UiT The Arctic University of Norway

The Faculty of Science and Technology
Department of Computer Science

Aquilier: An Ethereum-Based Smart Contract for Door-Lock Management in Home Assistant

Niklas Strand

Master's Thesis in Computer Science INF-3981

“”

—

Abstract

The widespread adoption of distributed computer systems, exemplified by platforms like Airbnb and Booking.com, has transformed homes into rental properties and streamlined vacation rentals by offering comprehensive tools for listing properties, processing payments, facilitating searches, and enabling communication. However, a critical gap remains: these platforms do not facilitate the coordination of physical access to these properties, a significant challenge affecting tenants and service personnel. This gap complicates logistics and raises substantial security concerns, as property owners must ensure safe and authorised access without direct oversight. Additionally, landlords are increasingly facing disputes concerning the true identity of tenants, questioning whether they are indeed who they claim to be. There are also concerns about whether the individuals who book the property are the same who occupy it.

In this thesis, we propose Aquilier, a novel decentralised system utilising blockchain technology and smart contracts to manage and log physical access to rental properties. Distinguishing itself from traditional methods, Aquilier integrates seamlessly with the HomeAssistant smart-home systems to control door locks, enhancing the security and convenience of property access. All operations are securely logged in a tamper-proof ledger using the Ethereum blockchain, ensuring unparalleled security and accountability in access management. We show that this approach is reliable and secure but that further research is needed to reduce the cost of blockchain operations.

Acknowledgements

I am grateful to my supervisor, Professor. Håvard Dagenborg, for his invaluable guidance, support, and expertise throughout this project. His insights and suggestions have been crucial in shaping this thesis.

I would also like to express my deepest appreciation to my girlfriend, Frida Omma, for her constant support and encouragement throughout this journey. Her constant reminders to stay focused and her dedication have been a significant source of strength and motivation for me.

Special thanks are due to Guro Jansurud and Morgan Prott for their great support and friendship. Their positive attitude has motivated me to complete the thesis.

I am sincerely grateful to everyone who has supported me in this journey.

Contents

List of Figures

List of Tables

1

Introduction

The vacation rental market has undergone a significant transformation, shifting from the simplicity of roadside vacancy signs to a digital ecosystem dominated by a vast array of internet platforms. These platforms cater to a broad audience, from major hotel chains and travel agencies to individual homeowners looking to rent out a spare room, effectively reshaping the industry's competitive landscape.

Digitalisation has brought numerous benefits to the vacation rental market, offering greater access and flexibility for users. Digital platforms allow individuals to rent out their homes, while owners rent other properties, effectively utilising housing capacity.

However, alongside these benefits, several drawbacks have emerged. Communities and cities have seen significant impacts on property prices. Governments face challenges with taxation and regulation due to the nature of these platforms [1, 2]. Fraudulent hosts and property misuse by renters have also become prevalent issues [3].

As a result of these developments, the current short-term rental market faces critical challenges, notably the need to build trust among governments, hosts, and tenants. There is also an emerging requirement for integrating seamless access in smart home systems, enabling automated rental management, addressing regulatory and taxation enforcement issues, streamlining operational processes to enhance property management efficiency, and ensuring security

and synchronisation across rental platforms.

These multi-faced challenges highlight the intricate features and requirements future systems must solve to encompass the market demand. Thus, the transformative potential of smart contract technology is shown to be beneficial [4]. Blockchain technology has emerged as a feasible solution to address similar issues by quantifying and decentralising trust [5]. The immutability and transparency properties of decentralised applications (DApps) present a viable solution to the current issues in the short-term rental market. Blockchain offers an innovative approach to distributing trust and streamlining transactions without the need for central authority or intermediaries. The technology could effectively address the prevalent gaps in the market, proposing a more organised, trustworthy, and efficient rental ecosystem.

1.1 Problem Statement

Given the apparent lack of trust and governance in the digital platforms used in the rental market, we explore and propose a potential solution to these issues, focusing on enhancing security and reliability in digital access management for rental properties. The intricate task of transferring access between parties essentially encapsulates the broader issues identified in our analysis; trust, transparency, and privacy. We believe that resolving the access transfer dilemma can redefine rental market standards.
The thesis focuses on applying smart contracts and leveraging the blockchain's immutable and transparent features to manage and log property access. How can property access be automated to ensure security and transparency while preserving privacy in the digital realm? The question is a cornerstone in exploring blockchain technology as a transformative tool for the vacation rental industry. Therefore, our thesis is as follows:

> Utilising Ethereum smart contracts to transfer property access can automate property management and enhance security and operational efficiency.

To address our thesis, we first conducted a market analysis of existing online short-term rental platforms. We second devised Aquilier, a DApp designed to address these multifaceted challenges in the vacation rental market. Aquilier stands out by leveraging the robustness of blockchain technology and the precision of smart contracts to manage and transfer physical access to rental properties. A Blockchain system is engineered to enhance trust, security and transparency, which are essential in the digital short-term ecosystem.

Aquilier's core functionality is anchored in a sophisticated locking mechanism seamlessly integrated with HomeAssistant smart-home systems. This mechanism is more than a physical barrier; it is part of a comprehensive security protocol incorporating decentralised identifiers (DID) and verifiable credentials(VC). Only authorised individuals, verified through a transparent and immutable blockchain system, can access the property. This approach significantly addresses trust issues by providing a reliable method for property owners to verify tenant identities and manage access permissions.

1.2 Methodology

This thesis adopts a mixed-methods approach, utilising qualitative and quantitative analyses, to comprehensively identify and address the challenges in the short-term real estate rental market, particularly focusing on integrating blockchain technology and smart home systems.

We adopted the methodology framework detailed in Anne Håkanson's comprehensive work on research methods and methodologies for research projects [6]. Håkanson's insights were instrumental across various stages, from the initial market analysis to the implementation phase of our project. Her emphasis on methodological rigour and adaptability significantly shaped our mixed-methods approach. This approach ensured that our research was rooted in sound academic practices and capable of effectively addressing the complex aspects of our study.

The methodology begins with an in-depth market analysis, examining 15 rental platforms and key governmental documents to identify market trends, regulatory perspectives, and technological gaps. This is followed by the practical implementation of 'Aquilier', where we develop and deploy a decentralised solution, utilising agile development practices and iterative testing. The conclusion synthesises the findings from both the market analysis and implementation phases, evaluating the efficacy of the proposed solution against the identified market needs and technological challenges. This multifaceted approach ensures a thorough understanding and validation of the research hypothesis, bridging the gap between theoretical analysis and practical application.

The Qualitative Market Analysis (chapter 3) was essential in exploring the dynamics of the short-term real estate rental market, and the integration potential of blockchain and smart home technologies. The qualitative method involved an in-depth analysis of 15 rental platforms and official EU and US documents. This analysis provided valuable insights into the market's current state, regulatory perspectives, and technological gaps.

The mixed-methods approach allowed for a comprehensive market understanding, blending detailed qualitative insights with structured quantitative summaries. The methodology's vigilant aspect, where qualitative and quantitative data complement and validate each other, ensured the robustness and reliability of the research findings.

1.3 Methodical Approach in Software Engineering

The thesis applies the Software Development Life Cycle (SDLC) framework [7]. While developing Aquilier, we adapted the traditional SDLC model, focusing on six key phases, from Requirement Analysis to Maintenance, to cater to the specific needs and constraints of the Aquilier project. This adaptation not only aligns with the objectives of this thesis but also underlines the importance of structured, methodical approaches in software engineering. We also blended this more traditional approach with a scrum sprint during the implementation phase due to the experimental nature of developing smart contracts.

Analysis

The first stage involved analysing requirements, a process we undertook through a comprehensive market analysis. This analysis, detailed in Chapter 3, helped us identify several existing gaps and potential requirements in the current market offerings. Based on these findings, we narrowed our focus to a specific feature that we believed was crucial and represented the issues we uncovered. The feature we chose to concentrate on was the locking mechanism of a property and the process of granting access, which we recognised as crucial for enhancing security and user experience in real estate rentals. Addressing key features, we aim to establish a foundation for resolving identified market gaps. The requirement is encapsulated in user stories.

Design

The user stories we had identified in the previous stage formed the design stage. In response, we developed three smart contracts that would synchronise data: Property Registry, User Registry and booking management. We recognised early on that a direct connection between the smart home system and the blockchain was unfeasible. Consequently, the need for implementing a proxy became evident. This proxy serves as a link, bridging the smart home system with the blockchain. Given the decentralised nature of the blockchain, maintaining a consistent connection between the blockchain and the proxy presented a challenge. We faced two options: operating a node on the network

or relying on an external service for a stable connection. We chose the latter option for simplicity and efficiency, selecting Infura [8] as our service provider. This choice facilitated seamless integration with the blockchain. The design choices, including this integration, are detailed in Figure 5.1.

Implementation

Knowing the desired design of our system, we read up and grasped the tools and prerequisites necessary to develop the system's code base. Our smart contract ecosystem was developed on a robust local development setup, enabling us to simulate a virtual Ethereum blockchain environment. This was essential for iterative testing, debugging, and developing the components of the smart contracts.

We used several tools during the implementation. The Truffle framework and Ganache provided a smooth workflow, including built-in smart contract compilation, linking, deployment, and binary management. Truffle also greatly facilitated the creation of migration scripts, which are JavaScript files that help deploy contracts to the network. These scripts allow for sequential deployment of smart contracts, essential for managing dependencies between contracts and ensuring that the deployment process is reproducible. We used Ganach as the local virtual blockchain network. We will go into details later in the report 5.5.

Testing

After implementation, rigorous testing was conducted. Testing is a critical part of smart contract development due to the immutable nature of blockchain. We wrote our test scripts in JavaScript, taking advantage of the Mocha testing framework and Chai for assertions, which are both compatible with Truffle. The distributed nature of the EVM makes debugging a tedious task where printing works, but stepping through the execution of the code in runtime is impossible. Tests are essential. These tests were run against the smart contracts deployed on the Ganache local blockchain, allowing us to validate our logic in a controlled environment before deployment to the Sepolia testnet. The three contracts were deployed to the public decentralised Sepolia test network as part of the implementation. The booking management contract can be viewed and verified at: Link to booking management contract. The Evaluation chapter will delve deeper into our testing and the deployment of the smart contracts 5.5.

Deployment

Throughout its development, we deployed all smart contracts on three platforms - Ethereum Testnet Sepolia, Remix Shanghai virtual machine, and Ganache local VM- with each platform providing unique insights and contributing to their development. Initially deployed in a controlled environment, Truffle Suite

and Ganache enabled rapid prototyping and testing. Subsequent deployments on Remix and the Shanghai VM were instrumental in advanced debugging and performance optimisation since Remix offers a virtual debugging session. Finally, deploying to the Sepolia testnet, which closely resembles the Ethereum mainnet environment. The deployment to Sepolia played a critical role in enhancing and ensuring the robustness of the contracts. In the evaluation chapter, we conducted a thorough assessment of contract execution, focusing on the intricate details of efficiency and cost of execution (chapter 5.5).

Maintenance

The final phase of SDLC, The Maintenance phase, is left to our section on future developments in this thesis, which will be discussed thoroughly in Future Work section 7.6.

By employing the SDLC framework, the thesis aims to deliver a robust and comprehensive understanding of the development of Aquilier, showcasing how structured and methodical approaches in software engineering can be effectively applied in a practical software development project.

1.4 Scope and Limitation

The primary scope of this thesis is focused on addressing property access challenges within the short-term rental market through the innovative application of blockchain technology and smart contracts. The objective is to explore and demonstrate how smart contracts can enhance property access's security, efficiency, and transparency in the short-term rental domain.

While acknowledging the importance of privacy in digital transactions, this aspect remains orthogonal to the core focus of this thesis and, as such, will not be addressed in detail. It is recognised that privacy is a critical component of digital interactions, but its complexities and implications fall outside the narrowed scope of this study.

Additionally, while this research leverages essential tools and protocols such as HomeAssistant, Decentralised Identifiers (DID), and Verifiable Credentials (VC), these components are considered operational elements rather than primary subjects of analysis. Their role in the overall system is acknowledged, but the thesis does not delve deeply into these aspects. These tools and protocols are instrumental in facilitating the implementation and functionality of the proposed smart contract solutions; however, the focus remains on how blockchain and smart contracts specifically address the identified challenges

in property access.

This thesis aims to investigate and articulate the potential of blockchain and smart contracts in revolutionising the process of property access. This involves evaluating the effectiveness of smart contracts in creating a more secure, transparent, and efficient environment for property access, thereby contributing to the broader conversation of technological innovation in the real estate sector.

1.5 Context

The master project is conducted in affiliation with the Cyber Security Group (CSG). The research group is dedicated to exploring fundamental systems problems with a strong foundation in practical application domains. The group's research centres on efficient, scalable, transparent, and secure distributed systems, which form the key digital services in modern society.

The CSG is affiliated with the Corpore Sano Centre, a research entity that drives high-impact life sciences research at the crossroads of computer science, sports science, and medicine. This unique intersection allows the Corpore Sano Centre to focus on interdisciplinary research, including elite sports performance development, injury prevention, preventive healthcare, large-scale population screening, and epidemiological health studies.

The latest research by CSG involves:
"Capturing Nutrition Data for Sports: Challenges and Ethical Issues", Sharma et al. [9]. This paper explores the challenges and ethical considerations of using Image-Based Dietary Assessment (IBDA) and AI methods for capturing nutrition data in athletes, highlighting the need for accuracy and legal compliance in nutrition and sports science research.

"Scalable Infrastructure for Efficient Real-Time Sports Analytics" Johansen et al. [10]. This research focuses on the development and deployment of PM-Sys, a smartphone-based athlete monitoring system, demonstrating promising outcomes in enhancing athlete performance and the effective use of artificial intelligence for predictive analysis in sports.

"Dutkat: A Multimedia System for Catching Illegal Catchers in a Privacy-Preserving Manner" Nordmo et al. [11]. This paper presents the design of a privacy-preserving AI surveillance system for monitoring professional fishing activities, which balances legal privacy concerns with the need to capture forensic evidence of illicit activities and operate efficiently by reducing data transfer over

satellite networks.

1.6 Contributions

This thesis makes several key contributions to property management and blockchain technology. The primary contributions are as follows:

Market Analysis

A comprehensive market survey was conducted, encompassing an analysis of 15 rental platforms and key governmental documents. This survey provides insights into market trends, regulatory perspectives, and technological gaps within the short-term rental market. The findings of this survey contribute to a deeper understanding of the current market dynamics and the evolving needs within this sector.

Aquilier

The development and implementation of 'Aquilier', a decentralised application, represent a significant contribution. Aquilier is designed to address the identified challenges in the vacation rental market, leveraging the robustness of blockchain technology and the precision of smart contracts to manage and secure physical access to rental properties. This system demonstrates a novel approach to enhancing trust and security in digital rental transactions.

These contributions aim to advance the discussion and development of technology-driven solutions in the realm of property rentals, focusing on utilising blockchain to improve operational efficiency, security, and trust.

2

Background and Related Work

2.1 Smart Contracts and Decentralized Applications

A smart contract, in essence, is a self-executing contract with the terms of the agreement directly written into lines of code. Unlike traditional contracts, where legal mechanisms enforce terms, smart contracts enforce these terms by cryptographic code. This means that once the contract conditions are met, it executes the specified actions automatically without intermediaries.

The concept of smart contracts is not entirely new. Nick Szabo, a computer scientist and cryptographer, introduced the idea of digital contracts that can be embedded in digital assets in 1994 [12]. Szabo described how these contracts could be autonomously executed contractually. Later, in 1997, Szabo published the paper titled “Formalizing and Securing Relationships on Public Networks” [13]. In this work, he emphasised the potential and versatility of smart contracts through various clauses.

Szabo pointed out that:

> Smart contracts go beyond the vending machine in proposing to embed contracts in all sorts of property that is valuable and controlled by digital means [13, p. 2].

Nick Szabo's vending machine analogy offers an insightful perspective into the essence of smart contracts. As a vending machine autonomously functions, validating coins and dispensing items without human intervention, smart contracts execute preset actions once specific conditions are met. This seamless operation minimises the need for intermediaries, guaranteeing that contract stipulations are upheld.

With the advent of blockchain technology, the scope of smart contracts has grown tremendously. Blockchain presents a decentralised and secure space where contracts are self-implementing, resistant to tampering, and transparent. Every network participant can ascertain the legitimacy and performance of a contract. Consider the straightforward workings of a vending machine, where integrated hardware and software rules direct the transaction. Transform that notion into a digital context, visualising the vending machine in an abstract light. The conventional machine accepts a form of payment, such as a coin, and dispenses the desired beverage based on user selection. In the transition to a digital domain, new complexities arise. Issuing a digital beverage is challenging or may not resonate with the values of most humans. However, digital assets like keys or proof of ownership can be digitalised and issued. Since the vending machine is now digital, any conditions can be scripted. The machine is now scalable and can manage multiple contracts and complex conditions, surpassing a physical vending machine's capabilities.

The rise of blockchain technology enables trust integration since the machine can join an extensive network of similar devices, validating, sharing, and recording their transactions. This interoperability means smart contracts can be applied across numerous scenarios like financial transactions, property transfers, and tickets, to name a few. Ultimately, smart contracts promise to transform these operations, introducing revolutionary automation, enhanced security, and scalability.

2.2 Ethereum Blockchain Fundamentals

Ethereum is a decentralised blockchain network derived from the Ethereum White paper [14] published by Vitalik Buterin in 2014, who founded Ethereum. Essentially, Ethereum is designed to be the next-generation smart contract plat-

form capable of executing more intricate transactions and operations than Bitcoin. While Bitcoin, as introduced by Satoshi Nakamoto in his whitepaper [15], primarily functions as a decentralised digital currency and a store of value, Bitcoin is the most extensive system for digital assets. In contrast, Ethereum extends beyond this foundational role by aiming to support DApps. Ethereum introduces the innovative concept of smart contracts and decentralised applications, which opens the door to many use cases beyond only transferring and holding digital assets.

2.2.1 Ethereum Virtual Machine-EVM

The EVM serves as the runtime environment for smart contracts on the Ethereum blockchain. Often described as the operating system of Ethereum, the EVM is responsible for executing and maintaining smart contracts. Hildenbrandt et al. [16] demonstrates that improper use of structures and storage can lead to significant security issues. Implementation of smart contracts executing on the EVM can lead to significant property losses, which is a vulnerability. The ability of Ethereum to execute intricate transactions and applications is fundamentally anchored in the capabilities of the EVM. The EVM is a Turing-complete virtual machine that allows anyone to execute arbitrary EVM Byte Code. In other words, the runtime environment for every smart contract on the Ethereum network. Every Ethereum node runs on the EVM to maintain consensus across the blockchain.

Unlike Bitcoin, which has a limited scripting ability primarily for transferring coins, Ethereum's smart contracts can encode any computational function. This means developers have the flexibility to create a wide range of applications, from decentralised financial tools to games and beyond.

With this flexibility, challenges arise to ensure that code execution doesn't burden the network. Ethereum addresses this through the concept of *gas*. Every operation, from simple transactions to complex smart contract interactions, requires a certain amount of gas to execute. Gas is a measure of computational work and serves as a fee mechanism. By requiring users to pay gas to execute operations, Ethereum ensures that the network remains efficient and discourages wasteful or malicious computations [16].

The versatility of the EVM and the gas mechanism allow Ethereum to expand the boundaries of blockchain applications, offering capabilities far beyond being a digital currency.

2.2.2 Ethereum State and Storage

Smart contracts on Ethereum have their own storage, which is persistent across function calls and transactions. This storage is an integral part of the Ethereum state and is used to keep the contract's state. As discussed by Luu et al. [17] in their work titled "Making Smart Contracts Smarter", properly handling and updating this storage is crucial for contract security, as vulnerabilities in this aspect can lead to significant exploits.

Each transaction the EVM executes leads to a state transition. This transition involves computational work that is quantified in terms of gas. The gas system is designed to ensure fair compensation for the computational effort contributed by the miners. This mechanism is vital for the security and efficiency of state transitions in Ethereum [16].

Solidity data structures and mappings

Ethereum smart contracts commonly use data structures like mappings and structs to manage and organise data efficiently. Mappings in Ethereum are key-value stores used in smart contracts for efficiently storing and retrieving data. They are similar to hash tables. They associate unique keys with corresponding values. In Ethereum, mappings are particularly useful when linking addresses to balances or states. It is crucial to manage mappings carefully, as improper handling can lead to vulnerabilities in contract logic,

Structs in Ethereum are custom-defined types that allow the grouping of several variables. In Ethereum, structs are used to create complex data types that can represent various entities like users, agreements, or assets. They provide a way to model real-world entities and relationships in a more structured manner. The correct definition and use of structures are essential to clearly and securely implement contract logic [16].

Security Considerations: While mappings and structs are powerful tools for data organisation in smart contracts, their usage must be carefully considered from a security perspective. Ying Fu et al. "EVMFuzzer: detect EVM vulnerabilities via fuzz testing" [18]. Emphasises that vulnerabilities in smart contracts can arise from incorrect or inefficient use of these data structures, leading to potential exploits and inconsistencies in contract behaviour.

2.2.3 Gas and Gwei in Ethereum

Gas in the Ethereum network is a unit that measures the computational effort required to execute operations. It is essential for the network's operation, sim-

ilar to how a car needs gasoline. Gas in Ethereum ensures that the network's transactions, which require computational resources, are not susceptible to spam or infinite loops. The cost of these computations is covered by gas fees, calculated as the amount of gas used multiplied by the cost per unit of gas. This fee is mandatory for all transactions, successful or not [19].

Gwei, short for 'giga-wei', is a denomination of Ethereum's native currency, Ether (ETH). One gwei equals one billion wei (the smallest unit of ETH, named after Wei Dai, the creator of b-money). Gwei is commonly used to quote gas prices, where one gwei is 0.000000001 ETH or 10^{-9} ETH. This denomination makes it easier to understand and calculate gas costs without dealing with very small numbers[20].

Each Ethereum block has a base fee acting as a reserve price. The offered price per gas must at least match the base fee, for a transaction to be eligible for inclusion in a block. The fee is calculated based on the size of previous blocks compared to the target block size, helping to make transaction fees more predictable. The base fee is burned upon block creation, removing it from circulation and preventing indefinite high block sizes.

2.3 Ethereum Block Time and Real-Time Implications

Ethereum operates on a blockchain protocol where transactions are grouped into blocks. These blocks are added to the blockchain at regular intervals, a process known as block time. Unlike real-time measurements that are consistent and linear, Ethereum block time is a probabilistic measure and is subject to fluctuations. Typically, Ethereum aims for a block time of approximately 13-15 seconds. However, this is not a fixed interval and can vary based on network conditions and the difficulty adjustment algorithm [21].

A significant characteristic of Ethereum is the variable unit of time. Due to network congestion and the computational complexity of mining, the time it takes to add a new block can change. This variability poses unique challenges, especially when aligning blockchain operations with time measured in real-world events. In the context of Ethereum smart contracts, this variability necessitates a design approach that accommodates the approximate nature of block time. For instance, contracts dealing with time-sensitive operations must account for that block time does not precisely align with real-world time. Developers often use block numbers instead of timestamps for operations that depend on time measurements, acknowledging the approximate nature of block

time. As Ethereum continues to evolve, with updates and enhancements like the transition to Ethereum 2.0 and the adoption of proof-of-stake, block time dynamics may also change. In summary, the block time of Ethereum is a crucial aspect of the architecture in blockchain-based applications, impacting how smart contracts are designed and executed. Understanding the probabilistic and variable nature is essential to developing or interacting with applications on the Ethereum network.

2.4 Decentralised Identifiers and Verifiable Credentials

A user initiates their decentralised identity model by generating a DID, a unique and self-sovereign identifier that links to a DID document containing cryptographic keys.

An issuer, such as an educational institution or employer, provides the user with a VC. These credentials, containing claims about the user, are secured with the issuer's digital signature. Users store these credentials in secure digital wallets, ensuring they are available for sharing. Users who need to verify their identity share the VC with a verifier, accompanied by proof, usually a digital signature linked to their DID. Allowing verifiers to authenticate the VC by cross-referencing the digital signature with the public keys in the issuer's DID. Ensuring the credential legitimacy and current validity. The streamlined process underscores a shift towards a more secure, user-centred approach in digital identity verification.

DID and VC represent a paradigm shift in digital identity and authentication away from centralised services. DIDs provide a mechanism for creating globally unique identifiers without reliance on a central registry, fostering an environment where trust is distributed. Conversely, VC enables the issuance, sharing, and verification of credentials in a secure and privacy-preserving manner.

According to C.Brunner et al. DID and VC:Untangling Decentralized Identifiers and Verifiable Credentials for the Web of Trust [22], the workflows for creating, sharing, and verifying VC are integral to their utility. The trust model underlying DID and VC presents unique challenges. Establishing a link between a DID and a real-world identity involves complexities and privacy concerns. Managing the cryptographic keys is critical for the security of DID and introduces usability challenges.

/3

Qualitative Market Analysis

This chapter systematically analyses platforms offering short-term rental properties, focusing on operational systems. The investigation encompasses both a governmental perspective and an in-depth analysis of a selection of platforms, examining their services and capabilities.

The analysis aims to understand and comprehend the short-term rental market. Map out the needs and services required by our system. This thesis adopts a mixed-methods [6] approach with a primary focus on qualitative analysis, supplemented by quantitative elements extracted from our data. Our analysis of 15 platforms and official documents from the US and EU provided deep insights into the current state of short-term real estate rentals.

The following sections will delve into the specifics of our methodology, including how qualitative data was quantified and the synthesis of this information to form a cohesive analysis.

3.1 Official Government Perspectives

This section of the thesis explores the perspectives of official government bodies in both the European Union and the United States on the short-term rental market, specifically examining the impact on housing affordability and regulatory responses.

In the European Union, the research document from the European Parliamentary Research Service [1] outlines several prominent issues in the short-term rental market, including the impact on housing markets property price inflation and displacement of long-term residents; challenges in balancing the needs and interests of stakeholders such as property owners, tenants, and local communities; the necessity for harmonised regulations across member states; concerns about unfair competition with traditional hospitality businesses; and the need for transparent and secure data handling in online rental platforms. These issues underscore the focus of the European Union on creating a balanced and well-regulated short-term rental market. The research states that an easier and harmonised system for data collection is needed.

Similarly, in the United States, local governments have experienced the impact of short-term rentals on affordable housing [2]. Official sources, such as the Internal Revenue Service and municipal councils, acknowledge the challenges posed by the proliferation of these rentals. Regulations vary widely across states and cities, but there is a common thread of attempting to balance the interests of property owners, residents, and communities while ensuring housing availability and affordability. The documents reveal that while the regulatory approaches differ, both the EU and the US are seeking solutions to the complex challenges presented by the short-term rental market.

3.2 Platform and services

Our market analysis methodically assessed 15 leading services and platforms, chosen for their significant presence in the global vacation rental market and their widespread popularity. This strategic selection facilitated a detailed investigation into each service as we can see in the table: (3.1). Focusing on these prominent platforms allowed us to thoroughly understand the dynamics and trends shaping the vacation rental sector.

This targeted selection was aimed at acquiring a representative cross-section of the market. The chosen platforms are diverse in terms of their operational models, customer base, geographic focus, and the range of services offered. Such diversity ensures a comprehensive understanding of the current landscape in

vacation real estate listings, especially in terms of how these platforms manage secure access control and booking management.

The analysis of these platforms provided valuable insights into prevailing trends, operational challenges, and potential areas for improvement in the vacation rental industry. By focusing on the most influential and widely used platforms, the study aims to capture the broader dynamics of the market while also paying attention to specific regional topics, particularly in Europe. We posed two research questions to deepen our understanding of the current vacation rental market about smart home technology integration, and platform specialisation and user experience.

1. How does the platform incorporate smart home technologies, specifically in terms of access control and guest management?

2. What are the distinctive features or services that define the platform's specialisation, and how do they enhance the user experience?

The first question is designed to explore the extent and nature of smart home technology integration, with a particular focus on features that enhance security and streamline property management. The second question aims to uncover each platform's unique offerings and how they contribute to creating a competitive edge in the market.

Through these questions, we sought to identify potential market gaps in smart home integration, especially in access control solutions, and to gain a comprehensive view of how each platform differentiates itself in the competitive landscape of vacation rentals.

3.2.1 Smart Home Integrations in Vacation Rental Platforms

In our study of vacation rental platforms, we observed a growing trend in the integration of smart home technologies, particularly smart locks, to enhance security and guest experience. Here are some notable examples:

AirBnB announced plans to integrate smart locks in the US and Canada with support for specific brands like Schlage, August, and Yale. This integration will enable hosts to automatically generate unique access codes for each reservation, accessible to guests via the Airbnb app. Additionally, discussions on various forums indicate that third-party smart home devices currently offer integration through export calendars, facilitating unique access code generation for hosts [38].

Platform	**Key Characteristics**
Airbnb [23]	Global reach, community-driven approach
Booking.com [24]	Wide range of accommodations, user-friendly booking
Rentalia [25]	European-based, variety of property types
Home2Book [26]	High-quality properties, guest experience focus
VRBO [27]	Wide range of vacation rentals, US market presence
Expedia [28]	Hotel and vacation rentals, travel accommodation insights
HomeAway [29]	Extensive international listings
TripAdvisor Rentals [30]	User reviews integrated with rental listings
FlipKey [31]	Variety of rentals, user review integration
Agoda Homes [32]	Private accommodations, Asia-Pacific market insights
TUI Villas [33]	European focus, wide property range
Homestay.com [34]	Local home stays, personalized rental experiences
Zillow Rentals [35]	Long-term rental insights, contrast with vacation rentals
Trivago [36]	Price comparison across various sites, including rentals
HomeToGo [37]	offering comprehensive comparisons and user-friendly search

Table 3.1: Analysis of Vacation Rental Platforms

Another example is Booking.com, which provides recommendations and support for hosts to use smart locks and security devices, though it does not offer direct integration. Hosts are advised on features such as single-use codes, remote unlocking via apps, app analytics, and connectivity considerations. The platform also allows hosts to indicate the presence of smart locks on their property listings [39].

3.3 Findings and Discussion

After analysing both the platforms and government documentation, it is apparent that there is a pressing need for seamless integration between listing platforms and smart home systems. The data also suggests that governments are seeking a uniform approach to managing user data. Furthermore, there is a notable lack of efficient systems for overseeing taxation and local market regulations. This analysis underscores significant technological gaps, particularly in integrating blockchain technology with smart home systems to enhance security and efficiency in property management. We can also identify notable technological gaps, particularly in the integration of blockchain technology with smart home systems for enhanced security and efficiency in property management. The analysis reinforces our initial hypothesis regarding existing gaps and deficiencies in the market, which could be addressed to meet market demands better.

Smart Home Technology Integration

Research question 1 reveals that there is a lack of seamless integration for automated access in smart systems. Our analysis found a notable absence of seamless integration between the examined rental platforms and smart home systems, particularly in the context of automated access. Despite technological advancements in both domains, we found no evidence that current systems offer the full scope of automatic access approval that could potentially be accomplished with advanced integration. This gap underscores the untapped potential of blockchain technology in facilitating this integration, thereby offering enhanced security and trust in access control systems.

Platform Specialisation and User Experience

Research question 2 identified multiple challenges in governmental regulation and taxation enforcement. Our analysis of official documents from the EU and US revealed significant challenges faced by governments in enforcing regulations and taxation in the rental market. This has contributed to elevated rental prices in many areas. The potential of a unified blockchain interface, with the inherent transparency properties and public ledger system, could be transforma-

tive in this context. Such a system could streamline regulatory compliance and taxation processes, potentially stabilising rental market prices and enhancing governmental oversight.

Additional findings

Operational inefficiencies in property management current systems and platforms often require manual interventions, leading to over-managed and time-consuming processes. Hosts frequently need to coordinate obvious tasks such as cleaning and general maintenance despite their simplicity, resulting in significant inefficiencies. The integration of smart contracts with smart home systems offers a promising solution by automating property management tasks, thus streamlining operations and enhancing efficiency for both owners and renters.

The last finding is regarding security and synchronisation challenges. Our research indicates a significant fragmentation in the rental market due to the vast amount of platforms available to hosts. Many hosts limit their listings to only one or two platforms despite the potential benefits of broader exposure, primarily due to the challenges in synchronising calendars and pricing across multiple services. Additionally, the lack of standardised market practices leads to the risk of less secure or potentially malicious platforms. Blockchain and smart contract technology could address these issues by offering a foundational platform. Utilising the inherent security of cryptography, blockchain could facilitate synchronised data management across various platforms, establishing more unified standards and enhancing overall market security.

3.4 Ethical and Security Considerations

Given the sensitive nature of granting someone access to a property, often one's own home, a particular emphasis was placed on the ethical and security aspects of the Aquilier system. The adaptation of blockchain technology with smart home control systems raises several important questions and concerns. These issues are intricately linked to user privacy, data security, and the ethical implications of automated access control.

In this thesis, while a comprehensive solution to these challenges is beyond the scope of our current work, they are thoroughly discussed, and their potential resolutions are suggested for future research. Key areas of concern include:

Data Privacy

The integration of smart home technology with blockchain requires measures to ensure the privacy and security of user data. This consideration involves protecting personal information related to home access, movements, and habits and ensuring compliance with data protection regulations.

Security of Smart Contracts

Given the central role of smart contracts in this implementation of access control, addressing potential vulnerabilities is essential. This involves implementing robust security protocols and conducting thorough audits and testing to prevent unauthorised access.

Ethical Use of Technology

The automated nature of blockchain-based access control raises ethical concerns regarding fairness and transparency. A future scenario where Aquiliers integrated home system with AI technology. It is imperative to evaluate and ensure that the system's decision-making processes are free from bias and that users understand how access decisions are made.

Responsibility and Accountability

With automated decision-making, delineating responsibility in case of system errors or misuse is complex. This aspect involves exploring the accountability of system developers, property owners, and the blockchain network, ensuring that there are clear protocols for addressing and rectifying any issues that arise.

These considerations are crucial for the responsible deployment and use of technologies like Aquilier. Although not fully resolved within the scope of this thesis, it is important that they are addressed and carefully considered in future development. The discussion of these ethical and security considerations lays the groundwork for future work to build upon, ensuring that the Aquilier system not only advances in technical capabilities but also aligns with the highest standards of ethical responsibility and security.

4

Requirements and Specifications

In this chapter, we methodically outline the requirements of Aquilier, detailing the specific requirements and specifications that emerged from our comprehensive market analysis. As elaborated in Chapter 3, this analysis identified key gaps and potential improvements in the current market offerings. We focused on addressing these gaps by defining features that would enhance security and user experience in real estate rentals, considering that our main focus is giving access to property via smart contracts.

4.1 Functional Requirements

Requirement 1. A user should be able to request access to a property they have booked.

Requirement 2. Each user must have a unique identifier, a name or username, contact details, phone number, email, and physical address.

Requirement 3. Users should be authenticated to ensure secure access to their accounts and transactions.

Requirement 4. A structured set of attributes must uniquely represent each

property. This includes a unique identifier, relation to owner, property title, location, daily cost of renting and a boolean indicating if the property is Active.

Requirement 5. A booking must have a relation to the property that is booked. It has to be uniquely identified, the definition for the booked time period, a reference to the user holding the booking, and an indication if it is active.

The first requirement implies several subsequent requirements, a representation of users and properties. A well-defined user and property model is essential to manage and ensure secure transactions effectively. The requirements for user representation ensure a seamless and secure user experience and lay the foundation for reliable and efficient synchronised access management. Subsequently, the first requirement also implies a structured representation of booking and properties.

4.2 Non functional Requirements

Requirement 6. Security User data must be securely stored and handled. Additionally, we need to ensure that only the correct users can access the property.

Requirement 7. User data should not be distributed to others other than itself, for example, when the user accesses the property, and personal information must be kept secret.

Requirement 8. The system has to execute efficiently. If a user has to stand and wait several seconds for the door to unlock, the system will be perceived slowly.

Requirement 9. We assume the cost has to be reasonable, looking at a month or per tenant. If the cost is higher than 10 USD for the entire booked period or per month, the cost is perceived as unreasonably high.

5

Design and Implementation

This chapter provides an in-depth exploration of the design and implementation aspects of the Dapp Aquilier, which has been developed to tackle the intricate challenges identified in the analysis Chapter 3. Aquilier is a proof-of-concept application specifically engineered to facilitate access to physical properties through the innovative use of smart contracts.

5.1 Overview

Aquilier consists of several components and applications. The three smart contracts illustrated furthest up on Figure 5.1 are central components. They also act as a database since they write all data to the Ethereum distributed ledger. The smart contracts have an inherent Application Programming Interface (API) since all data written to the ledger data can be queried or read. Additionally, we developed a proxy server written in Golang, illustrated in the centre of our system architecture in Figure 5.1 that acts as a bridge between the Ethereum smart contracts and the smart home system. We developed two different ways of triggering methods on the smart contracts, one via a frontend developed using the react framework and Typescript, and the second a test script written in javascript.

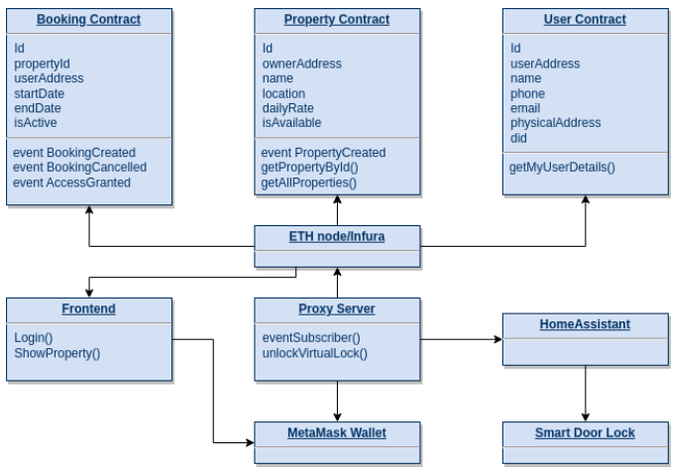

Figure 5.1: System architecture and data flow of Aquilier

For both the frontend and the Proxy, it is imperative to maintain a persistent connection to the Ethereum network. This can be achieved by either running an Ethereum node with a known IP or integrating the system towards a service that maintains this connection. In this initial phase, we decided to use the service Inrura in favour of running our own Ethereum node.

Utilising the Infura service is a suitable and resource-efficient alternative to self-hosting our Ethereum node. The Proxy server maintains the connection to the smart home system where we adopted the home Assistant docker image. In the home assistant container, we have defined a virtual lock to represent a physical door lock. This holistic approach underscores the commitment to building a robust and versatile DApp capable of addressing real-world challenges.

Throughout the development, we focused on leveraging well-established frameworks and technologies supported by extensive documentation and prior research. The choice of the Ethereum blockchain was particularly influenced by the widespread adoption and maturity of its programming language, Solidity, within smart contract development communities.

The following sections will delve into technical details essential to the implementation.

5.2 Smart Contracts

The three smart contracts form the pillars of the Aquilier system, serving as the key components for state management. These contracts are architected

to execute on the Ethereum blockchain, ensuring that interactions between the smart contracts, the Proxy, and the smart home system are conducted securely. Every operation written on the Ethereum network has a cost in the form of gas [19]. As a result, we have to write as efficient code as possible, and where operations can be executed outside the network, the so-called off-chain transactions should be considered.

In the context of Ethereum blockchain operations, data retrieval transactions or read operations do not incur any gas fees. This is because they do not necessitate state changes on the blockchain, which would require mining and, consequently, the cost of computational resources. Consequently, for operations that involve summarising or aggregating data, such as collating details of property instances, it is more cost-effective to execute these computations off-chain.

5.2.1 Property Registry

Property management is the smart contract primarily responsible for managing and storing information related to properties. Every time a property is added, removed, or modified by a user, the Property Registry smart contract captures and retains the state of these changes.

Central to the contract is the createProperty function, which allows property owners to register new properties on the blockchain. This function instantiates a new Property struct, assigns a unique ID based on the current number of properties listed, and sets the property as available. Upon successful creation, the contract emits a PropertyCreated event, which serves as a transparent ledger entry signalling the addition of a new property to the platform. This event logs the property's ID, owner's address, name, location, daily rate, and availability for the benefit of subscribers and listeners interested in real-time updates.

For retrieval of property details, the contract offers a getPropertyById function, which, given a property ID, returns the corresponding Property struct. This function ensures that any participant can verify property details in an immutable and trustless manner. Additionally, the contract provides a getAllProperties function that returns an array of all registered properties. This function, while comprehensive, is noted for its potential to be gas-intensive, indicating that it might require a significant amount of computational power and, therefore, incur higher transaction costs when used.

The property struct is a custom data type designed to store all relevant information about a property. Property Struct:

```
struct Property {
    address ownerAddress;
    string name;
    string location;
    uint256 dailyRate;
    bool isAvailable;
}
```

The Property struct within Aquilier encapsulates the essential attributes of a rental property on the blockchain. It associates property with its owner's Ethereum ownerAddress, ensuring that transactions, such as rental payments, are linked to the rightful recipient. The name and location fields provide descriptive identifiers, the former for recognition within the platform and the latter specifying the property's physical address. The dailyRate is set as an integer value, delineating the cost per day, which facilitates financial transactions and pricing transparency. Lastly, the isAvailable boolean flag signifies the property's current rental status, enabling dynamic booking availability management. This structure is simple yet fundamental for basic operations in the system, offering a secure and efficient mechanism for property listings.

These structures, functions and events form a robust interface for property management, leveraging the inherent benefits of blockchain technology—security, transparency, and immutability—to streamline operations within the real estate domain.

5.2.2 User Registry

The UserRegistry contract is a cornerstone of the Aquilier platform, providing a decentralised framework for managing tenant identities.

User Struct:

```
struct User {
    address userAddress;
    string name;
    string phone;
    string email;
    string physicalAddress;
    bytes32 did;
}
```

It defines the placeholder for users in Aquilier that associates a user's Ethereum address with personal and contact information, including name, phone number,

email, and physical address. Additionally, it employs a unique bytes32 decentralised identifier (DID) for each user, generated via a keccak256 hash of the user's Ethereum address, which enhances privacy and security.

Users are stored within a mapping, linking Ethereum addresses to corresponding User structs. The getMyUserDetails function allows users to retrieve their information, ensuring they have access to their data as stored on the blockchain.

The contract provides an addUser function for users to register their details. This public function is a gateway that internally calls addUserInternal, an internal function that encapsulates the logic for adding user information to the blockchain. This separation of concerns via internal function usage ensures that user data is added securely and consistently.

The contract also defines a tenant struct with a DID and a Verifiable Credential (VC). While the DID is set upon tenant registration, the VC is assigned later, serving as a marker for tenant verification. The registerTenant function allows for the initial setup of tenant data with the DID, and the setTenantVC function, complemented by the TenantVerified event, completes the verification process by assigning a VC. The isTenantVerified function provides a simple check to confirm whether a tenant has a non-zero VC, which correlates to a verified status.

In essence, the UserRegistry contract utilises the inherited features of blockchain technology with identity management. It leverages smart contracts to create a secure, transparent, and verifiable system that manages tenant identities. This contract ensures that user data integrity is maintained. Through the use of DIDs and VCs, it aligns with the principles of self-sovereign identity and trustless verification in the blockchain realm.

5.2.3 Booking Management

The BookingManagement contract in Aquilier is designed for handling bookings, offering functionalities to create, cancel, and verify booking statuses. It introduces a Booking struct encapsulating essential booking details like property ID, user address, start and end dates, and active status.

A key feature of this contract is mapping booking IDs to Booking structs, alongside a nextBookingId counter to ensure unique identifiers for each booking. The contract emits events such as BookingCreated, BookingCancelled, and AccessGranted to log significant actions on the blockchain. The createBooking function allows users to initiate bookings, incrementing nextBookingId with each new entry and emitting a BookingCreated event. The cancelBooking func-

tion enables users to cancel their bookings, provided they are the ones who initiated the booking and the booking is still active, triggering a BookingCancelled event. The checkBookingStatus function allows querying the current status of a booking, returning true if the booking is active and within the valid date range.

This contract is central to Aquilier's booking system, providing a secure and efficient mechanism to manage property rentals on the blockchain, ensuring transparency and user autonomy in the rental process.

```
struct Booking {
    uint256 propertyId;
    address userAddress;
    uint256 startDate;
    uint256 endDate;
    bool isActive;
}
```

The booking struct comprises several fields that capture the essential details of a booking transaction. The propertyId field holds a unique identifier, ensuring each property can be distinctly referenced. The userAddress field stores the Ethereum address of the user who initiated the booking, linking the booking to a specific user within the blockchain's secure environment. Additionally, the startDate and endDate fields record the time frame of the rental period. Lastly, the isActive boolean flag is utilised to indicate the booking's current status, determining whether the booking is active. This structure serves as the data representation of a booking when querying the smart contract for a booking.

The Booking management contract holds the method for granting access to a property. This is implemented by using an event that is emitted each time someone requests access by calling the method:

```
1    function grantAccess(uint256 _bookingId) public {
2        require(bookings[_bookingId].isActive, "Booking is not
    active");
3        require(block.timestamp >= bookings[_bookingId].
    startDate && block.timestamp <= bookings[_bookingId].
    endDate, "Booking is not currently valid");
4        emit AccessGranted(_bookingId, bookings[_bookingId].
    userAddress, bookings[_bookingId].propertyId);
5    }
```

Listing 5.1: function grantAccess

This naive method illustrates how one can apply criteria for granting access.

In this case, we only declare two requirements.

In summary, the grantAccess function checks if a booking identified by bookingId is active and within the valid booking window. If these conditions are met, access is granted to the user associated with the booking, which is acknowledged by emitting an event that can be observed and acted upon by other parts of the system, in this case, the Proxy.

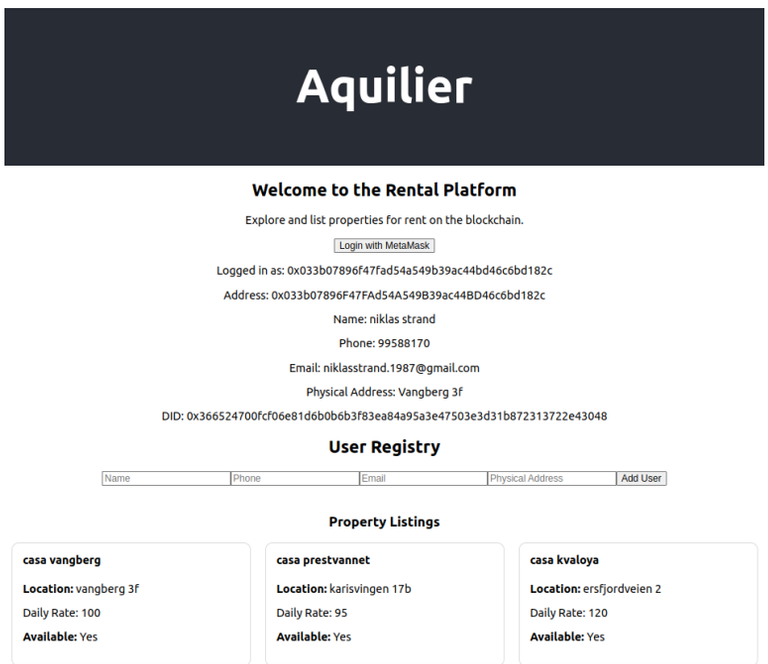

Figure 5.2: Aquilier frontend logged in

5.3 Frontend

The Aquilier system facilitates a frontend web application developed using React, a modern JavaScript library for building user interfaces. The design and development focused on creating a naive, functional, and responsive web application that can write and read data from the blockchain effectively. We have chosen Typescript in favour of JavaScript due to its strict typing and data structures.

5.3.1 User Interaction

Accessing the platform, users are greeted with a clean and straightforward interface encapsulating the application functionality as shown in Figure 5.2. The landing page has a button to log in with Metamask. When pressed, the user is forwarded to the Metamask extension, which will prompt the user to authenticate by confirming its access request in the Metamask popup extension window.

```
const handleLogin = async () => {
  try {
    if (typeof window.ethereum !== 'undefined'
                          && window.ethereum.request) {
      const accounts = await window.ethereum.request({
      method: 'eth_requestAccounts' }) as unknown as string
  [];
      if (accounts.length > 0) {
        setAccount(accounts[0]);
        alert('Logged in!');
      } else {
        alert('No accounts found.');
      }
    } else {
      alert('Please install MetaMask!');
    }
  } catch (error) {
    console.error(error);
    alert('Failed to login with MetaMask.');
  }
};
```

Listing 5.2: handleLogin

The code to log in via MetaMask is shown in listing 5.2. Since handleLogin is an async method, it sends the request to Metamask and expects a response at a later point. When the method has The handleLogin method is executed async. It checks the browser environment to see if an Ethereum object is present. MetaMask and other Ethereum-compatible browsers inject this object. Its presence indicates that the user has installed MetaMask or a similar wallet extension.

The **window.ethereum.request** property is a function that MetaMask provides to interact with users' Ethereum accounts. If MetaMask is detected, the function requests the MetaMask extension to get the users' Ethereum account(s). If the user approves, MetaMask will return an array of account addresses. The use of await means that the function will pause until MetaMask responds with the accounts or an error. Once MetaMask responds, the function checks if the array of accounts returned has at least one account. If it does, it means the user has successfully logged in. The first account in the array (accounts[0]) is typically the user's currently selected account in MetaMask.

If an account is found, the application's state is updated with the user's account address using a state setter function like **setAccount**. This would typically trigger a re-render of the component or update the user interface to reflect the user's logged-in status.

The user is then alerted with a simple browser alert that they have logged in successfully.

If MetaMask returns no accounts, the user is alerted that no accounts were found, which typically means they have not created an account in MetaMask or have not logged into it. Prompting MetaMask Installation.

If an Ethereum object is not present, it is likely that the user does not have MetaMask installed, and the user is prompted to install MetaMask to interact with the application.

```
useEffect(() => {
    if (account) {
      fetchUserDetails(account);
    }
  }, [account]);
  // This will run fetchUserDetails whenever
  // the account state is updated
```

Listing 5.3: useEffect

The **useEffect** React hook is instrumental in the application's lifecycle for invoking actions in response to state changes. In this instance, it monitors the account state. When a user authenticates via MetaMask, the account state is updated with the user's Ethereum address, triggering **useEffect**.

The hook then conditionally calls the fetchUserDetails function to retrieve the user's information from the smart contract. This process introduces a slight delay resulting from the time it takes to fetch data from the blockchain before the user's details are displayed on the frontend. Such delays are typical in dApps due to the decentralised nature of data retrieval from blockchain networks.

```
1  const fetchUserDetails = async (accountAddress: string) => {
2    try {
3      const detailsTuple = await contract.methods.
     getMyUserDetails().call({ from: accountAddress }) as
     unknown as UserDetailsTuple;
4      const details: IUserDetails = {
5        userAddress: detailsTuple[0],
6        name: detailsTuple[1],
7        phone: detailsTuple[2],
8        email: detailsTuple[3],
9        physicalAddress: detailsTuple[4],
10       did: detailsTuple[5],
11     };
12     setUserDetails(details);
13   } catch (error) {
14     console.error('Error fetching user details:', error);
15   }
16 };
17
```

Listing 5.4: fetchUserDetails

The **fetchUserDetails** function is an asynchronous operation employing an Application Binary Interface ABI to interact with a predefined smart contract on the Ethereum blockchain. The ABI serves as an interface between the smart contract and the application, dictating how calls to the contract are structured and interpreted. The ABI is a JSON file. All three smart contract has corresponding ABIs.

Upon invocation, the function uses the account address to formulate a request to the smart contract getMyUserDetails method. This method call is facilitated by the ABI, which specifies the necessary encoding of the call and the expected response format. The response, assumed to be a tuple conforming to the **UserDetailsTuple** structure, is then asynchronously retrieved from the blockchain.

The response tuple is mapped to a strongly typed object **IUserDetails**. This mapping is crucial for the type-safe representation of blockchain data within the application, enhancing the robustness of the frontend code. Each attribute of the user's details—such as their blockchain address, name, contact information, and physical address—is extracted from the tuple and assigned to the corresponding fields within the **IUserDetails** object.

For successful data retrieval, the user's details set the state using the setUserDetails function. This state update triggers a reactive UI refresh, presenting the fetched data to the user using the previously described **useEffect**. In the event of an error during this data retrieval process, the exception is caught, and an

error message is logged to the console, indicating a fault in the data-fetching operation.

In essence, the ABI is the critical component that defines the interaction pattern between the frontend application and the Ethereum smart contract, enabling the seamless retrieval and utilisation of on-chain data within the off-chain application context.

5.3.2 Property Management Interaction

In the execution context of Ethereum, data retrieval operations are generally free as they do not alter the blockchain state. The **fetchAllProperties** function encapsulates a critical feature of the Aquilier platform, acquiring property listings from the Ethereum blockchain. Leveraging the smart contract's ABI, this asynchronous function invokes the getAllProperties method, which retrieves an array of property entries per the smart contract's design.

```
1     const fetchAllProperties = async () => {
2         try {
3           const propertiesData = await contract.methods.
    getAllProperties().call() as unknown as Property[];
4           console.log(propertiesData);
5           setProperties(propertiesData);
6         } catch (error) {
7           console.error('Error fetching properties:', error);
8         }
9       };
```

Listing 5.5: fetchAllProperties

The fetchAllProperties function would not directly incur transaction fees or gas costs. This method retrieves the entire dataset of properties in a single call. While this is inherently not a gas-consuming operation, the underlying smart contract aggregates the dataset, something we point out in section: 5.2.1 and can potentially lead to a very costly operation. The design poses scalability challenges when considering a massive dataset, such as a million property listings. Each property entry adds to the storage requirements and the computational complexity of the retrieval operation on the blockchain. Furthermore, from a frontend performance perspective, loading a vast amount of data simultaneously can result in significant latency, impacting user experience. To mitigate such scalability issues, a more robust approach could involve paginated queries or indexed searches that only fetch a subset of the data from the blockchain. Alternatively, off-chain aggregation of the data set. This would provide a more scalable and cost-effective solution that maintains performance and adheres to the economic constraints of the Ethereum network.

While the fetchAllProperties function is a vital component of the system's data layer, its current implementation may not be optimised for the potential scale of the property database. A strategic redesign would be essential to ensure the platform's viability.

Communication with the Ethereum blockchain is facilitated by Infura, a service that provides a scalable and reliable API for interfacing with the Ethereum network. The frontend leverages this connection to fetch real-time data from the smart contracts, such as property details and user registry information, without the overhead of running an Ethereum node.

The state management capabilities of React are used to handle the dynamic updating of the user interface as blockchain data is retrieved. This ensures that the platform remains responsive and the displayed information reflects the current state on the blockchain.

Authentication is handled via MetaMask, which provides a secure and user-friendly way to manage identity and sign transactions. When a user logs in, MetaMask prompts them for permission to access their Ethereum account.

The integration with MetaMask also facilitates interaction with the smart contracts. When a user performs an action that requires a transaction, such as updating property information, the frontend constructs the transaction and sends it to MetaMask for user approval and signature. MetaMask then broadcasts the signed transaction to the Ethereum network via Infura.

5.4 Smart Home Integration

We have leveraged Home Assistant through a Docker container for the smart home system, which provides an isolated, controlled, and reproducible environment for running applications. Utilising Docker ensures consistent performance and streamlines deployment across different infrastructures without the hassles of manual configurations. The actual lock component in this implementation is a virtual lock defined inside the Home assistant container [40]. It can be either locked or open and represents a physical door lock. We have set up the virtual lock in a YAML file that the docker image uses while initiating the

HomeAssistant instance:

```
input_boolean:
  virtual_lock:
    name: "Virtual Lock"
    initial: off
sensor:
  - platform: template
    sensors:
      custom_virtual_lock:
        friendly_name: "Custom Virtual Lock"
        value_template: >
          {% if is_state('input_boolean.virtual_lock', 'on') %}
            Locked
          {% else %}
            Open
          {% endif %}
```

We chose Home Assistant due to its leading position in smart home automation platforms. Originating as an open-source project, it offers a wide variety of functionalities that enable users to manage and automate their smart devices seamlessly.

The Docker container for Home Assistant is derived from their official open-source project [41].

5.4.1 Proxy

The Proxy serves as the crucial bridge between the smart contract and the Home Assistant application. Implemented in Golang, it continuously monitors and listens to events emitted by the smart contract, focusing on those from the booking management contract. To ensure efficient and secure communication, the Proxy is deployed on a node that resides within the same network as the Home Assistant container. This strategic placement facilitates real-time responsiveness and minimises potential network-related delays or disruptions.

When the Proxy receives an event from the booking management contract, it acts on this by calling the method unlockVirtualLock. Before unlocking the virtual lock, the Proxy has to parse and validate the event. It does so by creating a filter that is then used in the query to subscribe to the event [42, 43]. The code facilitates both Infura and Ganache event subscribers, meaning that it is compatible with both local virtual blockchains via Ganache or any Ethereum network via Infura. In our case, we listen on the Sepolia network.

5.5 Tools and framework

We utilised Ganache, a component of the Truffle suite, for local blockchain simulation in our development process. Ganache offers a personal, local blockchain environment essential for deploying contracts, developing applications, and running tests efficiently. In this project, we utilised both its desktop application for its user-friendly interface and the command-line tool for greater flexibility in automated tasks.

Ganache significantly accelerated our development cycle, allowing for rapid deployment iterations and enabling easy inspection of transactions and blockchain states. Its dual functionality enhanced our development agility, providing both visual insights and scripting capabilities essential for an Ethereum-based project. In essence, Ganache's local blockchain environment proved critical for our application's development, ensuring a fast, flexible, and efficient testing and development process.

/6

Evaluation

This chapter will present the evaluation of our devised system Aquilier. We will present the results and findings encountered during the process. For simplicity and readability during the evaluation, we will refer to several hashes that will only be referenced as hash and can be found in the footnote. Each hash can be traced on any blockchain lookup tool. During the evaluation, we will use etherscan[1]. The primary objective of this project is to rigorously assess the feasibility of employing smart contracts and blockchain technology for the secure and reliable transfer of property access rights to tenants. We investigated this by implementing and developing a decentralised application, Aquilier, that harnesses the Ethereum blockchain to facilitate secure and transparent booking transactions, explicitly focusing on the mechanism granting property access via a smart door lock.

The Aquilier system comprises a suite of three smart contracts Booking Management, Property Registry, and User Registry, each written in Solidity. Together, these contracts facilitate the systematic creation, management, and oversight of users, properties, and the booking process, leveraging the robust features of Solidity for optimal contract performance and security.

1. etherscan: https://sepolia.etherscan.io

We deployed the contracts on three platforms during the development: the Ethereum Test net Sepolia, Remix Shanghai virtual machine and Ganach local VM. We will walk through all deployments in the evaluation since they provided different insights and played a role in the assessment. During the initial stages of development, the smart contracts were deployed in a controlled local environment, utilising the Truffle Suite for deployment and testing purposes, along with Ganache as a personal blockchain for Ethereum development. This setup allowed for rapid prototyping and testing, offering a simulated blockchain environment where transactions could be executed instantly without the costs associated with actual network deployment. This phase was critical for verifying the basic functionality and integrity of the smart contracts, enabling thorough debugging and refinement before moving to a more public and realistic blockchain test network.

The deployment of the contracts was also executed on Remix and the Shanghai virtual machine. This choice was motivated due to the sophisticated debugging capabilities and the opportunity for detailed inspection of the contracts' execution processes offered by these platforms. Such strategic deployment ensures an enhanced analysis and optimisation of contract performance, contributing to a more robust and reliable system implementation.

When deploying locally on Ganache or to the Shanghai VM provided by Remix, the gas consumed is purely virtual; thus, deployment can be conducted infinite times without incurring any economic impact on these networks. Later, when the contracts were deployed to the Sepolia testnet, it had an actual cost in the form of Sepolia Ethereum, which must be generated, unlike the tokens on the two virtual networks that can be replenished by simply restarting the networks. The Sepolia network operates as a persistent and interconnected web of nodes executing the Ethereum protocol, maintained by consensus within a community of developers. It is only by community consensus that this network is designated for testing and that the tokens within have no real-world value. The deployment of all three contracts on the Sepolia network was successful. The `BookingManagement` contract, for example, can be reviewed at the address[2]. The Proxy is designated to listen for events from the Booking Management smart contract, specifically for the event that signals when a tenant is granted access. This Proxy is implemented in Golang, leveraging the robustness and efficiency of a compiled language. Initially, the Proxy was programmed to monitor transactions solely on the local Ganache blockchain. Subsequently, it was extended to track the activity on the Sepolia network. This transition presented a challenge, as the Sepolia network's distributed and decentralised nature meant that without operating a full node, we lacked a direct entry point or knowledge of the network nodes' IP addresses. To address this, we utilised

2. address: 0x127f3D4914724011D4e59431397DDB85DE32B0c8

the Infura service as an intermediary to establish a connection with the Sepolia network. Consequently, the Proxy was extended to monitor the local blockchain environment and the external Sepolia network through the Infura connection. The Proxy is designed to be deployed on the same local network as the Homeasiste network, and the smart door lock is to secure the communication between them and lower the risk of malicious attacks

6.1 Deployment

Two approaches were adopted to verify the correct functionality of the contracts. Initially, tests were written in JavaScript, leveraging the capabilities provided by the Truffle framework. Subsequently, a script was developed to be executed within the Truffle console for direct interaction with the deployed contracts. This script is designed to engage with the BookingManagement contract: it deploys an instance, sets up start and end dates for a booking (a week apart), creates a new booking, and finally, attempts to grant access for the specified booking ID. The script used for this testing process:

```
let instance = await BookingManagement.deployed();

let startDate = Math.floor(Date.now() / 1000);
let endDate = startDate + (7 * 24 * 60 * 60); // One week from
    the current time
let createTx = await instance.createBooking(1, startDate,
    endDate);
let bookingId = createTx.logs[0].args.bookingId.toNumber();

let accessTx = await instance.grantAccess(bookingId);
```

Listing 6.1: Test script

This practical approach adds an extra dimension of validation, ensuring that the smart contracts perform as expected in an environment that closely mirrors the real-world operations on the live blockchain network.

6.1.1 Deployment Cost Estimation

Deploying the contracts Sepolia test-net was an essential step, as it closely mirrors a production environment and serves as a proxy for the Ethereum mainnet, offering a controlled setting for realistic deployment scenarios.

For deployment to the Sepolia network, it is necessary to have a wallet. We chose MetaMask [44] due to its widespread adoption and integration capabili-

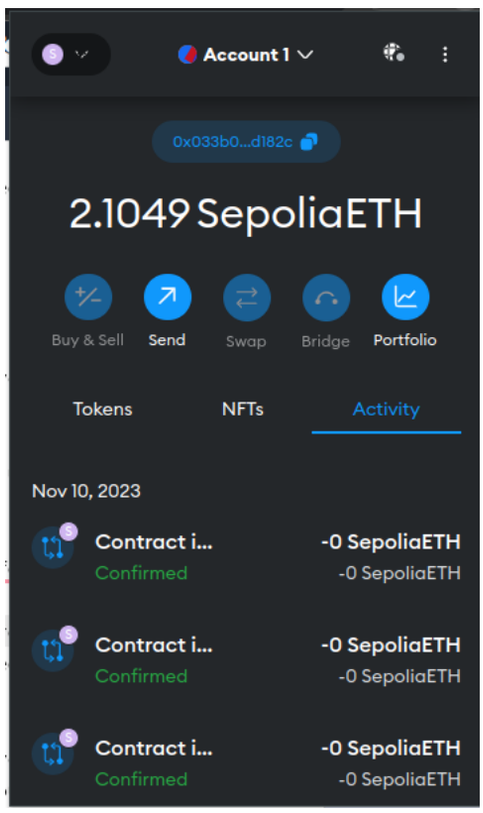

Figure 6.1: Screenshot of the MetaMask Wallet

ties. Figure 6.1 shows the logged-in view. The funds used on the network do not have any real value; we utilised a faucet to generate funds for the deployment [45]. We discovered that the Remix development environment provided a user-friendly and straightforward approach to deploying the contracts. We monitored the deployment transactions, documenting all associated costs and execution details.

From our observations, the transaction for deploying the contract had a cost equivalent to 158,109 gas units. To project this expense onto the Ethereum mainnet, it is necessary to consider the fluctuating nature of gas prices, which are influenced by network demand and are typically higher than those on testnets. Gas, a fundamental unit of computational measure on the Ethereum network, remains a constant value for a given transaction, irrespective of the network it is executed on. Therefore, the computational work, quantified as the gas used, would be identical on the Sepolia testnet and the Ethereum mainnet. The variable that distinguishes the cost between the two networks is the gas price, denoted in Gwei.

The cost of a transaction can be calculated with the formula:

$$\text{Transaction Cost} = \text{Gas Used} \times \text{Gas Price} \tag{6.1}$$

For the specific transaction conducted on the Sepolia testnet with a gas usage of 158,109 and a gas price of 2.500000018 Gwei, the transaction fee was computed as follows:

$$158,109 \times 0.000000002500000018 \text{ ETH} = 0.000395272502845962 \text{ ETH} \tag{6.2}$$

To simulate the same transaction on the Ethereum mainnet, the average gas price recorded on November 7, 2023, at 31.7641 Gwei, serves as a reference. Utilising this metric, the estimated cost on the main net would be:

$$158,109 \times 0.0000000317641 \text{ ETH} \approx 0.005019528749 \text{ ETH} \tag{6.3}$$

This exercise underscores the economic considerations of deploying and operating smart contracts on the Ethereum blockchain, particularly the implications of gas prices that directly impact transaction fees. These costs are only to make the contracts available, and they are only applied one time or every time we deploy the contracts.

6.2 Event listener

In the booking management smart contract, we have implemented the grantAccess method. This method is responsible for emitting the AccessGranted event, but only after verifying two key criteria: first, that the booking in question is active, and second, that the current time falls within the specified booking period. The code employs two required statements if active and within the designated time period as defined in 5.2.3. If both criteria are satisfied, the AccessGranted event is triggered, emitting the necessary information such as the booking ID, user address, and property ID.

The method ensures access is granted to the correct user at the right time. When an event is emitted, it is verifiable since it is written to the tamperproof ledger. From Figure 5.1, we can see that the user is triggering the grantAccess method and how the Proxy listens to events and requests the home assistant to unlock the door. In our recent series of tests designed to assess the costs associated with Ethereum operations, we evaluated the expenses incurred from granting access to a property via the blockchain. From our latest experiment conducted on December 6, 2023, we observed a transaction cost amounting to 0.00005185471484204 ETH.

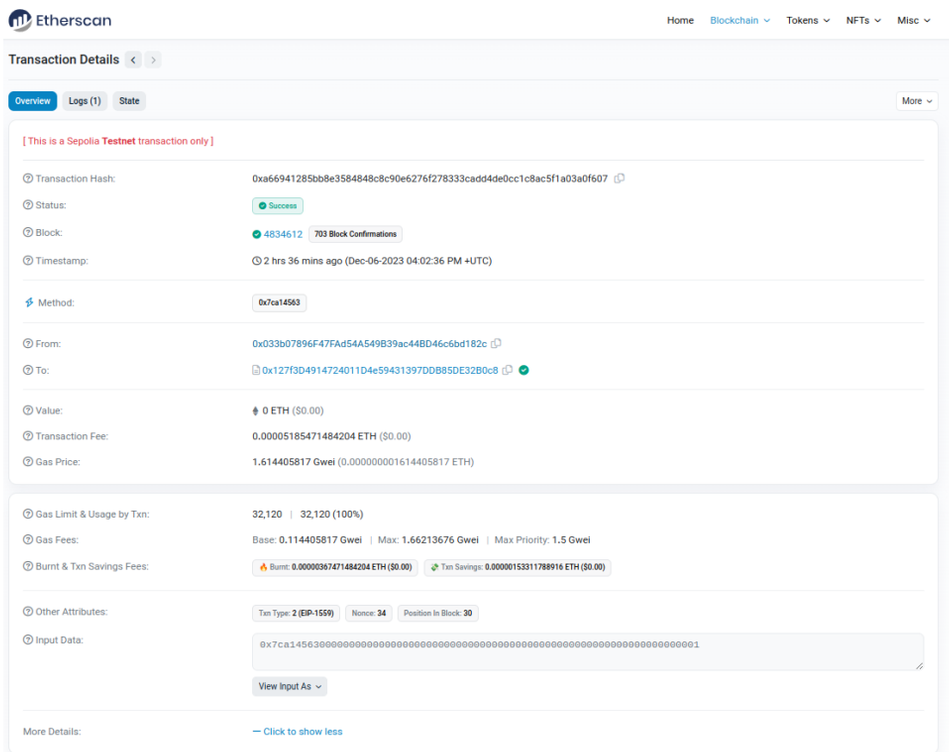

Figure 6.2: Screenshot of the transaction on Sepolia Testnet

As shown in Figure 6.2, the transaction with hash[3] was successfully included in block number 4834612 on the Sepolia Testnet. This transaction invoked a contract method, identified by the signature `0x7ca14563`, originating from our wallet hash at address[4] and targeting the contract at hash[5].

Notably, the transaction transferred zero Ether as part of the contract interaction. The incurred cost for executing the transaction is accurately measured at: `0.00005185471484204` ETH, calculated based on the fully utilised gas limit of 32,120 and a gas price of 1.614405817 Gwei. Adhering to the EIP-1559 standard [46], the transaction fees were composed of a base fee of 0.114405817 Gwei, a maximum fee of 1.66213676 Gwei, and a priority fee of 1.5 Gwei. The nonce for the transaction was 34, indicating its sequence in our wallet transactions, and it was the 30th transaction in the block, reflecting its order of processing.

3. hash: 0xa66941285bb8e3584848c8c90e6276f278333cadd4de0cc1c8ac5f1a03a0f607
4. Wallet hash address: 0x033b07896F47FAd54A549B39ac44BD46c6bd182c
5. Contract hash: 0x127f3D4914724011D4e59431397DDB85DE32B0c8

The input data, starting with `0x7ca14563...00000001`, represents the encoded parameters for the smart contract function call. On December 6, 2023, the Ethereum to US Dollar exchange rate was recorded at 2,269.27 USD per ETH [47], providing a basis for calculating the fiat monetary cost of blockchain operations conducted in Ethereum.

The specific transaction in question incurred a cost of 0.00005185471484204 ETH. Exchange rate, the cost in US Dollars is computed as follows:

$$\text{Transaction Cost (ETH)} \times \text{Exchange Rate (USD/ETH)} = \text{Transaction Cost (USD)} \tag{6.4}$$

$$0.00005185471484204 \times 2,269.27 = 0.12 \text{ USD} \tag{6.5}$$

Therefore, the transaction executed to unlock the door on December 6 2023, amounted to an estimated USD 0.12.

Subsequently, applying the same conversion rate, we observed that the prior transaction for creating the booking, accessible via this link, incurred a cost of USD 0.51. This expense is a prerequisite for granting access to the property.

6.3 Issues

We encountered several obstacles when connecting the Proxy to the local Ganache service. The library web3 we used in the Proxy implementation has connectivity incompatibilities regarding event subscribers towards local instances like Ganache. We observed a virtualisation issue despite both components being configured to listen on the same port and network. The Proxy consistently failed to detect the AccessGranted event emitted by the smart contract. This problem has proven to be quite tedious and remains unresolved. We suspect this issue might stem from inherent limitations in Ganache's event emission capabilities, which could restrict its ability to dispatch certain types of contract events reliably. The issue was not resolved but bypassed by deploying and debugging the contract on Remix, where the Proxy immediately detected the event. Subsequently, this resolution was also confirmed on the Sepolia testnet.

6.3.1 Dilemma of Block Time in Booking Transactions

Another significant challenge was managing time-related aspects within the EVM. While the booking system is naturally expressed in standardised, real-world time, the EVM operates solely on the concept of block time 2.3. This

exposed a unique challenge: effectively translating or synchronising block time with real-world time standards. Since the EVM lacks a direct concept of now. time, as is commonly used in conventional programming environments, finding a method to convert or relate block time to regular time has been a hurdle that impacts accuracy and functionality.

/7

Discussion

When considering the use of smart contracts for controlling access, a fundamental question arises: Is it necessary to record every lock and unlock action on a tamperproof ledger? Initially, it may seem excessive. However, this capability ensures verifiable actions by tenants or service personnel, which can be crucial in certain scenarios. Further exploration into specific use cases where this verification level is indispensable would be beneficial, as opposed to scenarios where simpler solutions could be adequately effective.

Another important reflection during this implementation is the potential of smart contracts in creating a unified interface for rental agreements. Utilising smart contracts could streamline property registration and management, offering transparency and ease of use. For instance, property owners could register their properties on Aquilier, which becomes a central, transparent interface accessible by various listing services such as Airbnb and Booking.com. This integration could significantly simplify the property listing process, addressing issues like double booking and inconsistent pricing. Utilising Aquilier and the inherent transparency from a blockchain system would also support regulations and governments to tax rental agreements directly. Such utilisation's technical and operational feasibility, including challenges and potential barriers, calls for a detailed discussion.

This approach could revolutionise property management on digital platforms, but it is essential to critically examine how blockchain technology can specifically address and resolve these issues beyond the conceptual advantages. The

discussion should, therefore, also focus on the technical specifics, operational challenges, and real-world implications of integrating blockchain with existing property listing platforms.

7.1 Advantages and Challenges

As we showed in our analysis and presented during the introduction to this thises 1 The inherent transparency of blockchain ensures openly recorded and traceable transactions, aiding tax collection and enforcement. Smart contracts on blockchain platforms can automatically enforce regulatory requirements. The immutable ledger provides a reliable, tamperproof record that is indispensable for audits and compliance. These are all features that potentially could answer the open question from the EU research publication [1].

Integrating blockchain with existing systems presents compatibility challenges. Scalability is crucial for handling large volumes of transactions. Balancing transparency with data privacy is a key concern. Constant changes in the regulatory environment and local variance pose barriers to adoption. Gaining public trust and acceptance is also critical.

7.2 Security and Privacy Concerns

Addressing the privacy and security risks associated with the emission of events by smart contracts in Aquilier, several issues have been identified, some implemented, and some left for future work. The anonymisation of data is crucial. By ensuring that the data emitted by these events is anonymised, we effectively prevent the possibility of identifying specific houses or locations, thereby safeguarding the privacy of users and properties.

Restricting who can subscribe to or access the event logs is essential. In this context, the use of a private or permissioned blockchain emerges as a suitable solution, offering controlled access while maintaining the benefits of blockchain technology.

Another vital strategy is the introduction of a layer of abstraction. This approach ensures that lock/unlock events are processed and stored securely before being recorded on the blockchain. This layer acts as a buffer, enhancing the security and integrity of the data.

The encryption of data within these events is critical. Encrypting this informa-

tion ensures that only authorised individuals or systems can interpret the event data. This step is crucial in enhancing the system's overall security, ensuring that sensitive information remains confidential and accessible only to those with the necessary permissions.

Together, these strategies form a comprehensive approach to mitigating the inherent risks in blockchain-based event logging, ensuring that Aquilier operates with the highest standards of privacy and security.

7.2.1 Potential Risks

When a smart contract associated with a home locking system emits events each time a door is locked or unlocked, these events are recorded on the blockchain. In a public blockchain scenario, this information becomes accessible to anyone who can query the blockchain. This raises significant privacy and security concerns, as malicious entities could potentially track these patterns to infer when a house is most likely unoccupied.

To illustrate this, consider the following code snippet from our proxy server used to subscribe and listen for events from a smart contract:

```
func EventSubscriber(contractABI abi.ABI, WsUrl string,
    ContractAddrHex string) error {
  client, err := ethclient.Dial(WsUrl)
  if err != nil {
    return fmt.Errorf("failed to connect to the WebSocket
    Ethereum client: %v", err)
  }
  contractAddress := common.HexToAddress(SepoliaContract)
  query := createFilterQuery(contractAddress, client)
  defer client.Close()
  subscribeToEvents(client, query, contractABI)
  return nil
}
```

Listing 7.1: Event subscription setup function

This function allows a client to connect to an Ethereum node via Infura and subscribe to events emitted by a specified smart contract. It demonstrates how, with basic knowledge of Ethereum and the appropriate tools, anyone can listen to events emitted by a public smart contract. Such ease of access underlines the potential risk: if the events contain identifiable information about lock/unlock actions, it could be exploited to track when a property is vacant. This scenario underscores the need for careful consideration of privacy and security measures in the design of blockchain-based home automation systems and the need for future work and elaboration on the security features for Aquilier. It also

highlights the imperative for continued research and development, particularly in enhancing the security features of the Aquilier system. This is crucial for ensuring that such systems provide convenience and efficiency and maintain the highest security and user privacy standards.

Blockchain's Unique Value Proposition
While the immediate utility of blockchain for routine actions like door locking may not be apparent, its role in enhancing security and trust is undeniable. Blockchain's immutable ledger provides a definitive record of actions, crucial in scenarios where proof of access is required. This technology could be particularly valuable in high-security environments or where transparency in access control is a priority.

7.3 Cost Analysis of Smart Contract Operations

The feasibility of using blockchain for every lock and unlock action hinges on a careful analysis of cost versus benefit. Given the transaction costs associated with blockchain, particularly on platforms like Ethereum, evaluating whether the enhanced security justifies the expense is essential. We will delve into this topic during this section. We unlocked the door on the 6th of December 2023 at 0.12 USD as we showed in chapter 6.2.

The booking management smart contract incorporates a method called grantAccess, which is the one that calls the smart lock and unlocks the door. This method adheres to strict checks before allowing property access; it verifies the booking status and ensures the access request aligns with the booking timeframe. These verifications are crucial for maintaining the integrity and reliability of the system. It is only three lines of optimises. We must consider that we pay pr executed code. The smart contract enforces these rules via the `require` statements, which serve as guardrails to prevent unauthorised access 5.2.3. Only upon successful validation does the `AccessGranted` event get emitted. That the proxy then picks up on and calls the `unlockVirtualLock`. The immutability of the blockchain ledger where this event is recorded ensures that the access grant is tamperproof and auditable.

The associated transaction cost is a critical aspect of deploying smart contracts on the Ethereum blockchain. Our observed analysis on the 6th of December, 2023, revealed a transaction cost of 0.00005185471484204 ETH for invoking the `grantAccess` method.

$$\text{Cost (USD)} = \text{Cost (ETH)} \times \text{Exchange Rate (USD/ETH)} \tag{7.1}$$

$$\text{Cost (USD)} = 0.00005185471484204 \times 2269.27 \approx 0.12 \tag{7.2}$$

7.3.1 Economic Feasibility

Given the transaction cost of approximately 0.12 USD to grant access and considering the previous decentralised transaction cost of 0.51 USD, we can assess the economic feasibility of smart contract operations for property access management. The total cost of 0.63 USD for a complete booking and access cycle is relatively minimal, suggesting that implementing such blockchain-based systems could be economically viable for commercial applications. It is important to note that the cost efficiency stems from the low operational expenses of running smart contracts and the EIP-1559 transaction pricing mechanism, which optimises fee structures [46].

In this example, we have only unlocked the door once the grant access cost of 0.12 USD would have to be applied each time the door needs to be locked or unlocked. The cost of creating the booking was 0.51; in this case, the booking was for the entire December 2023. We have previously discussed the feasibility of doing off-chain transactions. For example, all door access can be off-chain and only written once daily or once per booking. This would lower the transaction considerably.

7.3.2 Practical Implications

The findings indicate promising potential for integrating blockchain technology in real-world applications such as property management systems. The ability to execute secure transactions with verifiable access control at a low cost provides a strong case for adopting such systems. It is imperative to continuously monitor the fluctuating transaction fees on the Ethereum network to maintain the solution's cost-effectiveness.

Network Congestion and Transaction Cost

Gas prices can escalate rapidly during periods of high demand on the Ethereum network, such as during a significant market event or when any popular decentralised finance application is processing numerous transactions. An exemplary case of this phenomenon was observed on the 1st of May, 2022. That day, the network experienced considerable congestion, propelling the gas prices to an unusual high.

Given the gas price of 78.999 Gwei, as recorded on the date above, and the standard gas amount of 32,120 required for executing the `grantAccess` method, we can approximate the transaction cost in Ethereum and subsequently convert it to USD.

$$\text{Transaction Cost (ETH)} = \text{Gas Used} \times \text{Gas Price (ETH)} \tag{7.3}$$

$$32,120 \times 78.999 \times 10^{-9} = 0.00253744788\,\text{ETH} \tag{7.4}$$

The calculation above shows the transaction cost in ETH. To convert this cost to USD, we use the exchange rate at the time of the transaction. On the 1st of May, 2022, the exchange rate was approximately 2730 USD per ETH.

$$\text{Transaction Cost (USD)} = \text{Transaction Cost (ETH)} \times \text{Exchange Rate (USD/ETH)} \tag{7.5}$$

$$\text{Transaction Cost (USD)} \approx 0.00253744788 \times 2730 = 6.92723271\,\text{USD} \tag{7.6}$$

With the given gas price and exchange rate, the cost to execute the `grantAccess` transaction on the 1st of May, 2022, would have been approximately 6.92 USD. This theoretical exercise highlights the impact of network congestion influencing transaction costs on the Ethereum blockchain and underscores the need for strategic transaction timing to mitigate expenses.

Blockchain technology's inherent transparency and robust security features significantly contribute to building trust in digital access control systems. Trust comes with associated costs of the variable nature of market-driven expenses, such as transaction fees in blockchain networks. As we decentralise and anonymise the workload for everyday transactions, these costs will inevitably vary according to market fluctuations. Despite this, the trade-off may lead to a more secure and uniform solution, offering enhanced interoperability across different platforms and systems. This potent, decentralised, interoperable, and secure framework makes blockchain a compelling option for future digital access control solutions. In conclusion, the successful deployment and operation of the booking management smart contract on the Sepolia Testnet demonstrate the practicality and affordability of blockchain-based access control systems. The cost analysis underscores the potential for widespread adoption, especially as blockchain technology evolves and matures.

Alternatives and Hybrid Approaches

Ensuring security and trust should be considered, especially for routine actions. A hybrid approach could be more practical, where blockchain is reserved for critical transactions, and conventional methods are used for everyday actions. This approach could balance the benefits of blockchain with operational efficiency. Additionally, exploring advanced blockchain solutions like Layer 2 protocols or alternative, less costly blockchains might offer a more viable.

7.4 Automated Economic Security in Booking Management

Automated Economic Security in Booking Management via Ethereum In decentralised applications, Ethereum-based smart contracts offer innovative solutions for booking management and tenant transactions. Utilising Ethereum for these purposes introduces a system where tenants can be charged automatically, ensuring seamless and secure financial transactions. In Babel et al. [48] research titled "Clockwork Finance: Automated Analysis of Economic Security in Smart Contracts", he underscores the necessity of automated tools to analyse potential economic vulnerabilities, particularly in contracts handling tenant charges and booking logistics.

7.5 Etherum node vs service

The use of Infura as a designated service is a strategic choice, providing stable and continuous access to the Ethereum network. This eliminates the complexity of maintaining our blockchain nodes, ensuring the reliability and scalability of our DApp. Incorporating smart lock technology is the final piece in our system, enabling the physical enactment of digital permissions granted by our smart contracts. This convergence of digital authorisation and physical access minimises the innovative essence of our DApp, positioning it as a comprehensive solution for modernising and securing property rentals.

7.6 Future Work

As Aquilier continues to evolve, several avenues for future development and research have emerged, each offering potential enhancements and utilising to the current system.

Privacy and Anonymising data

A multifaceted approach is essential in addressing the privacy and security challenges posed by blockchain-based home automation systems. A key strategy involves anonymising or blurring event data within the blockchain. By masking sensitive details, we can prevent the direct association of lock/unlock events with specific properties or times, thereby enhancing privacy. Simultaneously, exploring permission-based blockchains or off-chain transactions is a viable solution for sensitive event logging.

These specialised blockchain environments restrict access to authorised participants, significantly reducing the risk of public data exposure.

Additionally, integrating advanced encryption protocols plays a crucial role in securing data transactions within the blockchain. This encryption ensures that any data emitted in blockchain events remains secure and indecipherable to unauthorised parties. Furthermore, refining the design and logic of smart contracts is crucial. By modifying these contracts to log events and incorporate privacy-preserving elements selectively, we can significantly minimise the emission of sensitive information.

Collectively, these strategies form a comprehensive approach to mitigate the inherent privacy and security risks in blockchain-based home automation systems, ensuring that the system remains functional and secure and adheres to the highest standards of user privacy and data protection.

Advanced User Authentication Methods

Implementing more sophisticated authentication mechanisms, like biometric verification or multi-factor authentication, could further strengthen the security of property access within Aquilier. Upgrade the design to aquilier, including its own node or multiple chain nodes running on the proxy server. This would allow users to, for example, scan a QR code and authenticate themselves at the door or via the phone.

Cost Efficiency

Implement off-chain transactions that can enable the user to choose the frequency of committing transactions to the ledger, for example, every day or month, alternatively on request. Continuous efforts to optimise the scalability and performance of Aquilier will be crucial, especially as the user base grows and transaction volumes increase.

Integration features

Offer a designated API that can easily provide the same service as Aquilier to other platforms.

Interoperability alternative Smart Home Systems

Expanding Aquilier's compatibility with a broader range of smart home systems could greatly increase its market adoption and user convenience.

Decentralised Governance Models

Investigating decentralised governance mechanisms for platform management could offer a more democratic and transparent way of handling updates and changes to the system.

Legal and Regulatory compliance

Further research into the legal and regulatory aspects of using blockchain in the rental market will be vital, particularly as laws and norms around digital transactions and property rentals evolve.

Environmental Impact Analysis

Assessing and minimising the environmental impact of blockchain technology, especially in terms of energy consumption, aligns with growing concerns over sustainable digital solutions. The system itself might contribute to environmental impact, for example, in areas where restrictions are applied or where the more efficient way of utilising homes might decrease the environmental impact due to lower demand for designated short-term rental units.

Each of these areas presents an opportunity to enhance Aquilier's functionality, security, and market relevance, ensuring that the application remains at the forefront of technological innovation in the rental market.

8

Conclusion

In this thesis, we stated that Utilising Ethereum smart contracts to transfer property access can automate property management and enhance security and operational efficiency. We confirm through the development of Aquilier that utilising Ethereum smart contracts to transfer property access is feasible. The adaptation of smart contracts brings inherited features such as security and transparency. Consequently, this approach further automates and integrates with smart home systems as we prove that blockchain technology is technically feasible and economically viable.

Given the requirements in chapter 4, Our devices system Aquilier fulfilled the technical requirement 1. A user should be able to request access to a property they have booked. We showed how this feature is implemented in 5.2.3 and evaluated its execution in 6.2. As a result of the initial requirements, Aquilier also fulfilled all other functional requirements. Our evaluation shows that using blockchain to unlock the door would cost, on average, approximately 0.12 USD. Suppose every access to the ledger is committed immediately, considering the worst-case scenario. In that case, one can afford to open the door 83 times a month or 2,6 times a day before reaching the stated threshold of 10 USD a month in our requirements 9. This cost can be further improved by implementing off-chain transactions as discussed in future work 7.6 and only committing daily or monthly bulk transactions to the Ethereum network, significantly reducing the cost.

By implementing authentication via MetaMask, we securely fulfilled requirements 3 and 6. We also assure privacy by authenticating users. At the same time, we exposed an important issue regarding data privacy. If each access is written to the public ledger, anyone can listen to the event on the blockchain and see when one is home. We discussed this in chapter 7.2.1 and proposed significant improvement in our section future work 7.6.

On days when no access has been requested, write operations would not be necessary, further reducing the cost. Additionally, with an off-chain solution, the user could control the cost themself if the frequency of writing transactions could be configured to the user's preference, for example, once per tenant in a short-term rental agreement or when the user requests to write the transactions. One of the inherent advantages of using smart contracts in property management is their enhanced security and transparency. The immutable and cryptographic security features of blockchain are an important aspect that brings added value, something we listed as a requirement in the chapter 6. We also see the transparency that comes with a public ledger system ensures that all transactions are traceable and auditable, fostering a trustworthy environment for all parties involved in the rental market.

While blockchain offers significant advantages in terms of security and trust, Its application in routine actions such as door access control must be carefully considered, weighing the costs, benefits, and impacts on user experience. Future advancements in blockchain technology could potentially make the application more practical and widespread in smart home systems. While the Aquilier design method is feasible, it is imperative to investigate alternatives and hybrid solutions where off-chain transactions might be an option.

9

Appendix

The code of our devised system Aquilier can be found on GitHub repo: https://github.com/niklasstrand/Aquilier

Bibliography

[1] European Parliament. *Blockchain and the General Data Protection Regulation: Can distributed ledgers be squared with European data protection law?* Accessed: 2023-11-18. 2023. URL: https://www.europarl.europa.eu/RegData/etudes/BRIE/2023/739334/EPRS_BRI(2023)739334_EN.pdf.

[2] Municipal Research and Services Center. *Affordable Housing and the Impact of Short-Term Rentals*. Accessed 2023-11-18. 2021. URL: https://mrsc.org/stay-informed/mrsc-insight/december-2021/affordable-housing-and-the-impact-of-short-term-re.

[3] Federal Trade Commission. *Rental Listing Scams*. Accessed: 2023-12-01. n.d.

[4] Primavera De Filippi, Morshed Mannan, and Wessel Reijers. "Blockchain as a confidence machine: The problem of trust challenges of governance." In: *Technology in Society* 62 (2020), p. 101284. ISSN: 0160-791X. DOI: https://doi.org/10.1016/j.techsoc.2020.101284.

[5] Nguyen. Truong et al. "A blockchain-based trust system for decentralised applications: When trustless needs trust." In: *Future Generation Computer Systems* 124 (2021), pp. 68–79. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2021.05.025.

[6] Anne Håkansson. *Portal of Research Methods and Methodologies for Research Projects and Degree Projects*. Tech. rep. Kista, Sweden: Department of Software and Computer Systems, The Royal Institute of Technology KTH, 2013.

[7] Nayan B. Ruparelia. "Software Development Lifecycle Models." In: *SIGSOFT Softw. Eng. Notes* 35.3 (May 2010), pp. 8–13. ISSN: 0163-5948. DOI: 10.1145/1764810.1764814.

[8] Infura. *Documentation*. https://docs.infura.io/. Accessed: 2023-11-28. 2023.

[9] Aakash Sharma et al. "Capturing Nutrition Data for Sports: Challenges and Ethical Issues." In: *29th International Conference on MultiMedia Modeling*. 2023.

[10] Håvard D. Johansen et al. "Scalable Infrastructure for Efficient Real-Time Sports Analytics." In: *Companion Publication of the 2020 International Conference on Multimodal Interaction*. ICMI '20 Companion. Vir-

tual Event, Netherlands: Association for Computing Machinery, 2020, pp. 230–234. ISBN: 9781450380027. DOI: 10.1145/3395035.3425300.

[11] Tor-Arne S. Nordmo et al. “Dutkat: A Multimedia System for Catching Illegal Catchers in a Privacy-Preserving Manner.” In: *Proceedings of the 2021 Workshop on Intelligent Cross-Data Analysis and Retrieval*. ICDAR ’21. Taipei, Taiwan: Association for Computing Machinery, 2021, pp. 57–61. ISBN: 9781450385299. DOI: 10.1145/3463944.3469102.

[12] Nick Szabo. *Smart Contracts*. Accessed: 2023-10-31. 1994. URL: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html.

[13] Nick Szabo. “Formalizing and Securing Relationships on Public Networks.” In: *First Monday* 2.9 (1997).

[14] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. Accessed: 2023-10-30. 2014. URL: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.

[15] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: 2023-10-30. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[16] Everett Hildenbrandt et al. “KEVM: A Complete Formal Semantics of the Ethereum Virtual Machine.” In: *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. 2018, pp. 204–217. DOI: 10.1109/CSF.2018.00022.

[17] Loi Luu et al. “Making Smart Contracts Smarter.” In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 254–269.

[18] Ying Fu et al. “EVMFuzzer: Detect EVM Vulnerabilities via Fuzz Testing.” In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. Tallinn, Estonia: Association for Computing Machinery, 2019, pp. 1110–1114. ISBN: 9781450355728. DOI: 10.1145/3338906.3341175.

[19] Ethereum. *Gas and fees*. Accessed: 2023-09-21. 2023. URL: https://ethereum.org/en/developers/docs/gas/.

[20] Renlord Yang. “Empirically Analyzing Ethereum’s Gas Mechanism.” In: *arXiv preprint arXiv:1905.00553* (2019).

[21] Ethereum.org. *Blocks*. Accessed: 2023-11-30. 2023. URL: https://ethereum.org/en/developers/docs/blocks/ (visited on 11/30/2023).

[22] Clemens Brunner et al. “DID and VC:Untangling Decentralized Identifiers and Verifiable Credentials for the Web of Trust.” In: ICBTA ’20. Xi’an, China: Association for Computing Machinery, 2021, pp. 61–66. ISBN: 9781450388962. DOI: 10.1145/3446983.3446992.

[23] *Airbnb*. https://www.airbnb.com. Accessed: 2023-09-21.

[24] *Booking.com*. https://www.booking.com. Accessed: 2023-09-21.

[25] *Rentalia*. https://rentalia.com/. Accessed: 2023-09-21.
[26] *Home2Book*. https://www.home2book.com/. Accessed: 2023-09-21.
[27] *VRBO*. https://www.vrbo.com. Accessed: 2023-09-21.
[28] *Expedia*. https://www.expedia.com. Accessed: 2023-09-21.
[29] *HomeAway*. https://www.homeaway.com. Accessed: 2023-09-21.
[30] *TripAdvisor Rentals*. https://www.tripadvisor.com/Rentals. Accessed: 2023-09-21.
[31] *FlipKey*. https://www.flipkey.com. Accessed: 2023-09-21.
[32] *Agoda Homes*. https://www.agoda.com/homes. Accessed: 2023-09-21.
[33] *TUI Villas*. https://www.tuivillas.com. Accessed: 2023-09-21.
[34] *Homestay.com*. https://www.homestay.com. Accessed: 2023-09-21.
[35] *Zillow Rentals*. https://www.zillow.com/rent/. Accessed: 2023-09-21.
[36] *Trivago*. https://www.trivago.com. Accessed: 2023-09-21.
[37] *HomeToGo*. https://www.hometogo.com. Accessed: 2023-09-21.
[38] Airbnb. *Airbnb 2023 Winter Release: Smart Lock Integration*. Accessed: 2023-11-08. 2023. URL: https://news.airbnb.com/product-releases/airbnb-2023-winter-release/.
[39] Booking.com. *Smart home technology for your short-term rental*. Accessed: 2023-11-15. URL: https://partner.booking.com/en-gb/help/legal-security/security/protecting-your-home-property-security-devices.
[40] Home Assistant. *Common Tasks for Container*. https://www.home-assistant.io/common-tasks/container/. Accessed: 2023-10-26.
[41] Home Assistant. *Home Assistant Core: Open Source Repository*. https://github.com/home-assistant/core. Accessed: 2023-10-26. 2023.
[42] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. "A Survey of Attacks on Ethereum Smart Contracts SoK." In: *Proceedings of the 6th International Conference on Principles of Security and Trust - Volume 10204*. Berlin, Heidelberg: Springer-Verlag, 2017, pp. 164–186. ISBN: 9783662544549. DOI: 10.1007/978-3-662-54455-6_8.
[43] Leyla Moctar M'Baba et al. "Extracting Artifact-Centric Event Logs From Blockchain Applications." In: *2022 IEEE International Conference on Services Computing (SCC)*. 2022, pp. 274–283. DOI: 10.1109/SCC55611.2022.00048.
[44] ConsenSys Software Inc. *MetaMask - A crypto wallet & gateway to blockchain apps*. Accessed: 2023-09-21. 2023. URL: https://metamask.io/.
[45] *Sepolia Faucet*. Accessed: 2023-09-21. URL: https://sepolia-faucet.pk910.de/#/.
[46] Yulin Liu et al. "Empirical Analysis of EIP-1559: Transaction Fees, Waiting Times, and Consensus Security." In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. ACM, Nov. 2022. DOI: 10.1145/3548606.3559341.
[47] *Ethereum Price*. Accessed: 2023-12-06. Crypto.com. URL: https://crypto.com/price/ethereum.

[48] Kushal. Babel et al. “Clockwork Finance: Automated Analysis of Economic Security in Smart Contracts.” In: *2023 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2023, pp. 2499–2516. DOI: 10.1109/SP46215.2023.10179346.