raw | xxd -p | tr -d \\n > output.raw
truncate -s 4K output.raw
dd status=none if=output.raw of=$(flashdrive output)
```

Listing 5.2: Off-chain Output Signing.

On the on-chain side, it is necessary to split the output of the signature to perform the verification. Also, the public-key had to be previously split into exponent and modulus. The output data, signature, exponent, and modulus parameters are required to call the signature verification function, `signatureVerifyRaw`. This function, Listing 5.3, compares the hash sha256 `_output` of the output received from the Cartesi Machine with the digital signature `_s` and the public-key with the join of the exponent `_e` and modulus `_m`.

```
    /** @dev Verifies a SHA256 signature
      * @param _data to verify
      * @param _s is the signature
      * @param _e is the exponent
      * @param _m is the modulus
      * @return 0 if success, >0 otherwise
    */
    function signatureVerifyRaw(
        bytes memory _output,
        bytes memory _s, bytes memory _e, bytes memory _m
    ) public view returns (uint) {
        return signatureVerify(sha256(_output),_s,_e,_m);
    }
```

Listing 5.3: Signature Verification Function.

If the hashes match, the function will return 0 and the output can be submitted as a claim with the `submitClaim` function of the Descartes smart contract.

Cost Analysis

Handling cryptography and key management is computationally intensive for any system. On the blockchain, anything that is computationally intensive has a monetary cost. Implementing this signature verification slightly increases the gas value associated with DApp and Descartes smart contracts. The cost of a transaction on the Ethereum network is calculated from: `Cost = Gas Price x Amount of Gas Consumed`. The addition of gas to the deployed smart contract with signature verification is about 650500. This solution is not implemented with gas optimization, which means that it is still possible to lead to lower costs. As an example, considering Ethereum at $1300 per unit and 15 Gwei: `Cost = 650500 x 15 = 9757500 Gwei`. Wei is the smallest denomination of ether, like cents are to the U.S. dollar, but Gwei is one-billionth of one ETH. Therefore: *9757500 Gwei = 0.011112 ETH*, so `Cost = 0.011112 x 1300 = $14.45`. There will be an increase of about $15 on the contract deployment and some increase also for calling some functions. This represents a relative increase of about 65%. It is not a significant increase, but is justified by the additional security it brings to the system. It should also be noted that this implementation will significantly reduce the occurrence of Verification Games, a process that has a high cost when triggered. In our proof of concept, the check is done after the claim submission (via the `submitClaim` function). However, it is possible to do the verification before the claim is submitted, and it will also save some fees. If the verification finds an improperly signed result, the claim will not be submitted and thus economize some gas fees associated to this function call.

A Verification Game is triggered when at least one node detects that the claim is different from the one it obtained, probably by maliciousness of the claimer. Using signature verification, a wrongly submitted claim by fraud will not be accepted, because it will not be properly signed. Thus, the Verification Game will not be used to resolve frauds, only to dispute different results but with the proper signature. This is in a case where only one node is compromised. In these cases, the Verification Game is enough to detect a fraudulent claim, as already mentioned. But in attacks on the entire infrastructure of nodes, the Verification Game would not even be triggered, and the claims would pass unpunished. So this implementation of authentication proves to be an additional security feature for Cartesi DApps, overcoming the cost increase.

Another point to consider is that although this implementation is for the Ethereum network, it is blockchain-agnostic, and is easily adaptable to any other blockchain. The Cartesi developers want the Cartesi layer-2 platform architecture to be perceived as blockchain-agnostic as well. However, at this time, Cartesi and Descartes only support the EVM, and as such can only be deployed on Ethereum and EVM compatible networks. As more new networks are supported, surely the cost associated with signature verification will be negligible, considering that Ethereum is one of the most expensive blockchains to operate.

Conceptual Improvements over the Proof of Concept

Although it effectively works as intended, the implemented authentication solution is simple, but inflexible. This forces the use of a key management system, where it is necessary to have a public-key

for every private-key in the smart contract. However, a more complex solution can be implemented to increase the trust guarantees of a DApp. A developer implementing this solution needs to make trade-offs about key management, especially the generation of public- and private-keys and the distribution of them to the parties. For this, there are two essential points to consider in key management: key distribution and key revocation. Key management helps to ensure secure handling practices for cryptographic keys used in this case with asymmetric encryption techniques. Key distribution is the method of delivering cryptographic keys to parties in order to maintain confidentiality and authenticity. Key revocation refers to the task of securely removing keys from the system that are known to be compromised.

There are several proposals for key distribution with different degrees of confidence. There are simpler and more convenient solutions based on attaching the key to the data or message to be sent. A greater degree of security can be achieved with solutions that maintain a dynamic directory for key handling by assigning the responsibility to a trusted entity or organization or, for an even more secure case, to an authority. Finally, a solution which fits this case study is the distribution of keys with Certificate Authorities. Project Amber[1] offers a solution that fits well in the architecture of Cartesi applications due to its interface with the cloud. The main goal of Project Amber is to increase trust in Confidential Computing. Project Amber is an innovative zero-trust approach to objective third-party attestation, that enables the use of a single trust authority regardless of where the applications run. This project enables remote verification and assertion of the reliability of computations, such as TEEs and Roots of Trust. Its service is independent of the provider of the Cloud/Edge infrastructure that hosts the confidential workloads.

The presented proof of concept implementation is a simpler solution to add authentication to a Cartesi DApp. This authentication layer protects a DApp by proving data integrity. In this case, a TEE-based implementation solution would be ideal for providing authentication to Cartesi DApps. Authentication through signature verification prevents an attacker from corrupting the Cartesi Machine by tampering with the output, since malicious output will not be properly signed. By running this solution in an enclave or any other trusted environment, the security of the application will be increased by decreasing the attack surface and system vulnerabilities. This is where TEEs play a special role, especially TEE containers, which allow running Docker containers in a secure environment. Deploying applications within these environments protects the data with confidential computing technology, with almost no change in the application itself.

In the specific case of Cartesi DApps, leveraging TEEs containers allows securing and isolate each component from Descartes Nodes. Since the Descartes Node consists of a set of Docker containers for each off-chain component of a DApp, using TEEs it is possible to run each component in a container in a secure environment. In addition to protecting from Docker's inherent vulnerabilities, it also protects containers from a compromised host that can exploit the system's weaknesses, as can be observed in Figure 28.

Although it is possible to run every component of the Cartesi DApps on a TEE, it should not be done in full. As in any TEE implementation, only the critical and essential parts should be secured.

[1] Project Amber by Intel - https://www.intel.com/content/www/us/en/security/project-amber.html

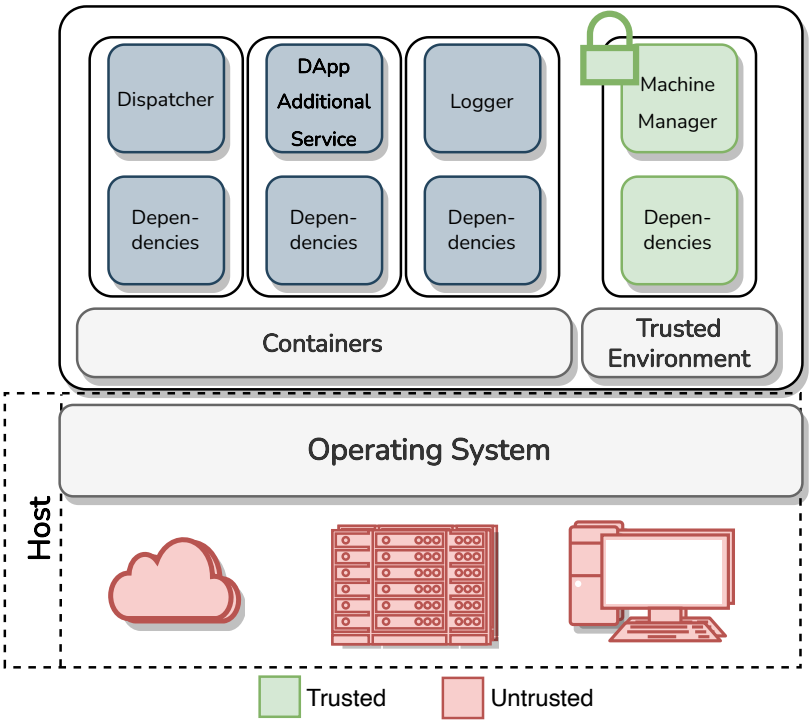


Figure 28: Cartesi DApp Containers Protected by TEEs.

When running Machine Manager inside a trusted environment, all communications will be protected. Since the Cartesi Machine results are signed, any tampering with the data will be detectable, assuming the signature verification is incorruptible. Even if after sending the output from the Cartesi Machine to the Dispatcher the data is corrupted, it will be verifiable on-chain since it would have to be properly signed. Thus, running all containers or components of a DApp in a trusted environment is not necessary, considering the performance losses it could cause to the system.

Chapter 6

Conclusions

In this chapter, we discuss the findings drawn from the analysis of the Cartesi project and from the security analysis performed on the present work. Finally, we present suggestions for improvement and future work based on the work developed in this thesis.

6.1 Conclusion

In this thesis we present the security analysis of a layer-2 blockchain solution, Cartesi, in which potential threats found are mitigated by leveraging TEE technology. This work stands out from other related work by combining TEE with layer-2, which until now has only been presented in layer-1 solutions. Moreover, it is a challenging topic because it encompasses several completely different areas and technologies, including blockchain and smart contracts, embedded systems, information systems and security. In addition to diversity, the topics related to Web3 are still very new and are poorly documented, which made the research even more challenging.

We presented the main concepts of web3 and blockchain in order to ground and build a foundation for insight into the later analysis. This background is essential given the novelty and exponential growth of these technologies that are still in an early stage of research and development. Given this immaturity, blockchain continues to face challenges mainly in terms of scalability, and so layer-2 solutions such as Cartesi have emerged presenting improvements in this domain. Even so, following the scalability trilemma, which encompasses scalability, security and decentralization, it is still a challenge to present protocols that succeed in improving two of these aspects without harming the other.

The analysis performed in this work shows that the protocol presented by Cartesi proposes to increase scalability, but at the expense of security, given the vulnerabilities exploited in chapter 4. We presented and analyzed the entire Cartesi platform, focusing on the crucial points for understanding the security

analysis and the proposed mitigation solutions. In addition, we presented the attack carried out on a Cartesi DApp that proves the stated threats and undermines Cartesi's system and security guarantees.

Given the enunciated problems, this thesis proposed the use of TEEs to increase the trust of the Cartesi protocol, offering developers greater guarantees in specific applications. We presented several TEE solutions, for various platforms, and we also demonstrated a proof of concept of a simple solution based on digital signatures. This solution allowed to increase the security of Cartesi DApps by adding an authentication layer to them, with the results being cryptographically signed. Finally, we concluded with a more complex solution that will potentially be even more secure and will be further explored in future work.

The objectives of this thesis were successfully achieved, and it was possible to conclude that the combination of TEE technologies with the Cartesi platform is effectively suitable for the development and adoption of this layer-2 solution.

6.2 Future Work

All the proposed goals were met, but during the course of this work several ideas for future improvements using these technologies have emerged. Besides improvements related to the system under study, there are many ways to expand on the approach of this thesis.

Beginning with those directly related to the main theme of this thesis, the first future work could be the real implementation of a mitigation solution with TEEs. This work already provides the theoretical know-how to effectively perform this implementation, being only necessary to resort to the necessary software and hardware resources. A more complex solution could be discussed with the Cartesi developers, for doing slight changes to the architecture to adapt an even more secure approach. This collaboration with the team could also help to reach a more scalable and flexible key management solution, and perhaps resort to Certificate Authorities, depending on the specific use case. Therefore, the first future work proposed is the implementation of the off-chain component of the Cartesi protocol, called Descartes Nodes, in a TEE.

Another suggestion is the porting of this solution to the Cartesi Rollups. Cartesi Rollups are a new technology to be implemented by Cartesi that differs slightly on Descartes, which was covered in this thesis, but relies on the same virtual machine that emulates the RISC-V ISA, the Cartesi Machine. Given the similarity of the components and the trust model, it will certainly be relatively easy to adapt the mitigation solution to the new technology. Cartesi Rollups were not addressed in this document because they were still under development at the time of this thesis. In summary, the second future work proposal is the implementation of the off-chain component of Cartesi Rollups in a TEE. This proposal goes outside the scope of the system under analysis, but builds on the analysis of other existing layer-2 solutions to study the possibility of leveraging TEEs. The work done in the second suggestion might be useful in this one, since

most of the large layer-2 solutions such as Arbitrum[1], Optimism[2] and dYdX[3] use the Rollups approach used in the new Cartesi protocol. In addition, other promising solutions like Immutable X[4], ApeX[5] and OMG Network[6] that do not use rollups but are also based on moving computations off-chain can also be researched. Therefore, the third proposed future work is a security analysis of another layer-2 protocol(s), using TEEs as a mitigation solution. To conclude, all proposals are based on increasing the security of layer-2 solutions using TEEs and thus contribute to developments in this field of research.

[1] Arbitrum - Solve Scaling without compromise. https://arbitrum.io/
[2] Optmism - Low-cost and lightning-fast Ethereum L2 blockchain.https://optimism.io/
[3] dYdX - Layer 2 protocol for cross-margined perpetual smart contracts https://dydx.exchange/
[4] Immutable X - Layer 2 for NFTs secured by Ethereum https://immutable.com/
[5] ApeX - Layer-2 trading platform built on the ApeX Protocol https://apex.exchange/
[6] OMG Network - Layer 2 Optimistic Rollup scaling solution https://omg.network/

Bibliography

[1] S. Nakamoto. *Bitcoin: A Peer-To-Peer Electronic Cash System*. White Paper. 2008. url: https://bitcoin.org/bitcoin.pdf (cit. on p. 1).

[2] T. Alladi et al. "Blockchain Applications for Industry 4.0 and Industrial IoT: A Review". In: *IEEE Access*. 2019 (cit. on p. 1).

[3] F. Ullah and F. Al-Turjman. "A conceptual framework for Blockchain Smart Contract Adoption to manage real estate deals in Smart Cities". In: *Neural Computing and Applications*. 2021 (cit. on p. 1).

[4] M. J. J. Gul et al. "Blockchain based healthcare system with Artificial Intelligence". In: *Proc. of CSCI*. 2020 (cit. on p. 1).

[5] J. Wu and N. K. Tran. "Application of Blockchain Technology in Sustainable Energy Systems: An Overview". In: *Sustainability*. 2018 (cit. on p. 1).

[6] B. K. Mohanta, S. S. Panda, and D. Jena. "An Overview of Smart Contract and Use Cases in Blockchain Technology". In: *Proc. of ICCCNT*. 2018 (cit. on p. 1).

[7] V. Buterin et al. *A next-generation smart contract and decentralized application platform*. White Paper. 2014. url: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf (cit. on p. 1).

[8] V.-C.-T. Linh et al. "Votereum: An Ethereum-Based E-Voting System". In: *Proc. of IEEE-RIVF*. 2019 (cit. on p. 1).

[9] S. Jianjun, L. Ming, and M. Jingang. "Research and application of data sharing platform integrating Ethereum and IPFs Technology". In: *Proc. of International Symposium on DCABES*. 2020 (cit. on p. 1).

[10] I. Tonev. "Energy Trading Web Platform Based on the Ethereum Smart Contracts and Blockchain". In: *Proc. of BulEF*. 2020 (cit. on p. 1).

[11] A. Altarawneh et al. "Buterins Scalability Trilemma viewed through a State-change-based Classification for Common Consensus Algorithms". In: *Proc. of CCWC*. 2020 (cit. on p. 1).

[12] Y. Liu et al. “Building Blocks of Sharding Blockchain Systems: Concepts, Approaches, and Open Problems”. In: *arXiv preprint arxiv:2102.13364*. 2021 (cit. on p. 1).

[13] K. Coutinho et al. “Enabling Blockchain Scalability and Interoperability with Mobile Computing through LayerOne.X”. In: *arXiv preprint arxiv:2110.01398*. 2021 (cit. on p. 1).

[14] Q. Wang et al. “HyperChannel: A Secure Layer-2 Payment Network for Large-Scale IoT Ecosystem”. In: *Proc. of IEEE International Conference on Communications*. 2021 (cit. on p. 1).

[15] A. Tanksali. “XipCOIN: A Proposal for Decentralized Offline Payments using Ethereum”. In: *Proc. IEEE TEMSMET*. 2021 (cit. on p. 1).

[16] A. Teixeira and D. Nehab. *The Core of Cartesi*. White Paper. 2019. url: https://cartesi.io/cartesi_whitepaper.pdf (cit. on p. 2).

[17] X. Li et al. “Blockchain Consensus Algorithms: A Survey”. In: *China Automation Congress CAC*. 2021 (cit. on p. 4).

[18] W. Viriyasitavat and D. Hoonsopon. “Blockchain characteristics and consensus in modern business processes”. In: *Journal of Industrial Information Integration*. 2019 (cit. on p. 4).

[19] B. Sriman and S. G. Kumar. “Decentralized finance (DeFi): The Future of Finance and Defi Application for Ethereum blockchain based Finance Market”. In: *Proc. of ACCAI*. 2022 (cit. on p. 5).

[20] S. Wang et al. “Decentralized Autonomous Organizations: Concept, Model, and Applications”. In: *IEEE Transactions on Computational Social Systems*. 2019 (cit. on p. 5).

[21] Q. Wang et al. “Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges”. In: *arXiv preprint arxiv:2105.07447*. 2021 (cit. on p. 5).

[22] M. Mita et al. “What is Stablecoin?: A Survey on Price Stabilization Mechanisms for Decentralized Payment Systems”. In: *Proc. of International Congress on Advanced Applied Informatics*. 2019 (cit. on p. 5).

[23] P. Boonparn et al. “Social Data Analysis on Play-to-Earn Non-Fungible Tokens (NFT) Games”. In: *Proc. of IEEE Global Conference on Life Sciences and Technologies*. 2022 (cit. on p. 5).

[24] S. Mishra et al. “Contribution of Blockchain in Development of Metaverse”. In: *Proc. of ICCES*. 2022 (cit. on p. 5).

[25] V. Chang et al. “How Blockchain can impact financial services – The overview, challenges and recommendations from expert interviewees”. In: *Technological Forecasting and Social Change*. 2020 (cit. on p. 5).

[26] R. Azzi, R. K. Chamoun, and M. Sokhn. “The power of a blockchain-based supply chain”. In: *Computers & Industrial Engineering*. 2019 (cit. on p. 5).

[27] A. A. Monrat, O. Schelen, and K. Andersson. “A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities”. In: *IEEE Access*. 2019 (cit. on p. 5).

[28] R. Casado-Vara et al. “How blockchain improves the supply chain: case study alimentary supply chain”. In: *Proc. of Procedia Computer Science*. 2018 (cit. on p. 5).

[29] M. S. Ferdous et al. “Blockchain Consensus Algorithms: A Survey”. In: *arXiv preprint arxiv:2001.07091*. 2020 (cit. on p. 5).

[30] R. Beer and T. Sharma. “A quick look at Cryptocurrency Mining: Proof of Work”. In: *Proc. of ICIPTM*. 2022 (cit. on p. 5).

[31] F. Saleh. “Blockchain Without Waste: Proof-of-Stake”. In: *SSRN Electronic Journal*. 2018 (cit. on p. 5).

[32] A. Hafid, A. S. Hafid, and M. Samih. “Scaling Blockchains: A Comprehensive Survey”. In: *IEEE Access*. 2020 (cit. on p. 6).

[33] N. Sarrar. “On transaction parallelizability in Ethereum”. In: *arXiv preprint arxiv:1901.09942*. 2019 (cit. on p. 6).

[34] X. Fan and Q. Chai. “Roll-DPoS”. In: *Proc. of EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 2018 (cit. on p. 6).

[35] M. Janssen et al. “A framework for analysing blockchain technology adoption: Integrating institutional, market and technical factors”. In: *International Journal of Information Management*. 2020 (cit. on p. 6).

[36] A. I. Sanka and R. C. Cheung. “A systematic review of blockchain scalability: Issues, solutions, analysis and future research”. In: *Journal of Network and Computer Applications*. 2021 (cit. on p. 7).

[37] A. Singh et al. “Sidechain technologies in blockchain networks: An examination and state-of-the-art review”. In: *Journal of Network and Computer Applications*. 2020 (cit. on p. 7).

[38] J.-H. Lin et al. “Lightning network: a second path towards centralisation of the Bitcoin economy”. In: *New Journal of Physics*. 2020 (cit. on p. 7).

[39] J. Poon and V. Buterin. *Plasma: Scalable autonomous smart contracts*. White Paper. 2017. url: https://www.plasma.io/plasma.pdf (cit. on p. 7).

[40] J. Adler and M. Quintyne-Collins. “Building Scalable Decentralized Payment Systems”. In: *arXiv preprint arxiv:1904.06441*. 2020 (cit. on p. 8).

[41] X. Sun et al. “A Survey on Zero-Knowledge Proof in Blockchain”. In: *IEEE Network*. 2021 (cit. on p. 8).

[42] L. Tremblay Thibault, T. Sarry, and A. Senhaji Hafid. “Blockchain Scaling Using Rollups: A Comprehensive Survey”. In: *IEEE Access*. 2022 (cit. on p. 9).

[43] N. Szabo. “Formalizing and Securing Relationships on Public Networks”. In: *First Monday*. 1997 (cit. on p. 9).

[44] S. Lande and R. Zunino. “SoK: unraveling Bitcoin smart contracts”. In: *Principles of Security and Trust LNCS*. 2018 (cit. on p. 9).

[45] S. M. Werner et al. “SoK: Decentralized Finance (DeFi)”. In: *arXiv preprint arxiv:2101.08778*. 2021 (cit. on p. 9).

[46] C. Chin-Ling et al. “A Traceable Online Insurance Claims System Based on Blockchain and Smart Contract Technology”. In: *Sustainability*. 2021 (cit. on p. 9).

[47] V. Aleksieva, H. Valchanov, and A. Huliyan. “Implementation of Smart Contracts based on Hyperledger Fabric Blockchain for the Purpose of Insurance Services”. In: *Proc. of International Conference BIA*. 2020 (cit. on p. 9).

[48] H. L. Pham, T. H. Tran, and Y. Nakashima. “A Secure Remote Healthcare System for Hospital Using Blockchain Smart Contract”. In: *Proc. of IEEE GC Wkshps*. 2018 (cit. on p. 9).

[49] A. Madhusudan et al. “SC2Share: Smart Contract for Secure Car Sharing”. In: *Proc. of ICISSP*. 2019 (cit. on p. 9).

[50] S. J. Pee et al. “Blockchain based smart energy trading platform using smart contract”. In: *Proc. of ICAIIC*. 2019 (cit. on p. 9).

[51] S. Damle, S. Gujar, and M. H. Moti. “FASTEN: Fair and Secure Distributed Voting Using Smart Contracts”. In: *Proc. of IEEE ICBC*. 2021 (cit. on p. 9).

[52] B. Yu et al. “Food Quality Monitoring System Based on Smart Contracts and Evaluation Models”. In: *IEEE Access*. 2020 (cit. on p. 10).

[53] R. Casado-Vara et al. “Smart Contract for Monitoring and Control of Logistics Activities: Pharmaceutical Utilities Case Study”. In: *Proc. of International Joint Conference*. 2018 (cit. on p. 10).

[54] M. Wohrer and U. Zdun. “Domain Specific Language for Smart Contract Development”. In: *Proc. of IEEE ICBC*. 2020 (cit. on p. 10).

[55] F. Spoto. “A Java Framework for Smart Contracts”. In: *Financial Cryptography Workshops*. 2019 (cit. on p. 10).

[56] I. Grigg. *EOS-An Introduction*. White Paper. 2017. url: https://www.iang.org/papers/EOS_An_Introduction-BLACK-EDITION.pdf (cit. on p. 10).

[57] R. Cheng et al. “Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts”. In: *Proc. of IEEE EuroS&P*. 2019 (cit. on p. 10).

[58] M. Saeed et al. “Trust Management Technique Using Blockchain in Smart Building”. In: *Engineering Proceedings*. 2022 (cit. on p. 10).

[59] P. Christodoulou, A. S. Andreou, and Z. Zinonos. “skillsChain: A Decentralized Application That Uses Educational Robotics and Blockchain to Disrupt the Educational Process”. In: *Sensors*. 2021 (cit. on p. 10).

[60] R. Burkert et al. “Decentralized Online Multiplayer Game Based on Blockchains”. In: *Blockchain and Applications*. 2022 (cit. on p. 10).

[61] S. Venkata Sai Santosh et al. “Decentralized Application for Two-Factor Authentication with Smart Contracts”. In: *Inventive Communication and Computational Technologies*. 2021 (cit. on p. 10).

[62] R. Tas and O. O. Tanriover. “Building A Decentralized Application on the Ethereum Blockchain”. In: *Proc. of MSIT*. 2019 (cit. on p. 11).

[63] T. Min et al. “Blockchain Games: A Survey”. In: *Proc. of IEEE CoG*. 2019 (cit. on p. 11).

[64] A. Laurent, L. Brotcorne, and B. Fortz. “Transaction fees optimization in the Ethereum blockchain”. In: *Blockchain: Research and Applications*. 2022 (cit. on p. 12).

[65] L. Marchesi et al. “Design Patterns for Gas Optimization in Ethereum”. In: *Proc. of IEEE IWBOSE*. 2020 (cit. on p. 12).

[66] Q. Feng et al. “A survey on privacy protection in blockchain system”. In: *Journal of Network and Computer Applications*. 2019 (cit. on p. 12).

[67] A. Feder et al. “The impact of DDoS and other security shocks on Bitcoin currency exchanges: evidence from Mt. Gox”. In: *Journal of Cybersecurity*. 2018 (cit. on p. 12).

[68] A. Mense and M. Flatscher. “Security Vulnerabilities in Ethereum Smart Contracts”. In: *Proc. of iiWAS*. 2018 (cit. on p. 12).

[69] C. Shier et al. “Understanding a revolutionary and flawed grand experiment in Blockchain: The dao attack”. In: *SSRN Electronic Journal*. 2017 (cit. on p. 12).

[70] *CertiK: Building Fully Trustworthy Smart Contracts and Blockchain Ecosystems*. White Paper. 2019. url: https://crebaco.com/planner/admin/uploads/whitepapers/6260310certik.pdf (cit. on p. 12).

[71] M. Rosenfeld. “Analysis of Hashrate-Based Double Spending”. In: *arXiv preprint:1402.2009*. 2014 (cit. on p. 12).

[72] C. Ye et al. “Analysis of Security in Blockchain: Case Study in 51%-Attack Detecting”. In: *Proc. of International Conference on DSA*. 2018 (cit. on p. 12).

[73] S. Shanaev et al. “Cryptocurrency Value and 51% Attacks: Evidence from Event Studies”. In: *SSRN Electronic Journal*. 2018 (cit. on p. 12).

[74] J. Teutsch and C. Reitwießner. “A scalable verification solution for blockchains”. In: *arXiv preprint arxiv:1908.04756*. 2019 (cit. on p. 13).

[75] H. Kalodner et al. “Arbitrum: Scalable, private smart contracts”. In: *Proc. of USENIX Security*. 2018 (cit. on p. 13).

[76] D. P. Mulligan et al. “Confidential computing—a brave new world”. In: *Proc. of SEED*. 2021 (cit. on p. 15).

[77] J. Kaplan David Powell and T. Woller. *AMD Memory Encryption*. White Paper. 2021. url: https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf (cit. on p. 15).

[78] Arm. *ARM Security Technology. Building a Secure System using TrustZone Technology ARM*. White Paper. 2009. url: https://documentation-service.arm.com/static/5f212796500e883ab8e74531?token= (cit. on p. 15).

[79] M. U. Sardar, S. Musaev, and C. Fetzer. "Demystifying Attestation in Intel Trust Domain Extensions via Formal Verification". In: *IEEE Access*. 2021 (cit. on p. 15).

[80] S. Pinto et al. "Virtualization on TrustZone-Enabled Microcontrollers? Voilà!" In: *Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium*. 2019 (cit. on p. 15).

[81] S. Pinto and N. Santos. "Demystifying Arm TrustZone". In: *ACM Computing Surveys*. 2019 (cit. on p. 15).

[82] E. Jan-Erik, K. Kostiainen, and N. Asokan. "Trusted execution environments on mobile devices". In: *Proc. of ACM SIGSAC CCS*. 2013 (cit. on p. 16).

[83] S. Pinto et al. "Towards a TrustZone-Assisted Hypervisor for Real-Time Embedded Systems". In: *Proc. of IEEE Computer Architecture Letters*. 2017 (cit. on p. 16).

[84] D. Lee et al. "Keystone: An Open Framework for Architecting Trusted Execution Environments". In: *Proc. of ACM EuroSys*. 2020 (cit. on p. 16).

[85] C. Garlati and S. Pinto. "Secure IoT Firmware For RISC-V Processors". In: *Embedded World Conference*. 2021 (cit. on p. 16).

[86] D. Oliveira, T. Gomes, and S. Pinto. "uTango: An Open-Source TEE for IoT Devices". In: *IEEE Access*. 2022 (cit. on p. 16).

[87] A. Kivity et al. "KVM: the Linux Virtual Machine Monitor". In: *Proc. of Linux Symposium*. 2007 (cit. on p. 16).

[88] J. Martins et al. "Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems". In: *Workshop on Next Generation Real-Time Embedded Systems*. 2020 (cit. on p. 16).

[89] R. Ramsauer et al. "Look Mum, no VM Exits!(Almost)". In: *Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*. 2017 (cit. on p. 16).

[90] P. Barham et al. "Xen and the Art of Virtualization". In: *Operating Systems Review (ACM)*. 2003 (cit. on p. 16).

[91] X. Chen et al. "Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems". In: *Proc. of ACM SIGOPS*. 2008 (cit. on p. 16).

[92] O. Hofmann et al. “InkTag: Secure Applications on an Untrusted Operating System”. In: *ASPLOS*. 2013 (cit. on p. 16).

[93] J. M. McCune et al. “TrustVisor: Efficient TCB reduction and attestation”. In: *Proc. of S&P*. 2010 (cit. on p. 16).

[94] J. Yang. “Using Hypervisor to Provide Application Data Secrecy on a Per-Page Basis”. In: *Proc. of Virtual Execution Environments*. 2008 (cit. on p. 16).

[95] J. Jang and B. B. Kang. “Retrofitting the Partially Privileged Mode for TEE Communication Channel Protection”. In: *IEEE Transactions on Dependable and Secure Computing*. 2020 (cit. on p. 16).

[96] S. Pereira et al. “Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems”. In: *Workshop on Next Generation Real-Time Embedded Systems*. 2008 (cit. on p. 16).

[97] B. Sá, J. Martins, and S. Pinto. “A First Look at RISC-V Virtualization From an Embedded Systems Perspective”. In: *Proc. of IEEE Transactions on Computers*. 2022 (cit. on p. 16).

[98] D. Cerdeira et al. “SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-Assisted TEE Systems”. In: *Proc. of IEEE S&P*. 2020 (cit. on p. 16).

[99] A. Nilsson, P. N. Bideh, and J. Brorsson. “A survey of published attacks on Intel SGX”. In: *arXiv preprint arXiv:2006.13598*. 2020 (cit. on p. 16).

[100] D. Cerdeira et al. “ReZone: Disarming TrustZone with TEE Privilege Reduction”. In: *Proc. of USENIX Security*. 2022 (cit. on p. 16).

[101] W. Li et al. “TEEv: Virtualizing Trusted Execution Environments on Mobile Platforms”. In: *Proc. of VEE*. 2019 (cit. on p. 16).

[102] M. U. Sardar. “Understanding Trust Assumptions for Attestation in Confidential Computing”. In: *Proc. of IEEE/IFIP DSN-S*. 2022 (cit. on p. 16).

[103] X. Lin et al. “A Measurement Study on Linux Container Security: Attacks and Countermeasures”. In: *Proc. of ACSAC*. 2018 (cit. on p. 17).

[104] S. Sultan, I. Ahmad, and T. Dimitriou. “Container Security: Issues, Challenges, and the Road Ahead”. In: *IEEE Access*. 2019 (cit. on p. 17).

[105] M. Russinovich et al. “Toward Confidential Cloud Computing”. In: *Communications of the ACM*. 2021 (cit. on p. 17).

[106] F. Brasser et al. “Trusted Container Extensions for Container-based Confidential Computing”. In: *arXiv preprint arxiv:2205.05747*. 2022 (cit. on p. 17).

[107] S. Arnautov et al. “SCONE: Secure Linux Containers with Intel SGX”. In: *Proc. of USENIX OSDI*. 2016 (cit. on pp. 17, 47, 48).

[108] A. Randazzo and I. Tinnirello. "Kata Containers: An Emerging Architecture for Enabling MEC Services in Fast and Secure Way". In: *Proc. of IOTSMS*. 2019 (cit. on p. 17).

[109] G. Ayoade et al. "Decentralized IoT Data Management Using BlockChain and Trusted Execution Environment". In: *Proc. of IEEE IRI*. 2018 (cit. on p. 18).

[110] J. Lind et al. "Teechain". In: *Proc. of ACM Symposium on Operating Systems Principles*. 2019 (cit. on p. 18).

[111] W. Li et al. "Securing Proof-of-Stake Blockchain Protocols". In: *Proc. of European Symposium on Research in Computer Security International Workshop on Data Privacy Management Cryptocurrencies and Blockchain Technology*. 2017 (cit. on p. 19).

[112] S. Andreina et al. "PoTS: A Secure Proof of TEE-Stake for Permissionless Blockchains". In: *IEEE Transactions on Services Computing*. 2020 (cit. on p. 19).

[113] H. Dang et al. "Towards Scaling Blockchain Systems via Sharding". In: *Proc. of the International Conference on Management of Data*. 2019 (cit. on p. 19).

[114] C. Woetzel. *Secret Network: A Privacy-Preserving Secret Contract & Decentralized Application Platform*. White Paper. url: https://scrt.network/ (cit. on p. 20).

[115] O. P. Project. *The Oasis Blockchain Platform*. White Paper. 2020. url: https://oasisprotocol.org/papers (cit. on p. 20).

[116] "Supporting private data on Hyperledger Fabric with secure multiparty computation". In: *Proc. of IEEE IC2E*. 2018 (cit. on p. 20).

[117] C. Gentry. "Fully Homomorphic Encryption Using Ideal Lattices". In: *Proc. of ACM Symposium on Theory of Computing*. 2009 (cit. on p. 20).

[118] E. Ben Sasson et al. "Zerocash: Decentralized anonymous payments from Bitcoin". In: *Proc. of IEEE S&P*. 2014 (cit. on p. 20).

[119] J. Xiao and K. Qin. *KubeTEE*. 2020. url: https://github.com/SOFAEnclave/KubeTEE (cit. on p. 47).

[120] C.-C. Tsai, D. E. Porter, and M. Vij. "Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX". In: *Proc. of USENIX ATC*. 2017 (cit. on p. 47).

[121] Z. Hua et al. "TZ-container: Protecting container from untrusted OS with arm TrustZone". In: *Science China Information Sciences*. 2021 (cit. on p. 47).

[122] C. N. C. Foundation. *Confidential Containers*. 2022. url: https://github.com/confidential-containers (cit. on p. 47).

[123] W. Fang et al. "Digital Signature Scheme for information non-repudiation in Blockchain: A state of the art review". In: *EURASIP Journal on Wireless Communications and Networking*. 2020 (cit. on p. 49).