POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Driving Innovation: Exploring the Potential of a Bitcoin Car-Wallet for Autonomous Vehicles

Supervisor:

Prof. Danilo Bazzanella

Candidate:

Giorgio Rasetto

Company supervisor:

Enrico D'Acquisto

Academic Year 2022/2023
Torino

To all those who live to be free

List of Figures

Acronyms

AI
Artificial Intelligence

M2M
Machine-to-Machine

PoW
Proof of Work

SHA-256
Secure Hash Algorithm 256-bit

EVM
Ethereum Virtual Machine

ICO
Initial Coin Offering

DAO
Decentralized Autonomous Organization

DApp
Decentralized Application

DeFi
Decentralized Finanance

HTLC

Hashed Time-Lock Contract

ACH

Automated Clearing House

DTCC

Depository Trust & Clearing Corporation

EVM

Ethereum Virtual Machine

TAP

Taproot Asset Protocol

UTXO

Unspent Transaction Output

NFT

Non-Fungible Token

SAE

Society of Automotive Engineers

CPU

Central Processing Unit

GPU

Graphics Processing Unit

V2V

Vehicle-to-Vehicle

V2I

Vehicle-to-Infrastructure

CNN

Convolutional Neural Network

RNN

Recurrent Neural Network

GAN

Generative Adversarial Network

NFC

Near Field Communication

QR

Quick Response

BTC

Bitcoin

BMW

Bayerische Motoren Werke AG (Bavarian Motor Works)

W3C

World Wide Web Consortium

DID

Decentralized Identifier

BIP

Bitcoin Improvement Proposal

RGB

Riemannian Geometry of Bitcoin's UTXO

PKI

Public Key Infrastructure

LSP

Lightning Service Provider

SDK

Software Development Kit

CA

Certification Authority

HTTPS

HyperText Transfer Protocol Secure

TLS

Transport Layer Security

BDK

Bitcoin Development Kit

LNPBP

Lightning Network Proposal and BOLT Process

API

Application Programming Interface

URL

Uniform Resource Locator

HTTP

Hypertext Transfer Protocol

IP

Internet Protocol

TCP

Transmission Control Protocol

UDP

User Datagram Protocol

GUI

Graphical User Interface

JSON

JavaScript Object Notation

UI

User Interface

RNG

Random Number Generator

RBF

Replace-By-Fee

UX

User Experience

Abstract

In recent years, the automotive market has undergone significant changes. Cars have evolved from simple means of transportation into actual platforms for services; from car sharing to pay-as-you-go services consumable directly through the car's user interface, and even autonomous driving. The latter, in particular, will dramatically alter our concept of mobility. Individuals will no longer require car ownership, and cars themselves will no longer rely on an owner to fulfill their needs: they will self-fuel, make payments for parking and tolls directly through machine-to-machine transactions, and when they will have some problems they will autonomously visit a mechanic for repairs. In essence, the car will become a fully self-sufficient economic agent, or even a source of income for its owner.

The scenario is awsome, yet like every great tale, it has its dark side. Envision a world where autonomous vehicles are the mobility standard, and an authoritarian state aims to hinder political dissidents or inconvenient individuals from moving within the country. Here, a revolutionary technology like autonomous driving could transform into a tool for coercion and mass surveillance.

Thus, it is imperative that privacy be upheld as an essential right. How can we achieve privacy in a wholly digital environment?
Fortunately, since 2009, we have a tool called Bitcoin, which, enabled by cryptography, empowers us with privacy in digital payments and beyond.
The concept I am advancing in this thesis is to embed within autonomous vehicles a "Car-Wallet" system—an entity that enables the autonomous agent to emancipate itself from third-party services concerning payments, micro-payments, and the notarization of interactions with other market agents. This way, the agent becomes entirely independent from intermediaries, ensuring security and privacy.

Aknowledgments

I would like to express my heartfelt gratitude to my supervisor, Danilo Bazzanella, for his unwavering support throughout the thesis and for his guidance within the student team BitPolito. Acknowledgements also go to Enrico and Debora from Turin Tech S.r.l., who provided me with the tools and support necessary for the development of the project.

I am also deeply appreciative of Gabo, Salvo, Push, Michi, and all the wonderful individuals who accompanied me on this journey, providing encouragement and camaraderie.

A special thank you goes out to my life partner, Alessia, for her constant presence and unwavering support through every moment of this endeavor. And, of course, my deepest gratitude to my mother, my sister, my father, my entire family and all my “Pucia” friends for their unwavering support.

Last but certainly not least, I must extend a heartfelt thank you to my grandfather Eraldo, the most remarkable person I know, a source of great inspiration, and an indefatigable worker.

Your collective support and encouragement have been invaluable to me, and I am truly grateful for each and every one of you who have made this journey possible.

Table of Contents

Chapter 1

Introduction

Cars are undergoing a transformation, becoming increasingly digital and integrating an operating system that include a dedicated app store. This signifies that the range of services accessible to users will expand significantly.

These services will take on an even more intriguing role in autonomous driving vehicles. Cars will evolve into independent economic agents. They will refuel themselves, make payments for parking, tolls, and more... they will even proceed to cleaning and self-adjustment in case of issues.

This topic isn't limited to just private cars; it extends to goods and passenger transport vehicles as well. In these cases, the array of accessible services will be even more diverse.

At some point, these vehicles will be entirely self-sufficient. For this reason, they will require a "Digital Identity," allowing them to identify within a network of independent agents and engage in "contracts" with other entities on the network. I'm referring, for example, to ownership contracts (who owns the car? From whom was it purchased?), insurance contracts, and maintenance and performance monitoring contracts. There's a need for an infrastructure that enables machine-to-machine or

human-to-machine contract agreements.

Because these vehicles act as economic agents, they need to be capable of conducting financial transactions with other economic agents. However, current payment methods are inadequate due to: 1. Geographical limitations: different countries, different networks. 2. High costs: major payment networks like Visa and Mastercard charge at least 2 to 3% per transaction. 3. Privacy concerns: current payment methods are susceptible to censorship. In a global economy with entirely independent artificial economic agents, total censorship isn't feasible; otherwise, mass surveillance is just around the corner.

An independent economic agent requires intelligence. It must be able to make autonomous decisions, necessitating two things:
Advanced AI algorithms enabling autonomous decision-making.
A form of programmable and censorship-resistant currency, ensuring autonomy and irreversibility of contracts.

The most promising solution to these challenges appears to be Bitcoin, a protocol that guarantees fundamental properties such as censorship resistance, immutability, and security, while notably offering a global and permissionless infrastructure for value exchange and cryptographic contract execution.

1.1 Objectives

This master's thesis is developed in collaboration with Turin Tech S.r.l., a company specializing in engineering consulting in the automotive industry. The main goal is to create a proof of concept for a "Car-Wallet," a software solution designed to facilitate the management of digital payments and cryptographic assets. Specifically, the objectives of this project are as follows:
• Researching the state-of-the-art systems currently in use;
• Analyzing the functional properties of Bitcoin as a trust layer, Lightning Network as a machine-to-machine (M2M) payment layer and RGB as a layer for issuing and manipulating cryptographic contracts.
• Developing a proof of concept for a Car-Wallet capable of managing Lightning Network M2M payments and RGB cryptographic contracts;
• Creating a cryptographic contract for vehicle identity and status traceability;
• Analyzing the results and simulating a use case.

1.2 Outline

The remaining parts of this document are organized as follows:

• Chapter 2 - *Background*: within this chapter, a brief description of main concepts needed as basic knowledge background is provided
• Chapter 3 - *State of the art*: in this chapter, current state-of-the-art of alternative solutions to car OSs and payments provider is given
• Chapter 4 - *Solution design*: this chapter introduces conceptual model and design of the proposed solution
• Chapter 5 - *PoC implementation*: this chapter describes the actual implementation of the Proof of Concept
• Chapter 6 - *Results and validation*: this chapter contains the achieved experimental results and the PoC validation, as a comparison with all requirements presented in previous chapters
• Chapter 7 - *Conclusion*: in the end, this last chapter summarizes the achieved results and draws the conclusions of the thesis

Chapter 2

Background

2.1 Bitcoin

2.1.1 Introduction

2.1.1.1 Context

In the context of online commerce, electronic transactions primarily rely on financial institutions as trusted third parties to process payments. Although this system works adequately for most transactions, it still presents inherent weaknesses due to its trust-based model. Completely irreversible transactions are unattainable, as financial institutions cannot abstain from intervening in disputes. This results in increased mediation costs, limiting the minimum feasible transaction size and precluding the possibility of small-scale transactions. Furthermore, there is a broader cost incurred from losing the ability to make non-reversible payments for non-reversible services. The potential for transaction reversal necessitates a widespread need for trust. Merchants must exercise caution towards their customers, requiring more information than would typically be necessary. Additionally, a certain level of fraud is accepted as inevitable. These costs and uncertainties in payments can be circumvented when physical currency is used for in-person transactions, but there is no mechanism that enables payments over a communication channel without the presence of a trusted party.

2.1.1.2 Objective

The objective of Bitcoin is to propose an electronic payment system based on cryptographic proof rather than trust, allowing two willing parties to transact directly each other without the need for a trusted third party. This system should ensure computationally infeasible transactions to reverse, safeguarding sellers from fraud. Furthermore, custody mechanisms should be implemented to protect buyers.

2.1.1.3 Proposed Solution

Bitcoin presents a solution to the double-spending problem by using a distributed peer-to-peer timestamp server to generate a computational proof of the chronological order of transactions. This system relies on a structure called the "Timechain," which permanently and securely records all transactions made. The Timechain ensures the integrity and immutability of transactions, providing cryptographic proof of their chronological sequence. Additionally, to ensure the security of the system, honest nodes, which are participants in the network, have to collectively control greater computational power than any cooperating group of attacking nodes.

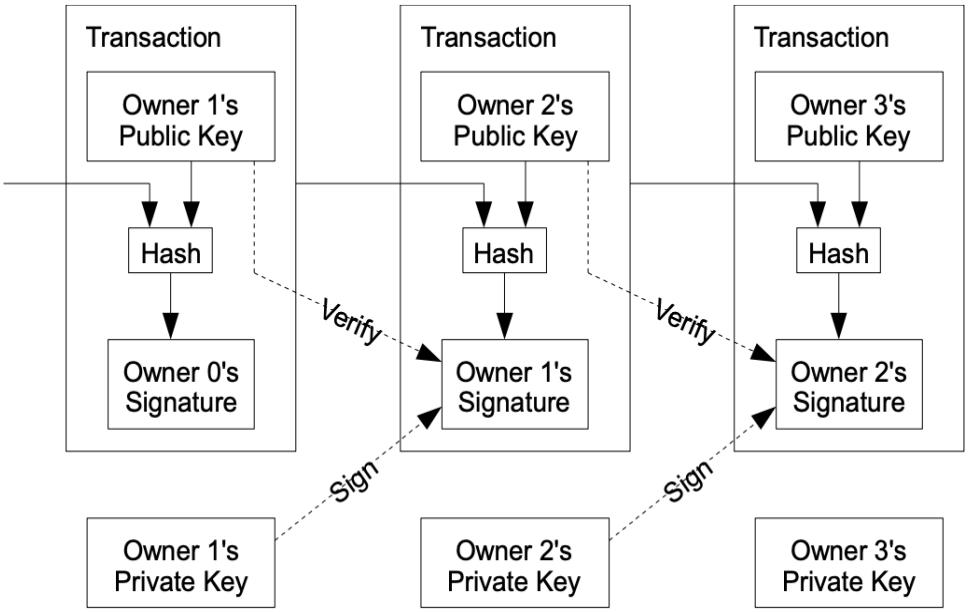
2.1.2 Transactions

Transactions are crucial to understanding the operations of Bitcoin. Transactions constitute the basic unit through which holders of Bitcoin can transfer ownership of these digital coins. This chapter will provide an in-depth explanation of what Bitcoin transactions are, how they are created, and how they ensure the chain of digital ownership.

2.1.2.1 Chain of Digital Ownership

In the context of Bitcoin, a "coin" represents a unit of value that can be transferred among network participants. The chain of digital ownership is the mechanism that keeps track of the ownership of these coins and verifies their authenticity. Each coin is represented as a chain of digital signatures.

Figure 2.1: Transaction chain of ownership (*Source*)



2.1.2.2 Digital Signature and Ownership Transfer

To transfer bitcoins, the current owner must affix their digital signature to a hash of the previous transaction along with the public key of the new owner. This establishes a cryptographic link between the current and previous transactions, ensuring the continuity of the chain of digital ownership.

2.1.2.3 Verification of Ownership Chain

The beneficiary of a transaction can verify the authenticity of the digital ownership chain by performing digital signature verification. The signature of the previous owner must match the public key of the new owner to ensure the

transaction's legitimacy. In case of forgery or alteration of signatures, the transaction would be deemed invalid.

2.1.2.4 Division and Combination of Value

To enable the division and combination of value, Bitcoin transactions can contain multiple inputs and outputs. Typically, a transaction will have a single input coming from a previous transaction with a larger amount, or multiple inputs combining smaller amounts. Similarly, there will be at most two outputs: one for the payment made and one to return any potential change to the sender.

2.1.2.5 Structure of Bitcoin Transactions

A Bitcoin transaction is composed of various fields that specify the input, output, and other necessary information for the execution and verification of the transaction. Common fields include the hash of the previous transaction, the inputs, the outputs, the digital signatures, and other auxiliary information.

2.1.2.6 Security and Integrity of Transactions

Bitcoin transactions are designed to ensure the security and integrity of financial operations on the network. The public key cryptography used to create and verify digital signatures ensures that only the legitimate owner can transfer the coins and that transactions cannot be altered by third parties without being detected.

2.1.3 Timechain

The technology of Bitcoin is based on an innovative data structure called the timechain (or blockchain). This chain of blocks represents the history of Bitcoin transactions and plays a fundamental role in ensuring the immutability and traceability of operations. This chapter aims to explain in detail what the Bitcoin timechain is, how it works, and how it guarantees network integrity.

2.1.3.1 The Timestamp Server

The Bitcoin timechain originates from a timestamp server. This server is responsible for taking the hash of a block of elements to be timestamped and widely broadcasting the obtained hash. Timestamping is performed on a set of transactions, allowing to demonstrate that the data had to exist at the time of the timestamp since they are included in the hash. This operation creates a cryptographic link between the block of transactions and the timestamp itself.

2.1.3.2 The Timestamp Chain

Each timestamp in the Bitcoin timechain includes the previous timestamp in its hash, thus creating a chain of timestamps. This concatenation of timestamps forms the timechain, with each timestamp reinforcing the previous ones. In other words, each block of transactions is linked to the one before it, ensuring the continuity and integrity of the entire chain.

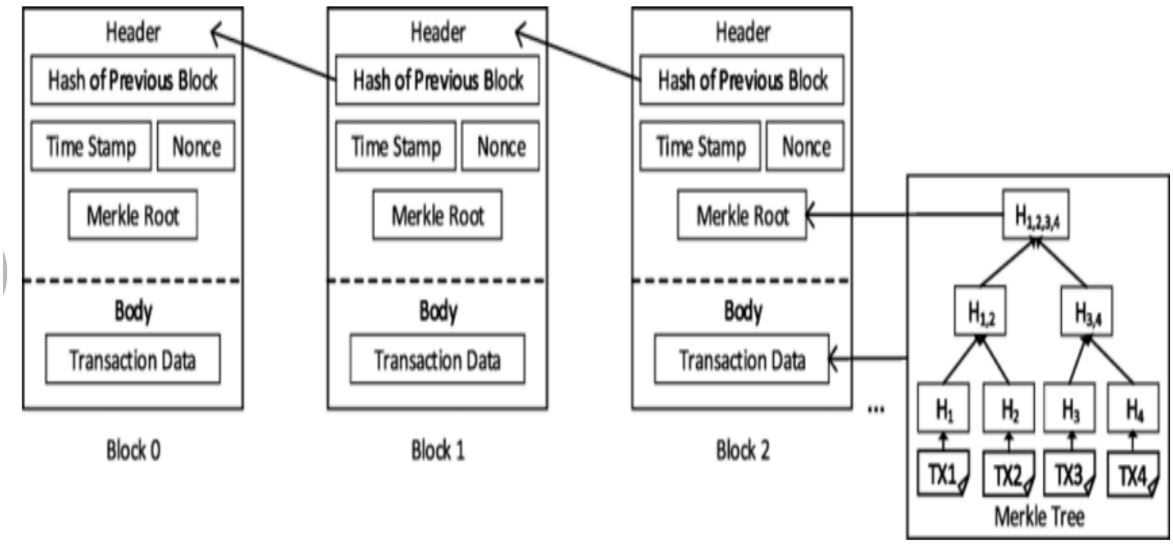


Figure 2.2: Time chain structure

2.1.3.3 Block Structure in the Timechain

A block in the Bitcoin timechain contains a series of crucial information. In addition to the transactions themselves, a block includes a header that contains the hash of the previous block, the timestamp of the current block, the nonce (a

value that is altered to achieve a valid hash), and other necessary metadata for the proper functioning of the network.

2.1.3.4 Consensus and Security in the Timechain

The Bitcoin timechain relies on a decentralized consensus system called proof-of-work (PoW). Miners employ computational power to add new blocks to the timechain and are rewarded with new Bitcoins for their work. This mining process ensures the security of the network since it demands significant computational power to alter the existing chronological chain. This topic will be further elaborated in the next paragraph.

2.1.3.5 Immutability and Traceability of Transactions

The Bitcoin timechain is designed to guarantee the immutability and traceability of transactions. Once a block of transactions is confirmed and included in the timechain, it becomes exceedingly challenging to modify. This implies that transactions are permanent and cannot be altered without the consensus of the majority of the network. This feature provides complete traceability of transactions, enabling anyone to verify the validity of operations.

2.1.4 Proof of Work

The technology of Bitcoin is built upon a distributed consensus system called proof-of-work (PoW). This cryptographic mechanism plays a fundamental role in ensuring the security, integrity and decentralization of the Bitcoin network. This chapter aims to explain in detail what Bitcoin's proof-of-work is, how it works, and how it ensures the reliability and security of the blockchain.

2.1.4.1 Implementation of a Distributed Timestamp Server

To implement a peer-to-peer based distributed timestamp server, it is necessary to use a proof-of-work system similar to Adam Back's Hashcash. This system requires finding a value that, when subjected to a hash function like SHA-256, produces a hash with a certain number of initial zeros.

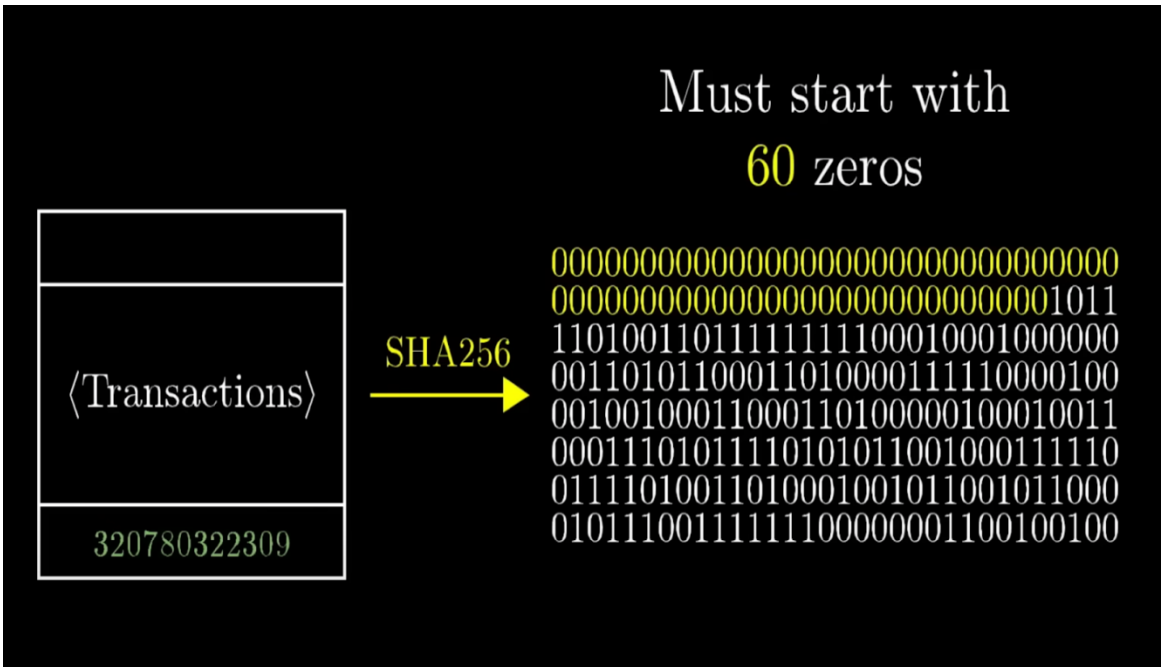


Figure 2.3: Proof of work difficulty target (*Source*)

2.1.4.2 Computational Work and Verification

The proof-of-work requires computational effort to find the suitable value that meets the required criteria. The amount of work required is exponential respect to the number of zeros required in the hash and can be verified by performing a single hash operation.
In the Bitcoin timestamp network, the proof-of-work is implemented by incrementing a value called nonce in the block until a value is found that produces a hash with the desired number of initial zeros. In Figure 2.3, the nonce is the number in green below the transactions, and the hash result is the very large number on the right.

2.1.4.3 Immutability of Blocks

Once the computational effort has been expended to satisfy the proof-of-work, the block cannot be modified without redoing all the work. Since subsequent blocks are linked to the previous one, changing a block would require redoing the work for all subsequent blocks. This feature ensures the immutability of blocks within the Bitcoin blockchain.

2.1.4.4 Majority Decision and Security

The majority decision in the Bitcoin network is represented by the longest chain, which requires the most proof-of-work effort invested. If the majority of the network's computing power is controlled by honest nodes, the honest chain will grow faster and surpass any competing chains. To modify a past block, an attacker would need to redo the proof-of-work for that block and all subsequent blocks, outpacing the work of honest nodes. The likelihood of a slower attacker overtaking the honest chain decreases exponentially as new blocks are added.

2.1.4.5 Difficulty of Proof-of-Work

To offset the increase in hardware speed and variations in interest in maintaining nodes over time, the difficulty of the proof-of-work is determined by a moving average targeting an average number of blocks per hour. If blocks are generated too quickly, the difficulty increases to maintain a consistent generation frequency.

2.1.5 Network

The Bitcoin network is a peer-to-peer system that allows the transmission, confirmation, and validation of transactions within the network. This chapter aims to explain in detail how the Bitcoin network operates and what key steps are involved in transaction management and block confirmation. The concept of the longest chain will also be discussed, along with how forks are managed within the network.

2.1.5.1 Transmission of New Transactions

When a new transaction is created, it is transmitted to all nodes in the Bitcoin network. The network consists of a wide distribution of interconnected nodes, which can be both users and miners. Transmitting new transactions ensures that all nodes are aware of the new activities on the network.

2.1.5.2 Mining and Proof-of-Work

Miner nodes gather valid transactions and group them into a block, then work to find a proof of work (proof-of-work) for their block. The proof-of-work involves searching for a value that, when subjected to a hash function like SHA-

256, produces a result that meets certain requirements, as shown earlier. This process requires computational power and resources.

2.1.5.3 Dissemination and Acceptance of Blocks

When a mining node finds a valid proof of work, it broadcasts the newly mined block to all other nodes in the network. Receiving nodes accept the block only if all transactions within it are valid and have not already been spent. This transaction verification is essential to ensure the integrity of the network and prevent fraudulent activity.

2.1.5.4 Longest Chain and Fork Resolution

Bitcoin nodes always consider the chain with the most work as the correct one and work to extend it. In the case of a fork, where two nodes broadcast different versions of the next block simultaneously, some nodes may receive one version first, and others the second. In this scenario, nodes work on the first received version but keep the other in case it becomes longer. Fork resolution occurs when the next proof of work is found, and one of the chains becomes longer, determining the selection of the correct chain.

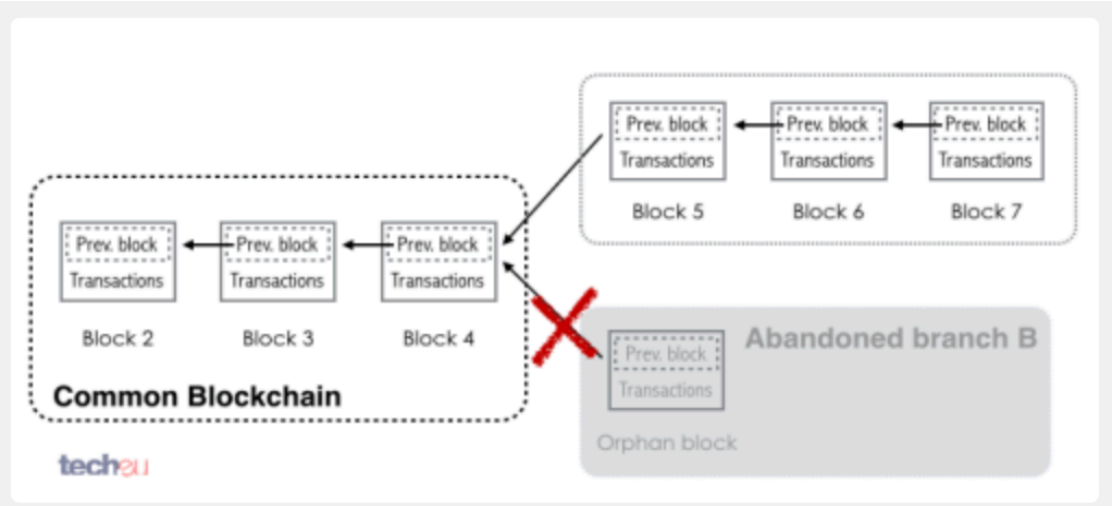


Figure 2.4: Bitcoin Branch Merging (*Source*)

2.1.5.5 Diffusion of Transactions and Blocks

New transactions don't necessarily have to reach all nodes in the network. As long as they reach many nodes, they will be included in a block sooner or later. Furthermore, block diffusion is tolerant to message losses. If a node doesn't receive a block, it will request it when it receives the next block and realizes it missed one.

2.1.6 Incentives

The Bitcoin incentive system plays a crucial role in ensuring network participation and security. This section aims to explain in detail how the Bitcoin incentive system works and how it motivates miners to dedicate computational resources to maintain and protect the network. The transition of incentive mechanisms from new coin issuance to transaction fees will also be discussed, along with the role incentives play in promoting participants' honesty.

2.1.6.1. Production of New Bitcoins and the Role of Miners

Conventionally, the first transaction in a block is a special transaction that creates new currency owned by the block creator. This introduction of new bitcoins provides an incentive for nodes to support the network and offers a way

to initially distribute coins in circulation since there is no central authority to issue them. The constant addition of a fixed quantity of new bitcoins is akin to the activity of gold miners investing resources to introduce gold into circulation. In the case of Bitcoin, the invested resources are computational effort and electricity.

2.1.6.2. Funding Incentives Through Transaction Fees

Incentives can also be funded through transaction fees. If the value of the output of a transaction is less than the value of the input, the difference becomes a transaction fee added to the incentive value in the block containing the transaction. Once a predetermined number of coins are in circulation, incentives can fully transition to transaction fees, making the system inflation-free.

2.1.6.3. Promotion of Honesty and Network Security

Incentives contribute to encouraging nodes to remain honest. If a greedy attacker manages to accumulate more computational power than all honest nodes, they must decide whether to use it to defraud people with double spending or to generate new bitcoins. It's likely they find it more profitable to follow the rules that ensure safe gains rather than compromise the system and risk seeing their wealth devalued amid network uncertainty and concern.

2.1.6.4. The 51% Attack

An attacker controlling the majority of computational power can jeopardize the security of the Bitcoin network. However, the incentive system makes it more advantageous to follow the rules rather than attack the network. If an attacker used their computational power to alter transactions or steal payments, they would compromise trust in the system and consequently the validity of their own bitcoins. Additionally, the cost of a 51% attack increases as the network grows, making the network more resilient to this attack over time.

2.1.7 Privacy

Despite Bitcoin transactions being public by nature, several strategies and techniques exist to maintain a certain degree of anonymity and safeguard user privacy. In this section, we analyze the most significant aspects of privacy in Bitcoin.

2.1.7.1 The Traditional Model and Privacy

The traditional banking model achieves some degree of privacy by limiting information access to the involved parties and a trusted third party. However, in the context of Bitcoin, the need to publicly announce all transactions makes adopting this approach challenging. Privacy can still be maintained by disrupting the flow of information at another point: keeping users' public addresses anonymous. This system is known as pseudonymous. Nevertheless, identifying the owner of a key could reveal other transactions belonging to the same owner.

2.1.7.2 CoinJoin Transactions for Privacy

One of the most used strategies to preserve privacy in Bitcoin is the use of CoinJoin transactions. In these transactions, coins from different owners are mixed together in a single transaction that has multiple outputs of equal amounts. This makes it difficult to link outputs to their respective input transactions, obscuring coin traceability. CoinJoin transactions provide a high level of anonymity and are a common practice for users desiring to protect their privacy.

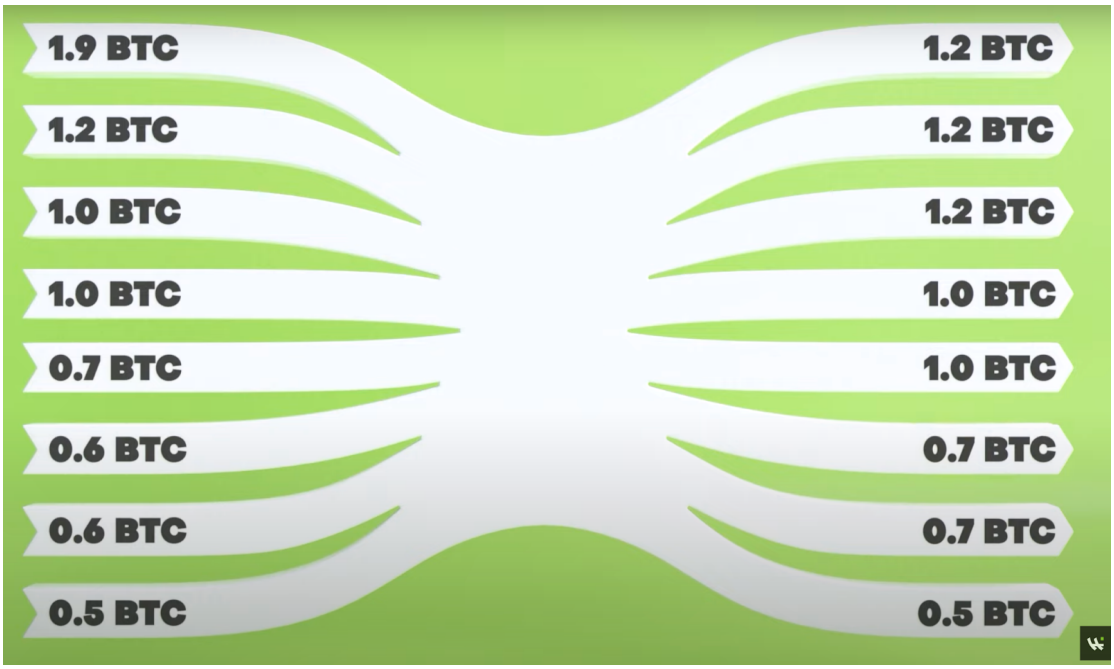


Figure 2.5: CoinJoin (*Source*)

2.1.7.3 Best Practices to Protect Privacy

In addition to CoinJoin transactions, there are other best practices that Bitcoin users can adopt to safeguard their privacy. Some examples include:

- Avoid Reusing Addresses: Using a Bitcoin address for each transaction or even for each fund reception contributes to making it harder to link different transactions.
- Avoid Multi-Input Transactions: Combining multiple coins in a single multi-input transaction can provide clues about the connection between different Bitcoin addresses, jeopardizing privacy.
- Manage Transaction Change Carefully: The "change" from transactions, which is the difference between the input and output of a transaction, can reveal information about coin ownership. Properly handling transaction change can help preserve privacy.

2.2 Lightning Network

2.2.1 Introduction

The introduction of the Lightning Network to Bitcoin offers a decentralized solution to improve scalability and transaction speed. This system, based on micropayment channels and multi-signature contracts, enables instant value transfers off the main blockchain, avoiding transaction fees and relieving the Bitcoin gossip network. The implementation of the Lightning Network promises to revolutionize the efficiency and privacy of Bitcoin transactions, opening new prospects for the Bitcoin network and the entire digital transaction market.

2.2.2 On-Chain Scalability Problems

While the Bitcoin blockchain is innovative and holds great potential, as a payment platform alone, it cannot cover global commerce in the near future. The blockchain is a public ledger containing all Bitcoin transaction history. If every node in the Bitcoin network must be aware of every transaction occurring globally, it can significantly impede the network's capacity. This is due to the storage capacity needed for nodes to store all transactions and the hardware performance required to handle the traffic of a global financial network.

2.2.2.1 The Gossip Protocol

The Bitcoin blockchain relies on a gossip protocol where all state changes are disseminated to all participants. While this approach ensures consensus on the state of the ledger, it presents scalability challenges. The requirement for every node to be aware of every global transaction can be a significant burden on the network's ability to handle a large number of financial transactions.

2.2.2.2 Block Size Limit

Currently, Bitcoin supports fewer than 7 transactions per second with a 1-megabyte block size limit. In comparison, the Visa payment network can process thousands of transactions per second during peak periods. To achieve a capacity similar to Visa's, Bitcoin would need blocks of around 8 gigabytes every ten minutes, which would entail storing and validating a massive amount of data.

2.2.2.3 Centralization and Trust Issues

Introducing such large blocks would imply centralization, as only a few entities would have the resources and infrastructure to validate them. This concentration of power contradicts Bitcoin's decentralized nature, compromising its security and making it more susceptible to the control of centralized entities. To maintain the trust and security provided by Bitcoin, it's necessary to mitigate these centralization effects.

2.2.3 Payment Channels

Payment channels are the foundational components of the Lightning Network. A payment channel is a Bitcoin transaction locked on the blockchain, serving as a starting point for conducting off-chain transactions between two participants. Once the payment channel is established, transactions can occur without involving the main Bitcoin blockchain.

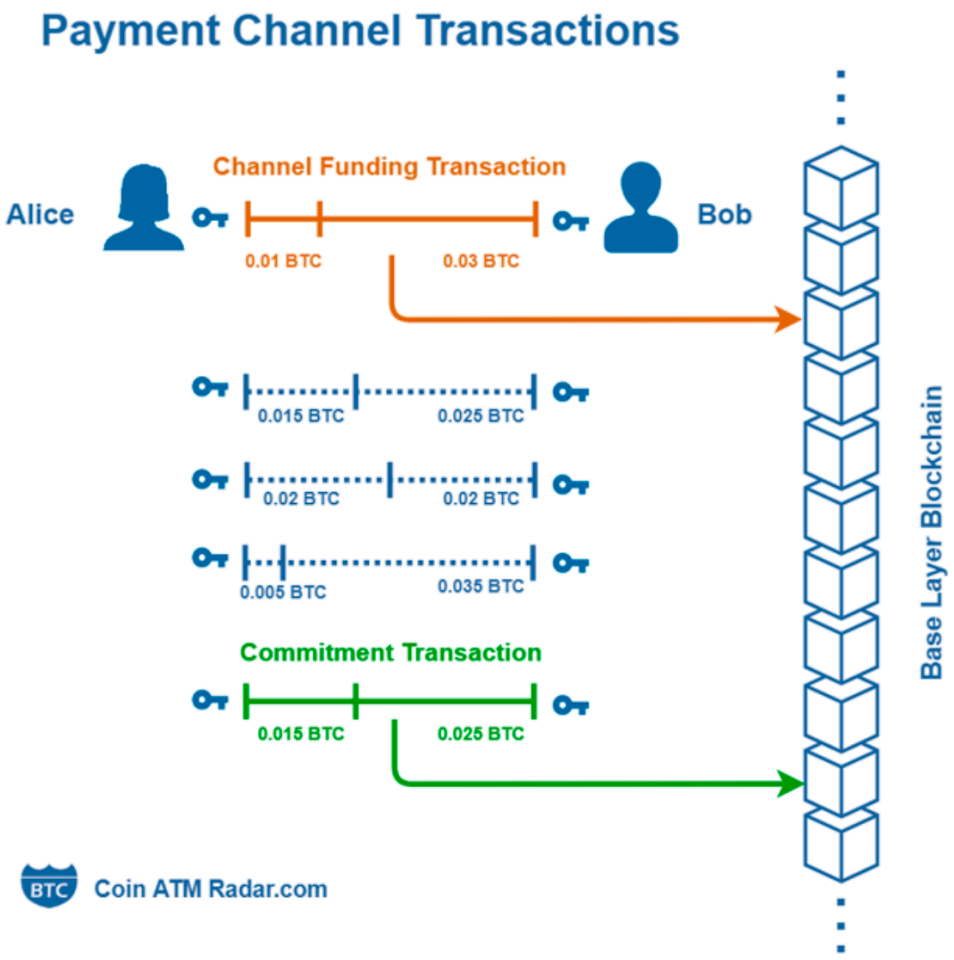


Figure 2.6: Payment channel transactions (*Source*)

2.2.3.1 Operation of a Channel

When two participants decide to open a payment channel, they create a transaction on the blockchain that locks a certain amount of Bitcoin in a smart contract. This amount represents the initial balance of the payment channel, so the transaction is called the "funding transaction." This transaction includes the input of funds from the participants and a special output called the "multi-sig output." This output requires the

signature of both participants to spend the funds.

The "funding transaction" also contains important information such as time locks, which specify the time conditions under which transactions can occur within the channel. These time locks establish the rules for channel access and closure, ensuring secure operation and preventing fraud or abuse. During the channel opening phase, participants also exchange what are known as "revocation keys." These keys are crucial for channel security, allowing participants to revoke previous transactions in case one party attempts to defraud the other. Revocation keys are generated using a specific cryptographic function and are exchanged between participants to ensure both have the ability to reliably revoke previous transactions. Once the opening transaction is confirmed on the main Bitcoin blockchain, the channel is considered open, and participants can start updating their balances through off-chain transactions within the channel. These transactions occur between participants and do not require recording on the main blockchain, enabling fast and scalable transactions within the channel. It's important to note that Lightning Network payment channels are not static but can be closed or updated at any time. Closing a channel involves recording a final transaction on the main Bitcoin blockchain, while balance updates within the channel are carried out through a series of off-chain transactions between participants.

2.2.3.2 Trustless Nature of Payment Channels

"Just as an ancient question about whether a falling tree makes a sound, if all parties agree that the tree fell at 2:45 pm, then the tree indeed fell at 2:45 pm. Similarly, if both parties agree that the current balance inside a channel is 0.07 BTC for Alice and 0.03 BTC for Bob, then that balance is indeed real. However, without encryption, an interesting problem arises: if the counterparty disagrees on the current balance of the funds (or on the time when the tree fell), it becomes a matter of word against word. Without cryptographic signatures, the blockchain won't know who owns what. If the balance in the channel is 0.05 BTC for Alice and 0.05 BTC for Bob, and the balance after a transaction becomes 0.07 BTC for Alice and 0.03 BTC for Bob, the network must know which set of balances is correct. Both parties can commit to signing a transaction without broadcasting it. Thus, if Alice and Bob commit funds to a 2-of-2 multisignature address (requiring consent from both parties to spend), they can agree on the current balance state. Alice and Bob can agree to create a refund from that 2-of-2 transaction for themselves, say 0.05 BTC each. This refund isn't broadcast to the blockchain. One party can do it, but it can also choose to hold that transaction, knowing they can redeem the funds whenever they want. To update the balance, both parties create a new spending from the 2-of-2 multisignature address, say 0.07 for Alice and 0.03 for Bob. However, without proper design, the issue of timestamping and information about which

of the two transactions is correct arises: the new transaction or the original refund. The restriction on timestamping and dates, however, is not as complex as the full ordering of all transactions as in the Bitcoin blockchain. In the case of payment channels, only two states are required: the current correct balance and any previous deprecated balance. Thus, a Bitcoin script can be devised where all old transactions are invalidated, and only the new transaction is valid. Invalidation is enforced by a Bitcoin output script and dependent transactions that compel the other party to give all their funds to the channel counterpart. By taking all the funds as a penalty to give to the other party, all old transactions are thus invalidated. This invalidation process can exist through a channel consensus process where, if both parties agree on the current ledger state (and on creating new states), the actual balance is updated. The balance is reflected on the blockchain only when a single party disagrees. Conceptually, this system is not an independent overlay network; rather, it is a deferral of state within the current system, as the application still occurs on the blockchain itself (though deferred to future dates and transactions)."

2.2.3.3 Payment Channels Network

Micropayment channels establish a relationship between two parties. However, requiring everyone to create channels with everyone else doesn't solve the scalability issue. Bitcoin's scalability can be achieved by using an extensive network of micropayment channels. Assuming the existence of a large network of channels on the Bitcoin blockchain, and all Bitcoin users participating in this graph by having at least one open channel on the Bitcoin blockchain, it becomes possible to create an almost infinite number of transactions within this network. Moreover, by using staggered timeouts, it's possible to send funds through multiple intermediaries in a channel network without the risk of intermediaries stealing the funds.

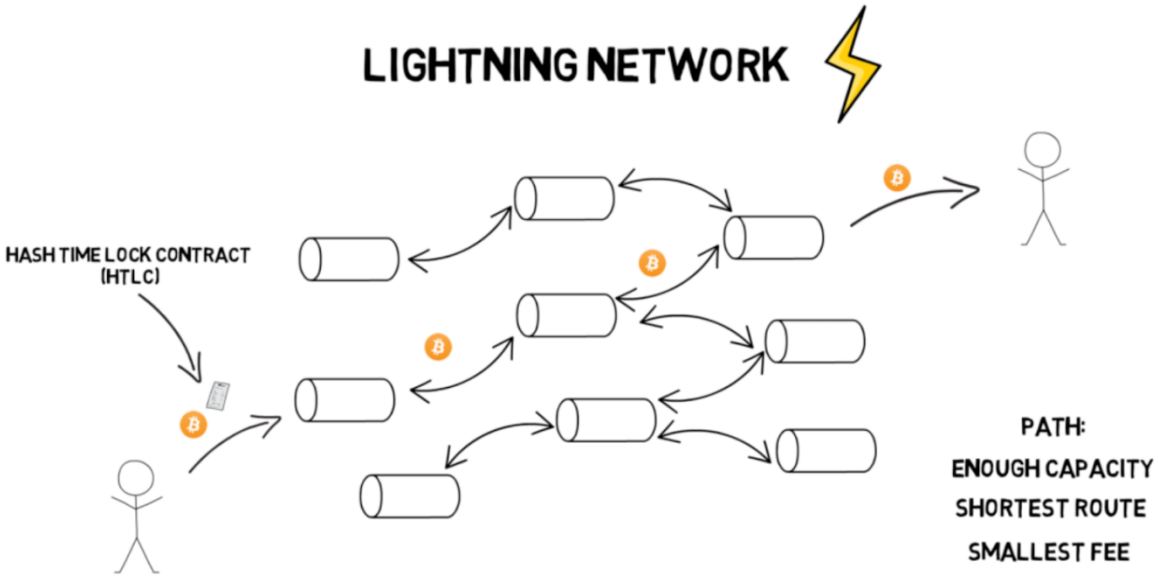
2.2.4 Payments Routing

Within the Lightning Network, a payment is initiated by creating an "invoice." An invoice is a unique code representing the payment amount and is generated by the payment recipient. The invoice includes information such as the amount, an expiration timestamp, and a public key to identify the recipient. When a sender wants to make a payment, they create an "onion packet" containing encrypted data. This onion packet is designed so that only the initial sender knows the final recipient's address. Each layer of encryption in the onion packet represents an intermediate step in the network. The sender sends the onion packet to their nearest payment channel. Each payment channel contains a shared balance among participants and is used to route the payment through the

network. Intermediate nodes on the Lightning network, known as "routers," receive the onion packet and, using information in encrypted metadata, determine the next payment destination. At each hop, the corresponding node removes one layer of encryption and forwards the onion packet to the next node until the payment reaches the final recipient. Once the payment reaches the final recipient, the destination node verifies that the amount matches the initially created invoice. If the amount is correct, the destination node sends a payment confirmation to the sender, and the payment is considered complete. In summary, payment routing in the Lightning Network involves the use of an onion structure, intermediate nodes routing through shared payment channels, and intermediary nodes determining fees. This process enables instant, secure, and low-cost transactions within the Lightning network.

2.2.4.1 Route Calculation

In the Lightning Network, payment routing occurs even without complete knowledge of the network's topology. This is made possible by using an algorithm called "source routing." When a sender wants to make a payment, their Lightning Network client queries nearby nodes with which they have an open channel to obtain information about the network's topology. This information includes available channels, channel balances, and routing policies of intermediate nodes. Using this information, the client can create an approximate view of the network and build a potential route for the payment. The source routing algorithm works by sending payment requests to intermediate nodes along the potential route. Each intermediate node responds with information about channel.



During the routing process, fees are calculated for the routing service. These fees are defined by intermediate nodes and can vary depending on factors such as network congestion, channel capacity, and node policies. Fees can be based on a percentage of the payment amount or a fixed fee.

Figure 2.7: Lightning Network Routing (*Source*)

2.2.4.2 Routing Fees

Routing fees are calculated and paid for the service provided by intermediate nodes that route payments across the network. These fees are an important component to incentivize intermediate nodes to participate in payment processing and maintain network efficiency and security.

Routing fees can be calculated in various ways, depending on the routing policy implemented by intermediate nodes. Two common approaches for fee calculation are based on the percentage of the payment amount or a fixed fee.
In the case of a percentage-based fee, intermediate nodes can charge a percentage of the payment amount as a fee. For example, if a node sets a 1% fee and the payment is 0.1 BTC, then the fee will be 0.001 BTC.
In the case of a fixed fee, intermediate nodes can set a fixed fee regardless of the payment amount. For instance, a node might charge a fixed fee of 0.0001 BTC for every payment routed through it.
Once the routing fees are calculated, the payment can proceed along with the actual payment of fees to the intermediate nodes along the route. This is done using a function called "fee update."

When a payment reaches an intermediate node, the due fee for that node is calculated based on the chosen routing policy. The node then sends a fee update message to the sender, which includes the fee amount and necessary information to identify the node. The sender verifies and accepts the fee update and sends the fee amount to the intermediate node.
This process continues for each intermediate node along the route until it reaches the final recipient. Each intermediate node will receive its fee during the payment routing.

It's important to note that fee calculation and structure can vary from node to node and can be influenced by factors like network congestion, channel capacity, and individual node policies. Therefore, routing fees can be dynamic and subject to change over time.
Furthermore, intermediate nodes can also apply "fee balancing" policies to incentivize proper network operation. For instance, a node might prefer routing payments that help redistribute channel balances or maintain balance between channels more efficiently.
It's clear that in the Lightning Network, the scarce resources for which fees are paid are liquidity and connectivity, unlike Bitcoin, where block space is the scarce resource.

2.2.5 The Bitcoin Lightning Network

By using a micropayment channel with hash locks and timelocks limited contracts, transactions can be settled on a multi-hop payment network using a series of decreasing timelocks without the need for additional central clearinghouses.
Traditionally, financial markets settle transactions by transferring the delivery obligation to a central point and regulating ownership transfer through this central hub. Payment and fund transfer systems (such as ACH and the Visa card network) and stock clearinghouses (like DTCC) operate in this way.

As Bitcoin enables programmable money, it's possible to create transactions without having to involve a central clearinghouse. Transactions can occur off-chain without the need for third parties to collect all funds before distributing them: only non-collaborative channel counterpart transactions are automatically adjudicated on the blockchain.

The obligation to deliver funds to a final recipient is achieved through a chain delegation process. Each participant along the path takes on the obligation to deliver to a specific recipient. Each participant passes this obligation to the next participant in the path. The obligation of each subsequent participant along the path, defined in their respective HTLCs (Hashed Time-Lock Contract), has a shorter completion time compared to the previous participant. This way, each participant can be sure to claim the funds when the obligation is passed along the path.
Bitcoin transaction scripting, a form of what some call "Smart Contracts," allows systems without trusted clearinghouses or custodial services.

2.3 Smart Contracts

2.3.1 Introduction

Smart contracts represent autonomous digital contracts that enable transaction execution and process automation without the need for central intermediaries, ushering in a new era of trust and transparency in digital transactions.

Smart contracts form the foundation for the concept of "programmable money." This vision transforms money from a mere exchange tool into a programmable digital asset capable of executing specific instructions and conditions automatically and autonomously. Through contract programmability, rules, logic, and conditions determining how money is managed, distributed, and used can be defined, enabling a wide range of innovative applications beyond traditional financial transactions.

The introduction of programmable money opens the door to scenarios such as decentralized finance, tokenization of real assets, automated payments, conditional transactions, and much more, revolutionizing how we conceive of and interact with money in the digital age.

A central aspect of the analysis concerns smart contract implementation approaches, considering both traditional models and the new frontiers represented by autonomous smart contracts and those based on formal logic. We will understand that the most commonly used platforms today do not guarantee fundamental properties such as decentralization or censorship resistance and we will identify new development frontiers.

2.3.2 Fundamentals of Smart Contracts

2.3.2.1 Definition of Smart Contract

Smart contracts are a form of digital contracts that are executed and enforced automatically through the use of blockchain technology. These contracts are coded in a specific programming language, like Solidity for Ethereum, and are immutable once registered on the blockchain. The essence of smart contracts lies in their ability to automate the execution of agreements and transactions without the need for centralized intermediaries.

A smart contract can be defined as a set of rules and logic governing interactions and transactions between involved parties. Each smart contract contains a series of predefined conditions that must be met for operations to be executed automatically and autonomously. Once these conditions are verified, the contract is triggered, and the execution of the specified actions is initiated on the blockchain.

A key aspect of smart contracts is their autonomy, meaning they can act without the need for a central authority to oversee or control the operation. This is made possible by the distributed nature of blockchain technology, where each network node participates in transaction validation and smart contract management. Furthermore, smart contracts are immutable, meaning that once they're registered on the blockchain, they cannot be changed or altered.

2.3.2.2 Key Characteristics of Smart Contracts

First and foremost, autonomy is a fundamental characteristic of smart contracts. They can operate without the need for human intermediation or a central

authority. Once triggered, smart contracts automatically execute the specified instructions and conditions without requiring the intervention or trust of third parties. This autonomy allows for the automation of complex processes and the elimination of intermediaries, leading to greater efficiency and reducing the likelihood of errors or fraud.

Immutability is another key characteristic of smart contracts. Once a contract is published and registered on the blockchain, it cannot be modified or deleted. Immutability ensures that the established contract conditions and logic remain intact and cannot be arbitrarily altered by any of the involved parties. This attribute provides greater security and trust in the operation of smart contracts, as users can rely on the fact that the established rules will be upheld without manipulation or external interference.
Deterministic execution is another fundamental characteristic. This means that the execution of a specific smart contract will always yield the same result, regardless of when it is executed or which node performs the execution. This deterministic property is ensured by the processes by which these contracts are formulated and the distributed nature of the blockchain, where each node must agree on the contract's state. Deterministic execution offers greater predictability and consistency in outcomes, contributing to trust in interacting with them.

2.3.2.3 Oracles

Oracles play a crucial role in the smart contract ecosystem, as they enable the retrieval of data external to the blockchain and its use within contracts. Smart contracts based on blockchain technology can execute code autonomously and deterministically, but without a mechanism to access external information, they would be limited in their functionalities and interoperability with the outside world.

Oracles act as intermediaries between the blockchain and external data sources, providing smart contracts with access to information such as financial prices, weather data, sports event outcomes, and more. However, the use of oracles introduces challenges related to trust in a third party.

One of the main issues is trust in the oracle itself. Since oracles are entities external to the blockchain, one must rely on their correctness and reliability in providing the requested data. If an oracle provides unreliable or compromised data, smart contracts based on that information could make incorrect decisions or be subject to manipulation.

Another issue relates to the security of communication channels between oracles and the blockchain. As data needs to be transmitted between the two

entities, there's a risk of such communications being intercepted or manipulated by malicious actors. This could lead to incorrect information or attacks against the blockchain itself.

To address these challenges, various alternatives have been proposed to reduce reliance on traditional centralized oracles and mitigate the risks associated with trusting a third party. One possible solution is decentralized oracles, which leverage "crowdsourced wisdom" or collective reputation to ensure data correctness. In this approach, multiple independent oracles provide the same information, and the majority of results are considered reliable.

Another option is oracles based on cryptography and zero-knowledge proof cryptography. These oracles allow the verification of data correctness without revealing the entire information, thus protecting privacy and reducing dependence on an intermediary.

2.3.3 Platforms for Smart Contracts

2.3.3.1 Distributed Virtual Machine Approach

This is the most common approach and is followed by various platforms like Ethereum and many others.
When a smart contract is executed in a distributed virtual machine within a blockchain network, the execution process follows a series of steps ensuring the correctness and security of operations.

Firstly, the smart contract is written using a specific programming language, like Solidity for Ethereum. The contract is then compiled into bytecode, representing the executable version of the contract.
Once the contract is compiled, it is deployed to the blockchain network. This occurs through a special transaction called a "contract creation transaction." During this transaction, the contract bytecode and any initialization parameters are specified.
Once the contract creation transaction is included in a blockchain block, the contract is permanently stored on the blockchain. The contract is then associated with a unique address, which serves as an identifier for future interactions.

When you want to interact with an existing smart contract, a transaction containing the contract's address and required parameters is sent. This transaction is broadcasted across the blockchain network and included in a block. Once the transaction is confirmed, the contract is executed.
The execution of a smart contract occurs within a distributed virtual machine called the EVM (Ethereum Virtual Machine) in the case of Ethereum. The EVM

is designed to deterministically execute smart contract bytecode and ensure that all transactions are consistently validated across all nodes on the network.

During execution, the EVM reads the contract bytecode and interprets the instructions one by one. These instructions can include mathematical operations, access to contract state variables, and calls to other contracts. The execution of instructions may also involve modifying the contract's state, updating participant balances, or issuing new transactions.

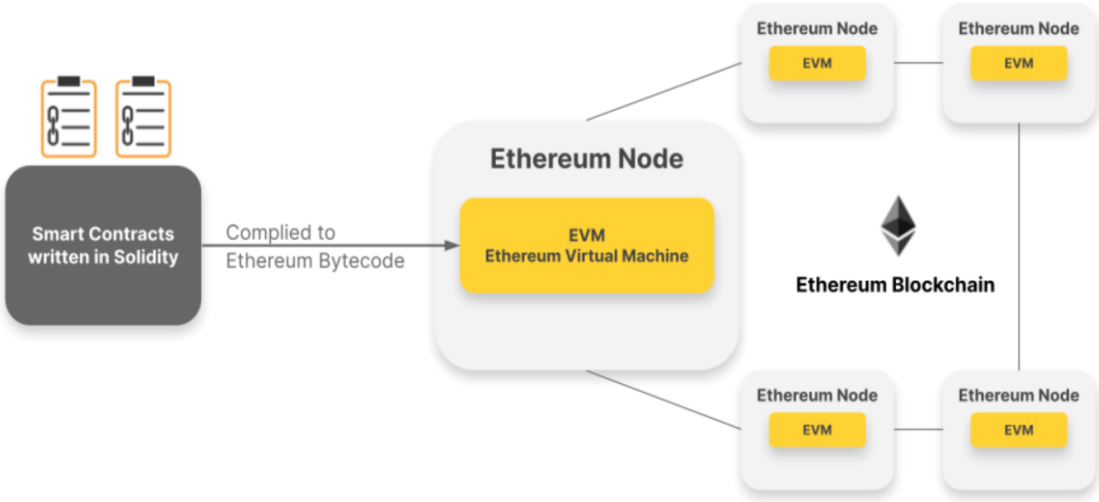


Figure 2.8: Ethereum Virtual Machine (*Source*)

It is important to note that the execution of smart contracts occurs on all participating nodes of the blockchain network, which collaborate to reach consensus on the final state of the contract. This ensures immutability and data security since all nodes must agree on the same result. However, on the other hand, this leads to a series of chain events that jeopardize the fundamental properties of the network.

For example, the growth of node size is an important factor to consider during the execution of a smart contract. As contracts are permanently stored on the blockchain, every participating node in the network must store the entire history of contracts. This can lead to a significant increase in node size over time, as each new added contract contributes to further growth of data stored on each node. This growth can pose a challenge to the network's scalability as it requires increasing resources for data storage and maintenance.

A consequence of complete contract storage is the potential loss of decentralization. Since all nodes must retain the entire history of contracts, control and network management can become more concentrated in the hands of a few participants with sufficient resources to support this growing data size. This can lead to reduced participant diversity and potential centralization of decision-making power within the network.

Furthermore, one of the main concerns related to smart contract execution is the issue of privacy. Since contracts are entirely public and accessible to all blockchain participants, transactions and data associated with the contract are visible to everyone. This lack of privacy can limit the adoption of some smart contract applications in sectors that require a high level of confidentiality, such as identity management or handling personal information.

To address these issues, various solutions have been proposed. For instance, the use of data compression techniques could reduce node size without compromising information validity. Additionally, implementing encryption mechanisms could protect the privacy of transactions and sensitive data within smart contracts. The most plausible solution seems to be client-side validation of contracts, as we will see in the next paragraph.

2.3.3.2 Client-Side Validation Approach

An alternative to executing smart contracts on a distributed virtual machine is client-side validation, which introduces new concepts like client-side validation and single-use sealing. This approach has been adopted in projects like RGB and TAP (Taproot Asset Protocol). To explain in detail how the client-side validation paradigm works, we'll analyze the design of RGB.

RGB is a protocol designed to issue and transfer digital assets and other generic rights on the Bitcoin blockchain through the use of customizable smart contracts outside of the blockchain itself.

This protocol leverages the client-side validation paradigm to keep all RGB-specific data off the blockchain, using the Bitcoin blockchain solely as a commitment layer to protect against double spending.
This allows for greater scalability since blockchain usage is reduced, increased privacy since no data is visible to blockchain observers, and enhanced flexibility as the protocol is not constrained by Bitcoin transaction data structure.

The design of RGB focuses on creating and transferring digital assets on the Bitcoin blockchain using customizable off-chain smart contracts. This protocol aims to provide a scalable, flexible, and private solution for digital asset tokenization and rights management.

The protocol employs customizable off-chain smart contracts to define and manage digital assets. Assets are described through deterministic contract schemes that establish rules for asset creation, transfer, and redemption. These contracts define the conditions and logics governing asset behavior, enabling a high degree of customization.

Asset management within RGB occurs through "Consignment Contracts." These contracts act as custodians for digital assets and define the conditions for asset transfer and redemption. Consignment Contracts are used to protect and ensure proper asset management within the RGB system.

A distinctive aspect of RGB's design is the use of "single-use seals." These seals create a unique bond between digital assets and the contract that manages them, ensuring asset integrity and authenticity throughout their lifecycle. Single-use seals enable efficient and secure asset integrity verification without the need for complex smart contracts on the blockchain. In the case of RGB, the single-use seals used are essentially Bitcoin UTXOs.

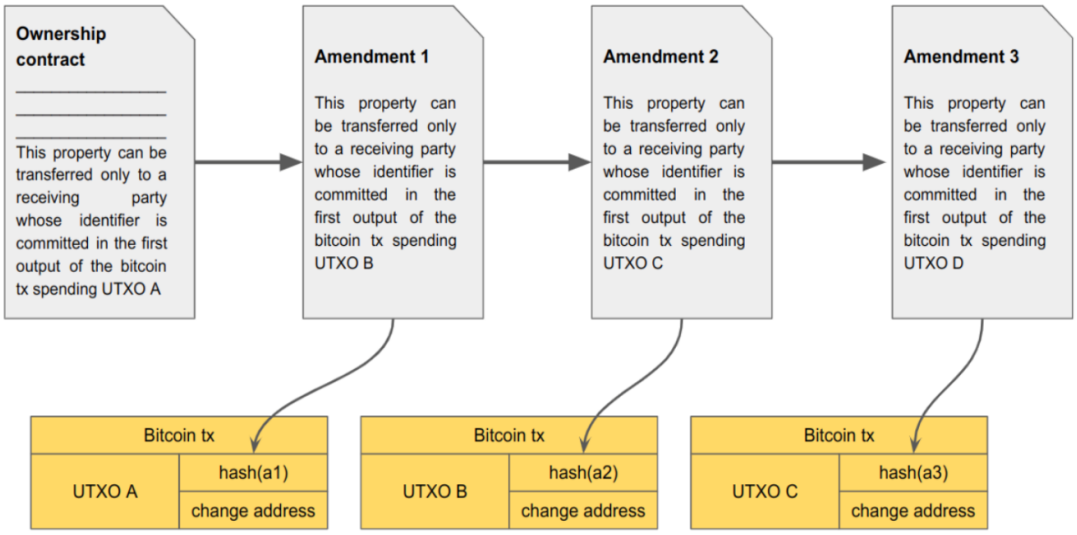


Figure 2.9: RGB single use seal design (*Source*)

Finally, RGB also offers flexibility in asset design, allowing for the definition of both fungible and non-fungible assets, as well as complex and hierarchical assets. This enables the creation of a wide range of digital assets with different characteristics and behaviors.

2.3.4 Applications of Smart Contracts

2.3.4.1 Fungible Token (Securities)

Smart contracts offer a wide range of applications in the securities sector. Through fungible tokens, it is possible to represent and transfer ownership rights to traditional financial assets such as stocks, bonds, and other financial instruments. Fungible tokens represent interchangeable units of value and enable the creation of liquid and accessible markets for the trading of such assets. For example, a company can issue tokens representing its shares and use smart contracts to manage the issuance, transfer, and distribution of profits to token holders. Smart contracts enable the programmability of rules and conditions, facilitating the automation of dividend payments, voting, and more. Additionally, the transparency provided by the blockchain gives investors greater confidence and visibility into transactions and the status of securities.

2.3.4.2 NFT / Collectibles

Smart contracts are widely used in the context of digital collections and Non-Fungible Tokens (NFTs). NFTs represent unique digital assets, such as artwork, music, videos, games, and virtual objects. Smart contracts allow the creation, transfer, and management of these unique assets, providing proof of authenticity and ownership. For example, an artist can create a digital artwork as an NFT and use a smart contract to establish ownership rights, sales, and automatic royalties for each subsequent transaction. NFTs enable the creation of global markets for digital artworks, with authentication and traceability guaranteed by the blockchain. Moreover, smart contracts can define specific rules for NFTs, such as the ability to access exclusive content, participate in special events, or receive rewards.

2.3.4.3 Other Applications

In addition to securities and NFTs, smart contracts find application in a wide range of sectors and use cases. For instance, smart contracts can be used for supply chain management, enabling the traceability of products throughout the entire production and distribution process. They can also be used for the creation of decentralized voting and governance systems, ensuring transparency and integrity of information. Additionally, smart contracts can support the creation of financial services such as loans, asset exchanges, insurance, and other forms of traditional financial services. These are just a few examples of

the numerous possibilities offered by smart contracts, which are revolutionizing various industries.

2.4 Autonomous Driving

2.4.1 Introduction to Autonomous Driving Vehicles

2.4.1.1 Context and Impact of Autonomous Driving Vehicles

Autonomous driving vehicles represent one of the most revolutionary innovations in the automotive sector and have a significant impact on society, the economy, and the environment.

The context of autonomous driving vehicles is characterized by a growing demand for safer, more efficient, and convenient mobility solutions. The primary goal of these vehicles is to reduce human errors, which are responsible for a large portion of road accidents, thereby improving road safety and safeguarding human lives. Additionally, autonomous driving vehicles promise greater traffic efficiency, reduced fuel consumption, and pollution through their ability to adapt to real-time traffic conditions and optimize routes.

Furthermore, this technological innovation is bringing about changes in the automotive sector and mobility industry, with new players emerging and the potential to develop new business models based on shared autonomous vehicles. However, the context of autonomous driving vehicles also presents significant challenges, such as the need to develop appropriate regulations and standards, address concerns about data privacy and security, and overcome public acceptance and adoption.

2.4.1.2 Definition and Classification of Autonomous Driving Vehicles

The definition of an autonomous driving vehicle is based on its ability to operate independently without direct human intervention, using advanced technologies such as sensors, artificial intelligence, and control systems.

Autonomous driving vehicles are classified based on a series of automation levels, as defined by the SAE International standardization body. These levels range from 0 to 5, where level 0 represents a vehicle entirely driven by a human with no automatic assistance, while level 5 indicates a fully autonomous vehicle capable of operating independently without human intervention under any driving condition.

It's important to note that when referring to autonomous driving, it encompasses not only privately owned automobiles but also extends to all modes of transportation for goods or people, whether public or private.

2.4.2 Architecture of Autonomous Driving Vehicles

2.4.2.1 Key Components of an Autonomous Driving Vehicle

In the context of autonomous driving vehicles, various components work together to enable safe and efficient operation. These components play crucial roles in ensuring environment perception, data processing, and vehicle control.

One of the main components is sensors, which gather information about the surrounding environment. Sensors may include cameras, lidar (light detection and ranging), radar, and ultrasonic sensors. Cameras capture images and videos to understand the road scene. Lidar sensors use laser pulses to measure the distance and shape of surrounding objects, providing a 3D representation of the environment and enabling precise perception of the distance, shape, and position of moving objects. Radar emits radio waves to detect moving objects, while ultrasonic sensors detect the distance from nearby objects. This combination of sensors allows the vehicle to obtain a comprehensive view of the surroundings.

Another crucial component is data processing systems. These systems process the information collected by sensors and analyze it to understand the road scene, recognize objects, identify road signs, assess the speed and trajectory of objects, and make real-time decisions. The computing power required for these operations is provided by central processing units (CPUs), graphics processing units (GPUs), and other high-speed data processing technologies.

Vehicle control is another fundamental element. Autonomous vehicles use actuators such as motors and brakes to control vehicle movement. These actuators are managed by control systems that translate decisions made by the data processing system into physical commands for the vehicle. Vehicle control takes into account road regulations, traffic conditions, and interactions with other road users to ensure safe and respectful driving.

Furthermore, autonomous driving vehicles can also integrate vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication technologies. These technologies enable vehicles to exchange information with other vehicles and

the surrounding road infrastructure, enhancing situational awareness and facilitating cooperation between vehicles for safer and more efficient driving.

2.4.2.2 Data Processing and Artificial Intelligence Systems

Autonomous driving vehicles utilize a combination of machine learning and deep learning algorithms to analyze and understand data from sensors and make autonomous decisions based on that information.

One of the primary approaches used is deep learning, which relies on deep neural networks. These deep neural networks are capable of autonomously learning from large amounts of data and identifying complex patterns and relationships within the data. For example, convolutional neural networks (CNNs) are often used for image analysis and object identification within the road environment. Other neural network architectures such as recurrent neural networks (RNNs) or generative adversarial networks (GANs) can be employed to perform specific tasks such as predicting the actions of other vehicles or generating synthetic data for model training.

To implement AI algorithms within autonomous vehicles, powerful processing hardware is required, such as high-performance processors, graphics processing units (GPUs), and artificial intelligence accelerators. This hardware enables real-time execution of the intensive calculations required by AI models, allowing the vehicle to make autonomous decisions and respond to road environment dynamics.

AI algorithms are trained on large datasets, which may include images, videos, lidar data, and other sensor information. These datasets are used to train AI models through supervised or unsupervised learning processes, in which the model learns to recognize patterns and make accurate predictions.

Once trained, AI models are implemented within the vehicle's control system, which uses them to constantly analyze sensor data, recognize objects, interpret road signs, predict the actions of other road users, and make decisions based on this information.

2.4.3 Levels of Automation and Autonomous Driving

Classifying levels of vehicle automation helps understand the degree of autonomy and human involvement required during driving. The SAE

International standardization body has established a scale of five levels of automation, ranging from level 0 to level 5.

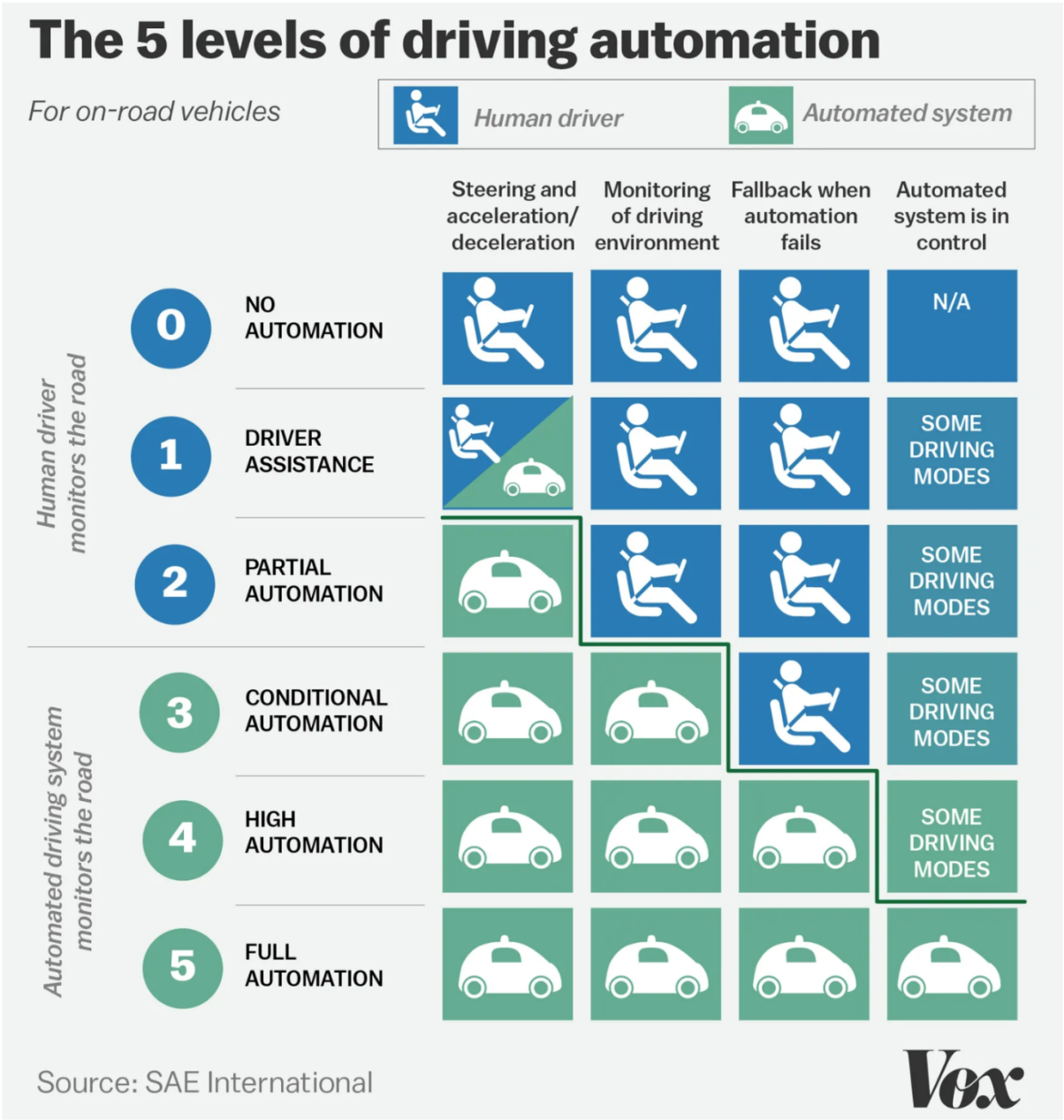


Figure 2.10: SAE international automation levels (*Source*)

- **Level 0: No Automation**

The vehicle is fully driven by humans without any automatic support. All driving and vehicle control functions are under the complete control of the human operator.

- **Level 1: Driver Assistance**

The vehicle offers some driver assistance functions that can provide support in certain situations. However, the vehicle's control is still in the hands of the human driver, who must remain actively engaged in driving and supervision.

- **Level 2: Partial Automation**

The vehicle is capable of performing some driving functions autonomously, such as lane-keeping, speed control, and automatic parking. However, the driver must remain actively engaged in driving and be ready to take control of the vehicle if necessary.

- **Level 3: Conditional Automation**

The vehicle is capable of performing certain driving functions autonomously and can handle driving operations under specific conditions and contexts. However, the driver must be able to intervene and regain control of the vehicle if required by the system.

- **Level 4: High Automation**

The vehicle is capable of operating autonomously in most situations and contexts without requiring active human driving. However, there might be limitations, such as in certain geographic areas or adverse weather conditions, that require human involvement.

- **Level 5: Full Automation**

The vehicle is fully autonomous and able to operate independently without any human intervention. There are no context-specific or specific condition restrictions, and the vehicle can handle all driving situations without the need for a human driver.

2.4.4 Challenges and Future Perspectives

2.4.4.1 Regulatory and Legislative Challenges for the Adoption of Autonomous Vehicles

The adoption of autonomous vehicles is accompanied by significant regulatory and legislative challenges. While technology is advancing rapidly, it is essential to develop an adequate legal framework to regulate the safe and responsible use of autonomous vehicles.

Currently, many jurisdictions are working to adapt existing laws and regulations to address the specificities and implications of autonomous vehicles. Challenges involve various aspects, such as legal liability in case of accidents, defining the roles and responsibilities of human drivers and autonomous driving systems, as well as vehicle safety and personal data protection.

Furthermore, international regulatory harmonization is necessary to enable global adoption of autonomous vehicles. Another crucial aspect is the interaction with road users and the sharing of road infrastructure with autonomous vehicles. Regulations and laws must also address complex ethical issues, such as priority decisions in emergency situations. Addressing these challenges requires close collaboration between governments, the automotive industry, regulatory organizations, and stakeholders to develop an appropriate regulatory framework that promotes innovation, ensures road safety, and instills public trust in the adoption of autonomous vehicles.

2.4.4.2 Implications for Safety and Data Privacy in Autonomous Vehicles

Autonomous driving raises significant implications for safety and data privacy. Firstly, to ensure safe operation of autonomous vehicles, cybersecurity challenges need to be addressed. Autonomous vehicles are complex computer systems that can be vulnerable to cyberattacks, such as data theft, control system intrusion, or vehicle sabotage. Therefore, advanced security measures like data encryption, secure authentication, and communication protection are essential to ensure integrity and confidentiality of sensitive information.

Secondly, autonomous driving involves the collection and processing of a vast amount of data, including location data, images, videos, and other sensory information. Proper data management is crucial for individual privacy. Clear policies for data collection, use, and retention, ensuring informed consent, and compliance with privacy regulations are necessary. Additionally, anonymization

of personal data and the adoption of access control mechanisms can contribute to protecting user and driver privacy.

Data sharing between autonomous vehicles and road infrastructure also raises further safety and privacy issues. Wireless communication between vehicles and the surrounding environment must be protected against potential attacks or intrusions. Furthermore, it's essential to ensure that shared data is used only for legitimate purposes and adequately anonymized to prevent disclosure of personal information.

Finally, the aspect of legal responsibility and attribution of responsibility in case of incidents or damages related to autonomous vehicles raises additional safety and privacy concerns. Clear obligations and responsibilities of involved parties, including vehicle manufacturers, service providers, and end-users, need to be established to ensure an adequate legal framework and appropriate protection for all parties involved.

Chapter 3

State of the Art

In this chapter, we will analyze the state of the art of various technologies relevant to this study. We will start with autonomous vehicles and then move on to in-car payment systems and currently available decentralized identity protocols on the market. The aim is to understand the current situation, what can be used for our purposes, and what needs to be innovate.

3.1 Autonomous Vehicles

In recent years, several automotive and technology companies have heavily invested in the development of autonomous vehicles. Technical and regulatory challenges have been tackled with determination, leading to significant progress in the field. Different levels of autonomy have been achieved, as defined by the Society of Automotive Engineers (SAE), enabling vehicles to operate under specific conditions without human intervention.

However, despite the achieved progress, there are still significant challenges to overcome. Safety, legal liability, integration with road infrastructure, and public trust are just a few critical aspects that require ongoing attention.

In this section, we will examine major technological advancements, levels of autonomy achieved, and key players in the autonomous vehicle industry.

3.1.1 Market Benchmarking of Autonomous Vehicles

Here, we analyze four leading companies in this sector, highlighting the achieved level of autonomy and listing completed experimental activities.

a. **Tesla**:
   - Achieved Level of Autonomy: Tesla has achieved Level 2 autonomy according to the SAE classification. Its Autopilot system offers advanced driver assistance functions, including acceleration control, braking, and lane-keeping.
   - Experimental Activities: Tesla has tested its autonomous driving technologies on public roads in various countries, including the United States. It utilizes data collected from its vehicles to continue improving and developing its autonomous driving system.

b. **Waymo:**
   - Achieved Level of Autonomy: Waymo has achieved Level 4 autonomy according to the SAE classification. Its vehicles can drive completely autonomously in specific predefined geographic areas without human intervention.
   - Experimental Activities: Waymo has conducted extensive trials on public roads in various US cities, including Phoenix, Arizona, where it launched an autonomous taxi service called Waymo One. It has also tested in various weather conditions and is gradually expanding its geographic presence.

c. **Cruise:**
   - Achieved Level of Autonomy: Cruise has achieved Level 4 autonomy according to the SAE classification. Its autonomous vehicles can operate completely autonomously in specific predefined geographic areas without human intervention.
   - Experimental Activities: Cruise has conducted tests on public roads, particularly in San Francisco, California. It collaborated with local authorities to obtain necessary approvals for testing its autonomous vehicles on highly trafficked public roads and addressing unique challenges associated with complex urban environments.

d. **Uber:**
   - Achieved Level of Autonomy: Uber temporarily suspended autonomous vehicle testing after a fatal accident in 2018. At the time of suspension, Uber had achieved Level 3 autonomy, where vehicles can drive autonomously but still require a human driver ready to intervene.
   - Experimental Activities: Uber experimented with its autonomous vehicles on public roads in various US cities, including Pittsburgh, Pennsylvania, and Tempe, Arizona. After a tragic accident Uber suspended testing. They revised its autonomous driving strategy and

initiated collaborations with other companies to continue developing safe autonomous vehicles.

3.2 In-Car Operating Systems

3.2.1 Introduction

In-car operating systems provide a view of the external world and enable the management of functions and operations of autonomous vehicles. Apart from managing autonomous driving functions, these systems also offer advanced user interfaces that allow vehicle occupants to interact with the system and access a wide range of external services.

3.2.2 Types of In-Car Operating Systems

The in-car operating systems used by major companies like Tesla, Waymo, and Cruise offer intuitive and customizable user interfaces that allow vehicle occupants to interact with the system easily and safely. Here's how the in-car operating systems of these companies handle the integration of external services:

a. **Tesla:**
Tesla has developed its own operating system called Tesla OS, which offers an intuitive and customizable user interface. Through the central touchscreen display in the vehicle, Tesla users can access various external services through dedicated integrations. For example, Tesla supports the integration of streaming services like Netflix and Spotify, allowing occupants to watch movies, listen to music, and enjoy other forms of entertainment during their journeys.

b. **Waymo**:
Waymo uses a Linux-based operating system to power its autonomous vehicles. Waymo's user interface is designed to provide clear and intuitive information about ongoing trips and available features. Currently, it does not integrate external entertainment services as it is still in testing, but being based on a Linux operating system, future implementation can be easily done.

c. **Cruise:**
Cruise uses a customized version of Linux as the operating system for its autonomous vehicles. Cruise's user interface is designed to offer a smooth and intuitive autonomous driving experience, providing clear information about the vehicle's status and journey. Currently, it is not known whether Cruise supports

the integration of external entertainment services, but as with Waymo, future implementation can be easily done.

3.2.3 Interesting Use Cases for Our Study

In addition to integrating entertainment services like Netflix and Spotify, in-car operating systems also offer the potential to integrate payment services. This opens the door to several interesting use cases, including:

- **Service Booking:** Vehicle occupants can book services such as hotels, restaurants, or event tickets directly through the in-car operating system's user interface. This offers greater convenience and flexibility during travels.
- **Delivery Services:** By integrating with delivery services, vehicle occupants can order food, groceries, or other products while on the go. The in-car operating system facilitates payment management and interaction with delivery services.

- **Car Sharing:** In-car operating systems can support integration with car-sharing services, allowing occupants to book and pay for shared vehicles directly through the user interface. This streamlines the booking and payment process for car-sharing services.

3.3 Payments

3.3.1 In-Car Payments

In-car payments enable vehicle occupants to conduct transactions directly within the vehicle. In this section, we will examine the state of the art of in-car payments, analyzing the current market, the payment networks used, and their respective advantages and disadvantages.

3.3.1.1 Market Benchmarking of In-Car Payments

In the market of in-car payments, several actors are emerging as leaders and pioneers. Examples of in-car payment solutions include:

a. **Apple CarPlay and Google Pay:**
Apple CarPlay and Google Pay are platforms that allow users to sync their mobile devices with the vehicle's infotainment system and make payments

through dedicated applications. These solutions enable occupants to pay for fuel, tolls, parking, and other transactions directly from the vehicle's display.

b. **FordPass and Ford Pay:**

Ford introduced FordPass, an app that enables users to manage various vehicle functionalities, including payments. Through Ford Pay, occupants of Ford vehicles can make in-car payments for services like refueling and parking.

c. **BMW ConnectedDrive:**
BMW offers the ConnectedDrive service, which integrates the vehicle's infotainment system with a range of online services, including payments. BMW vehicle occupants can use the ConnectedDrive platform to make in-car payments for fuel and other services.

3.3.1.2 Current Payment Networks Used: Advantages and Disadvantages

In in-car payments, various payment networks are used, each with specific advantages and disadvantages. Examples include:

1. **Traditional Payment Networks:** Many in-car payment solutions allow the use of traditional payment networks such as credit cards, debit cards, and digital wallets like Visa, Mastercard, Apple Pay, etc. These networks offer broad acceptance and user familiarity. However, they may require manual input of payment details and potential security risks.

2. **Near Field Communication (NFC):** NFC technology enables devices to communicate at short distances, facilitating contactless payments. This approach offers a quick and convenient payment experience without the need to manually enter payment details. However, it requires specific infrastructure and widespread adoption of compatible devices.

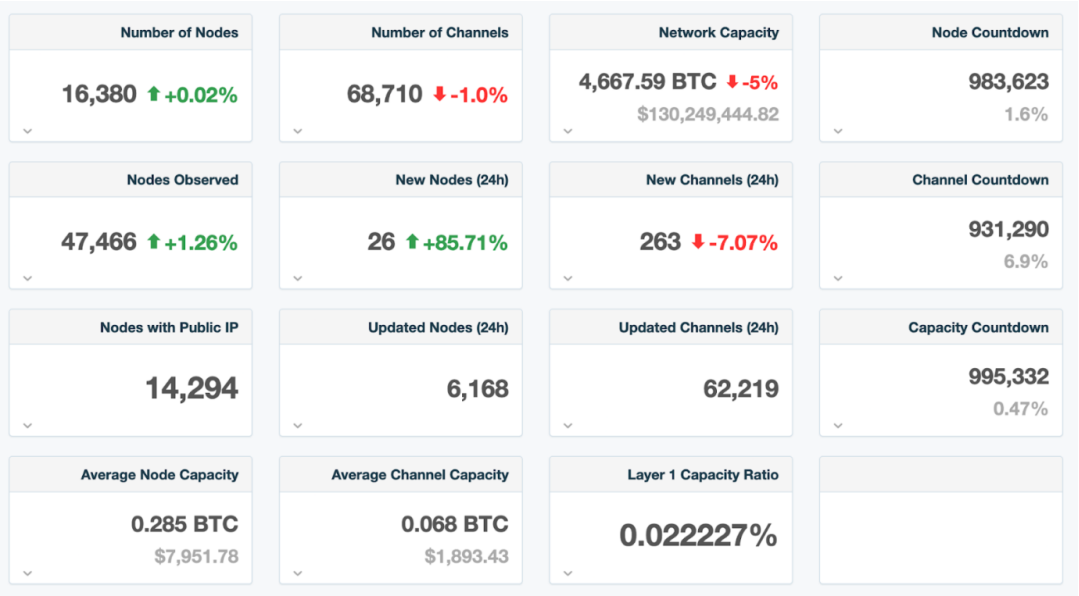
3. **Mobile Payment Services:** Mobile payment services like Apple Pay, Google Pay, and Samsung Pay offer a convenient solution for in-car payments. Users can utilize their mobile devices to make contactless payments using technologies like NFC or QR codes. These services offer ease of use and security through biometric authentication or PIN. However, dedicated applications and device compatibility may be required.

3.3.2 Lightning Payments

As described earlier, the Lightning Network is a second-layer scalability solution for Bitcoin aimed at reducing transaction costs and speeding up confirmation times.

Currently, the Lightning Network is experiencing a real boom. The number of transactions processed on the Lightning Network has reached impressive levels, with a consistent growth in the number of payments routed through the network. The number of nodes in the Lightning Network is continually expanding, with over 16,361 nodes currently. Similarly, the number of channels opened between Lightning Network nodes is rapidly increasing, creating an increasingly interconnected network with over 60,000 open channels and a liquidity of 4,865.30 BTC. This development is crucial to ensure greater liquidity and accessibility of payment channels for users, enabling them to send and receive payments quickly and cost-effectively on the Bitcoin network.

Figure 3.1: Lightning Network Statistics (*Source*)



Lightning payments permit the use of Bitcoin for small purchases without virtually paying any fees. This is highly significant because it allows small Bitcoin communities to emerge. Examples of this phenomenon include Bitcoin Beach in El Salvador and the Bitcoin Jungle in Costa Rica, where Bitcoin is utilized as a medium of exchange.

3.3.3 Machine-to-Machine (M2M) Payments

With the advancement of Internet of Things (IoT) technologies, there is a need to develop payment solutions that enable Machine-to-Machine (M2M) transactions to occur rapidly, scalably, and economically.

Conventional payment systems do not accommodate these types of payments for various reasons: primarily due to the programmability of payments, the costs associated with financial intermediaries and bureaucracy, the lack of payment irreversibility, and the timing of payment confirmations.

On the other hand, the Lightning Network offers several intriguing properties such as:

1. Scalability and Transaction Speed:
The Lightning Network can facilitate rapid M2M payments, allowing for a high number of transactions per second without congesting the main Bitcoin blockchain. Through off-chain payment channels and instantaneous transactions, the Lightning Network provides nearly instant confirmation times, enabling real-time M2M transactions.

2. Microtransactions:
A crucial aspect of M2M payments is the ability to conduct microtransactions, i.e., payments of very small amounts. The Lightning Network is particularly suited for managing this kind of transaction due to its off-chain structure. Transaction costs are negligible, enabling M2M payments even for minimal amounts. This paves the way for new business scenarios in the IoT sector, such as monetizing data generated by devices through real-time payments.

3. Automation of Payments:
The Lightning Network can be programmed to automate M2M payments through the use of smart contracts. Smart contracts allow for the definition of specific rules and conditions, enabling machines to autonomously make payments based on predefined events. For example, an IoT device could receive an automatic payment from another device when a particular activity is performed or a specific goal is achieved. This significantly simplifies the management of M2M payments, eliminating the need for human intervention in every single transaction.

Features	Banking system	Lightning Network
Amount of transaction	High	Low
Jurisdiction	Currency dependent	Cross border transaction, Geography agnostic
Payment setup	Human intervention needed	Programmable payments
Access to asset	Sequential	Simultaneous
Type of service rendered	Use of equipment	Access to data and energy
Transaction cost	Fixed fees up to few EUR	Virtually none (few sats for routing)
Transaction settlement	From 1d to several days for international transactions	Instantaneous
Privacy	Banking legacy systems	Anonymous: routed via TOR

Figure 3.2: Payment technology comparison (*Source*)

Here some examples of how these payment technologies can be used:

1. Micropayments for digital services usage:

The Lightning Network can facilitate real-time payment for utilizing digital services, such as accessing premium content, using applications, or accessing online platforms. For instance, an IoT device could automatically pay for cloud computing services based on actual consumption, eliminating the need for traditional contracts or billing.

2. Payments for shared physical resources usage:

An intriguing use case involves M2M payments for shared physical resources like vehicles, equipment, or devices. The Lightning Network enables the establishment of an autonomous payment system between devices, facilitating efficient sharing and real-time payment management based on actual usage.

3. Device-to-Device payments for connected services:

In the context of IoT, connected devices can offer complementary services to each other. For instance, an IoT device might provide data monitoring services to another device and receive automatic payment for the provided service in return. The Lightning Network streamlines real-time execution of such transactions without the need for intermediaries.

4. Microtransactions for Data and Information:

The Lightning Network enables M2M payments for purchasing or selling data and information. For example, an IoT device could pay another device for access to specific data, such as weather forecasts, market information, or sensor data. This type of transaction opens up new opportunities for monetizing data generated by IoT devices.

5. Payments for electricity and resource usage:

The Lightning Network can be employed to enable M2M payments for resource usage, including electricity. For instance, an IoT device requiring energy to operate can make instant payments for consumed energy in real-time, simplifying billing processes and promoting more efficient resource utilization.

3.4 Cryptographic Assets in the Automotive Market

3.4.1 Benchmarking Crypto Solutions in the Automotive Market

To benchmark crypto solutions in the automotive market, analyze the solutions being explored by BMW it’s enaugh. Other companies are also engaging in similar endeavors, but BMW seems to be a leader in this field. The following information is extracted from an article directly sourced from BMW's official website.

The article titled "Blockchain in the Automotive Sector: BMW's Innovation" on BMW's official site provides insights into the use cases offered by BMW in the realm of blockchain technology. Here's a summary of the mentioned use cases:

1. Supply Chain Transparency:
BMW utilizes blockchain technology to track and record information related to its vehicle supply chain. This ensures the origin and quality of components used in BMW vehicle production, contributing to the company's sustainability and social responsibility.

2. Secure and Smart Transactions:
BMW explores blockchain-based transactions' potential to create a smart mobility ecosystem. Through the use of smart contracts, transactions such as fuel payment, tolls, or vehicle rentals can be facilitated and automated, enhancing operational efficiency and security.

3. Decentralized Digital Identity:
BMW is investigating the concept of decentralized digital identities for vehicles. Through blockchain, secure digital identities for vehicles can be established, allowing safe authentication and authorization for accessing specific services and functionalities.

4. Microtransactions and Shared Services:
BMW is also experimenting with blockchain-based microtransactions to enable immediate and secure payments for shared services like car sharing. This approach simplifies payment processes for customers and reduces administrative complexity for service providers.

These use cases are also areas of focus for companies like Ford, Nissan, Toyota, Mercedes, Stellantis, and many others.

3.4.2 Focus on NFT Solutions

For years, automakers have been experimenting with NFT technology for various purposes. Here's an overview extracted from a CoinTelegraph article.

The article titled "Automakers Are Minting NFTs, But Is There a Strong Use Case?" explores real-world use cases of NFTs implemented by automakers. Here's a summary of the mentioned use cases:

1. Alfa Romeo and Vehicle NFT:
Italian automaker Alfa Romeo created a unique NFT for one of its luxury vehicles. This NFT allows owners to access exclusive content such as high-resolution videos and photos of the vehicle, along with providing a digital certificate of authenticity.

2. Mercedes-Benz and Automotive Art NFTs:
Mercedes-Benz launched a series of NFTs representing artworks inspired by its automotive history. These NFTs offer collectors the chance to own a digital part of Mercedes-Benz's automotive heritage.

3. BMW and Vehicle Customization NFTs:
BMW experimented with NFTs to allow customers to personalize their vehicles with unique digital content. This offers exclusivity and allows owners to express their identity through digital design and customization.

4. Ford and Vehicle Ownership NFTs:
Ford introduced a pilot project where vehicles are associated with an NFT that serves as proof of ownership and authenticity. This could simplify resale processes and reduce the risk of counterfeiting in used vehicle purchases.
It's important to note that the article also highlights the debate over the value and utility of NFTs in the automotive sector, emphasizing that a strong definitive use case for this technology in the industry has yet to be established.

3.4.3 Focus on Decentralized Identity Solutions for Vehicles

Several experiments are ongoing in the field of decentralized identities for vehicles. Here are some examples:

LG CNS is working on creating a decentralized identity system for autonomous vehicles. This system would enable vehicles to authenticate and securely communicate with other entities, such as other vehicles or road infrastructure, without relying on centralized intermediaries. This could enhance the security

and efficiency of communications between autonomous vehicles and their surrounding environment.

As mentioned in the previous chapter, BMW is also working on a decentralized identity protocol for its vehicles. Ford is doing the same, with the goal of supporting the second-hand car market.

In general, all companies working on this technology are converging on a standard provided by the World Wide Web Consortium (W3C) called Decentralized Identifier (DID).

The DID standard defines a framework for creating decentralized digital identities. A DID is a unique, persistent, and global identifier assigned to a subject, such as an individual, organization, or device. This allows subjects to own and control their digital identities without relying on centralized entities or intermediaries. The W3C DID standard offers a set of principles and technical specifications to ensure the security, privacy, and interoperability of decentralized digital identities. The use of DIDs enables autonomous management of digital identities, secure authentication, selective sharing of personal information, and the creation of reliable and scalable digital ecosystems.

Goal	Description
Decentralization	Eliminate the requirement for centralized authorities or single point failure in identifier management, including the registration of globally unique identifiers, public verification keys, services, and other information.
Control	Give entities, both human and non-human, the power to directly control their digital identifiers without the need to rely on external authorities.
Privacy	Enable entities to control the privacy of their information, including minimal, selective, and progressive disclosure of attributes or other data.
Security	Enable sufficient security for requesting parties to depend on DID documents for their required level of assurance.
Proof-based	Enable DID controllers to provide cryptographic proof when interacting with other entities.
Discoverability	Make it possible for entities to discover DIDs for other entities, to learn more about or interact with those entities.
Interoperability	Use interoperable standards so DID infrastructure can make use of existing tools and software libraries designed for interoperability.
Portability	Be system- and network-independent and enable entities to use their digital identifiers with any system that supports DIDs and DID methods.
Simplicity	Favor a reduced set of simple features to make the technology easier to understand, implement, and deploy.
Extensibility	Where possible, enable extensibility provided it does not greatly hinder interoperability, portability, or simplicity.

Figure 3.3: DID design (*Source*)

Chapter 4

Solution Design

4.1 Introduction to Solution Design

In this chapter, we delve into the heart of the thesis project and present the design conceived for prototyping a Bitcoin wallet within an autonomous driving car, named "Car-Wallet." This wallet will serve as software integrated into the car's operating system, enabling the vehicle to interact with the Lightning Network and the Bitcoin network. Furthermore, it will be capable of managing RGB assets, thus offering a wide range of functionalities.

4.1.1 Objectives

The primary goal of this project is to realize an innovative and secure wallet that allows the car to conduct financial transactions and interact efficiently and reliably with the Bitcoin network. The specific objectives of the Car-Wallet are as follows:

1. Integration with the Lightning Network:
The Car-Wallet must support the Lightning Network, which facilitates instant and scalable micro-payment transactions. The car will be capable of conducting peer-to-peer transactions on the Lightning network, enhancing payment efficiency and achieving full autonomy. It will have no reliance on any kind of financial intermediaries, ensuring global and borderless operability.

2. Integration with the Bitcoin Network:

The wallet must connect to the Bitcoin network, enabling the car to execute transactions and verify the status of past transactions. This will enable the car to open and close Lightning channels, make payments, receive funds via the Bitcoin network, and manage RGB assets through Bitcoin's UTXOs.

3. RGB Asset Management:
The Car-Wallet must support the management of RGB assets, allowing the creation and management of cryptographic contracts on the Bitcoin blockchain. This will enable the car to represent and exchange digital assets within the system.

4. Non-Custodial Property:
To ensure maximum security, the Car-Wallet must be "non-custodial" in nature. This means that the private keys used to access the car's funds will be protected and exclusively owned by the car owner. This approach reduces the risk of fraud or fund theft, allowing the user to retain full control over their financial assets.

5. Security:
Since the wallet will be connected to the internet, implementing best security practices will be necessary to safeguard the car from potential external attacks. Advanced security measures will be adopted to prevent privacy breaches, unauthorized access, and financial information theft.

4.1.2 Reading Guide

This chapter provides a detailed overview of the Car-Wallet's solution design. After a brief introduction presenting the idea in general terms, the project's design details will be presented. In the same section, an example of what the user interface could be, via a wireframe, will be proposed.

The "Solution Description" section will explain in detail how the required functions are intended to be implemented. Specifically, it will address key management, interaction with the Lightning Network, and RGB contract management.

Subsequently, automatic machine-to-machine (M2M) payments with the Lightning Network will be examined, including descriptions of the implemented smart contracts for automatic payments and an explanation of the HTTP 402 "Request for Payment" error implementation. A comprehensive discussion of the advantages and challenges of implementing automatic M2M payments using the Lightning Network will follow.

The implementation of certifications for public keys through the internet's Public Key Infrastructure (PKI) will be discussed. How does PKI work, and why is ensuring non-repudiation important in our use case?

Lastly, a concrete example of an ownership contract, focusing on the car's passport, will be presented. A description of the contract's design, explaining the process of creating and maintaining the contract, will be provided. Finally, security and the integrity of cryptographic contracts will be discussed.

4.2 Model Design

4.2.1 Project Design Description

The aim of this software is to provide a comprehensive solution for managing Bitcoin, Lightning Network, and RGB transactions, allowing the user to perform the following set of operations:

RGB Smart Contract Creation, Signing, Storage, and Modification:
The Car-Wallet must offer advanced functionalities for managing RGB smart contracts. Smart contracts enable users to create autonomous and customized contracts for managing digital assets. The wallet must facilitate the creation of new contracts, their digital signing, secure contract storage within the wallet, and provide the ability to modify them in accordance with specified rules and conditions.

Manual Lightning Network Invoice Generation:
The car wallet should offer the ability to manually generate invoices for the Lightning Network. These invoices represent payment requests that can be sent to other network participants to receive Bitcoin payments quickly and reliably.

Lightning Network Invoice Payment:
The Car-Wallet should enable the user to make payments via Lightning Network invoices. The user will receive invoices from other network participants and, using the car wallet, will be able to authorize and send payments corresponding to those invoices securely and immediately.

Creation and Configuration of Automatic Payments for Third-Party Services:
The Car-Wallet must support the creation and configuration of automatic payments for third-party services. These payments can be set up in either a pay-as-you-go mode, based on actual usage, or in a subscription mode, where payment occurs periodically for continuous service. The wallet must allow the user to specify desired payment modes, set spending limits, deadlines, and other

configuration options to adapt to the user's and third-party service's specific needs.

The Car-Wallet's design must efficiently and effectively integrate these functionalities while ensuring a high level of security and reliability. Advanced encryption practices will be employed to protect private keys and transactions, and communication protocols will be designed to ensure the confidentiality and integrity of information exchanged with the Bitcoin and Lightning Network. Security measures will also be implemented to safeguard the wallet against unauthorized access and prevent fraud or cyberattacks.

4.3 Solution Description

4.3.1 Key Management

For our Car-Wallet, it is crucial to adopt a solution that guarantees secure and efficient management of cryptographic keys. To achieve this, the decision has been made to use the infrastructure based on Bitcoin Improvement Proposals (BIP) 32 and BIP 39, which provide a solid foundation for creating and managing keys within the context of Bitcoin wallets.

The primary goal of BIP 32 is to enhance the security and usability of Bitcoin

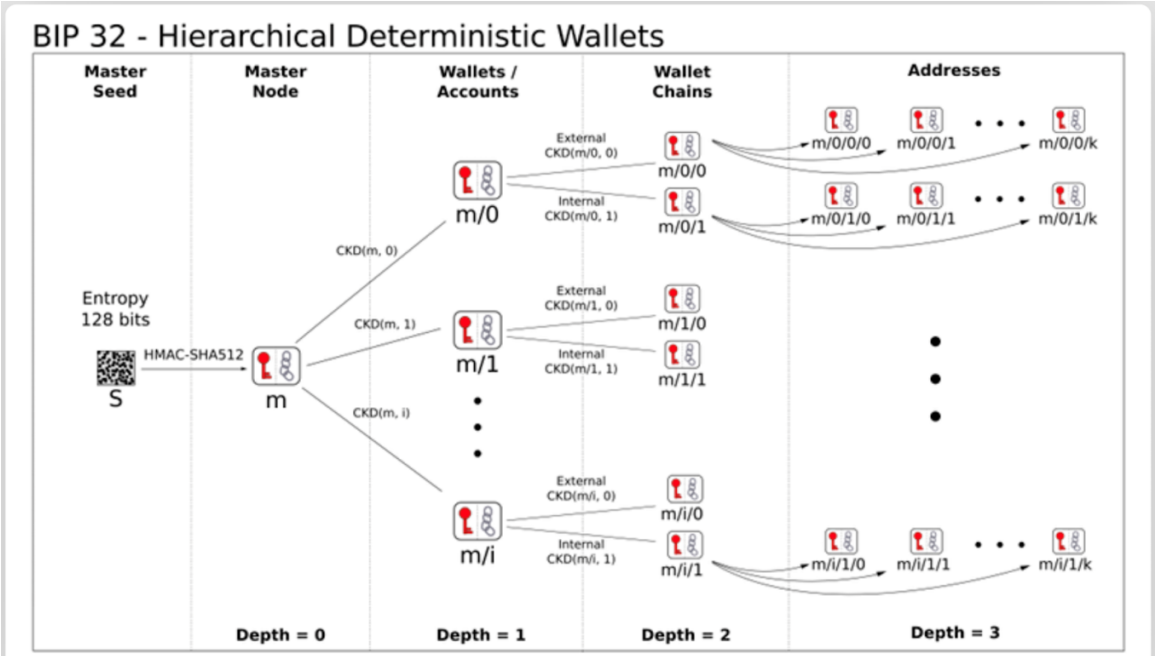


Figure 4.1: BIP32 Hierarchical Deterministic Wallets (*Source*)

wallets. Thanks to the hierarchical structure of keys, it's possible to create unique Bitcoin addresses for each transaction, thus providing increased privacy. Moreover, the use of BIP 32 simplifies wallet backup and restoration, as it is sufficient to store the seed to recover the entire key structure.
The key technical features of BIP 32 include deterministic key generation, the ability to create sub-trees for specific purposes, such as organizing funds into different sub-wallets, and the capability to derive public keys without knowledge of the corresponding private keys.
Bitcoin Improvement Proposal (BIP) 39
BIP 39, titled "Mnemonic code for generating deterministic keys," provides a standardized method for representing the seed of a Bitcoin wallet using a mnemonic phrase easily memorizable by the user.

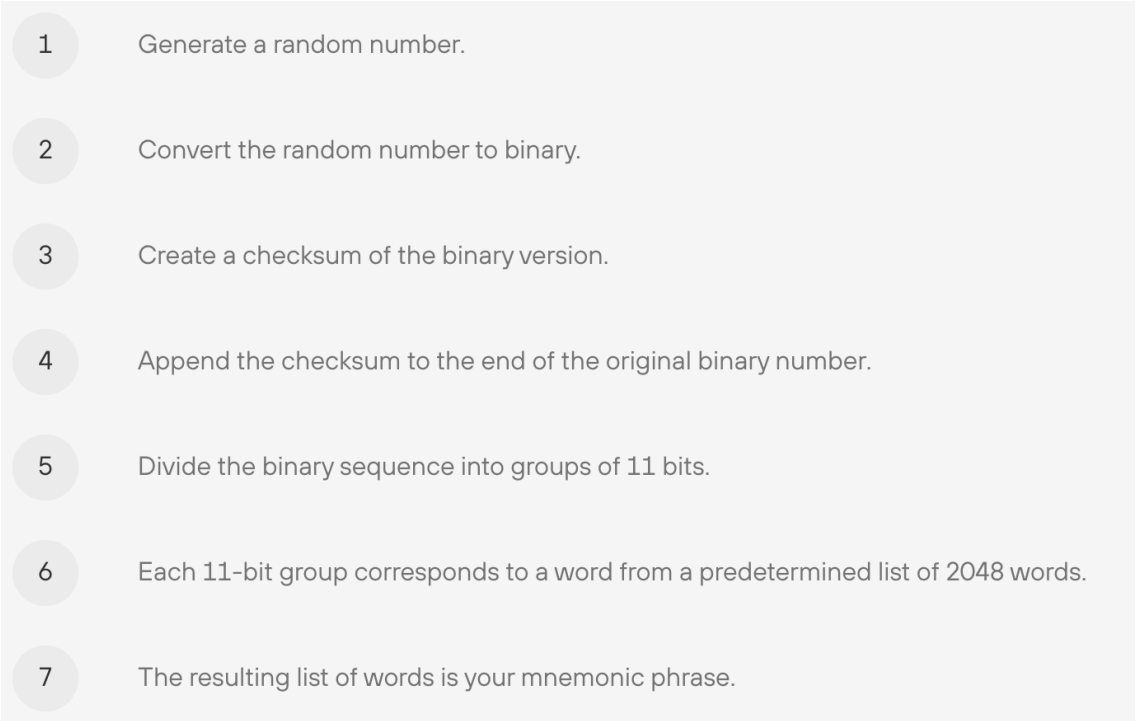


Figura 4.2: BIP 39 Seed Generation Procedure (*Source*)

The primary goal of BIP 39 is to simplify the process of backup and restoration of keys within a Bitcoin wallet. By utilizing a mnemonic phrase of 12 or 24 words, the user can securely store their seed without resorting to complex character sequences or lengthy private keys. Seed generation through BIP 39 is highly resistant to typing errors and provides enhanced user-friendliness.

There are multiple benefits arising from the utilization of BIP 32 and BIP 39 for key management in the Car-Wallet. Firstly, thanks to the deterministic hierarchical structure, it becomes feasible to easily create and extend sub-wallets for separately managing Bitcoin UTXOs, lightning funds, and RGB assets within the wallet. This modularity enables efficient organization of various fund types, streamlining the process of transaction management and tracking.

Furthermore, the adoption of BIP 32 and BIP 39 ensures high security within the Car-Wallet. Deterministic key generation results in a structured and predictable framework, simplifying the process of key backup and restoration. The mnemonic phrase from BIP 39 provides a secure method of storing the seed, minimizing the risk of key loss or misplacement.

4.3.2 Lightning Network Wallet Design

In the context of the Car-Wallet, the primary goal of implementing the Lightning Network wallet is to maintain a fully trustless and non-custodial approach. To achieve this objective, a strategy is adopted that combines the use of the Breez SDK library and the implementation of the Greenlight server for managing Lightning nodes.

The implementation of the Breez SDK simplifies interaction with a Lightning Service Provider (LSP) through an intuitive set of APIs. This enables the Car-Wallet to leverage the capabilities of the Lightning Network without directly tackling the technical complexities associated with managing a Lightning node. Furthermore, the use of the Breez SDK abstracts the intricacies of the Lightning wallet, allowing the Car-Wallet to benefit from Lightning Network features without dealing with the complexities of native development.

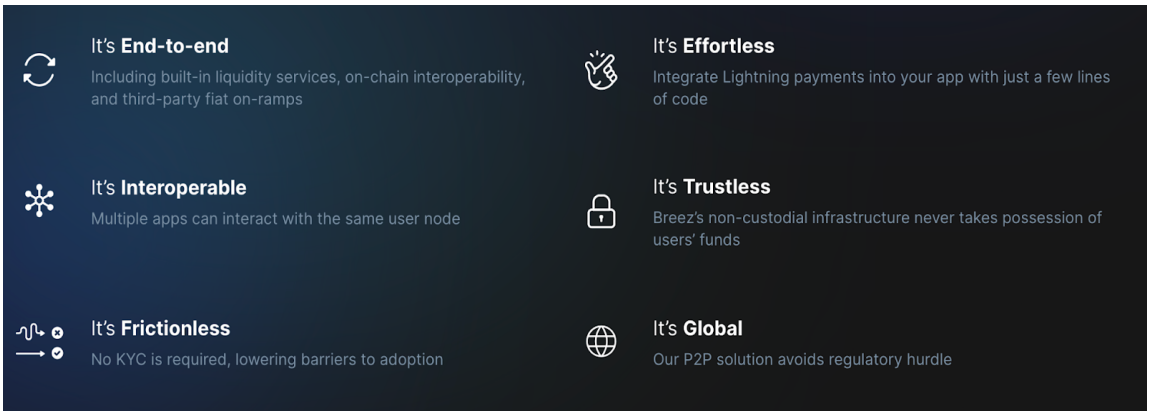


Figure 4.3: Breez SDK advntages *(Source)*

On the other hand, the implementation of the Greenlight server handles the management of Lightning nodes in a centralized manner. Greenlight offers a reliable infrastructure that enables the Car-Wallet to utilize an on-demand node while still maintaining full key custody by the user. This way, Greenlight ensures user privacy and prevents third parties from accessing the private keys within the Car-Wallet.

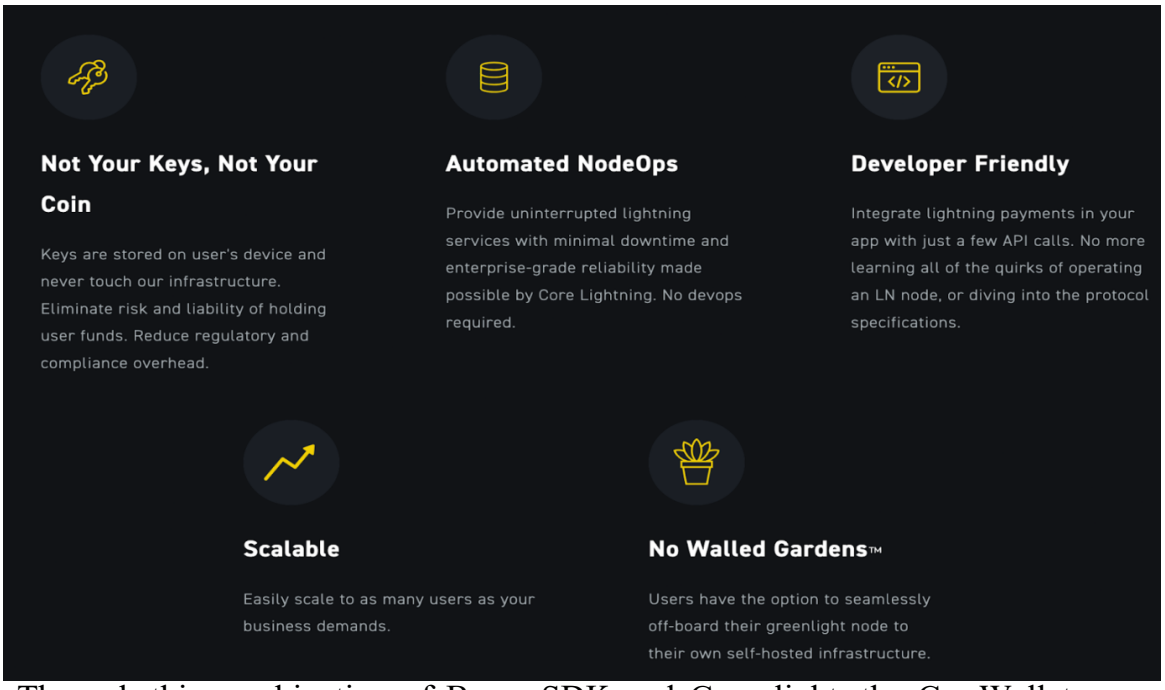


Figure 4.4: Greenlight Advantages (*Source*)

Through this combination of Breez SDK and Greenlight, the Car-Wallet can benefit from the advantages of the Lightning Network without compromising non-custodial control. The user maintains full control over their private keys, while the complexity of Lightning node management is taken care of by Greenlight, providing a streamlined and secure experience for using instant payments within the Car-Wallet.

Additionally, the importance of offering more experienced users the option to connect their own Lightning node to the Car-Wallet is recognized. This will allow users to leverage their own resources and customized configurations within the Car-Wallet environment. To this end, support for integration with services like Zeus and other similar solutions that may develop over time is anticipated.

4.3.3 RGB Wallet Design

In the context of the Car-Wallet, the implementation of an RGB wallet is envisaged to enable the car owner to manage information related to RGB contracts in a simple and secure manner. It is important to note that the RGB protocol is currently under development, and therefore, the design proposal presented here may undergo changes in the near future. However, at present, interacting with RGB contracts appears to be quite straightforward.

As previously mentioned, the entire logic of RGB contracts is managed on the client side, which means that all relevant information for the car owner can be stored locally without the need to directly interact with the network, unless specific operations on these contracts are desired. In that case, the Car-Wallet will connect to the Bitcoin network to communicate the desired changes.

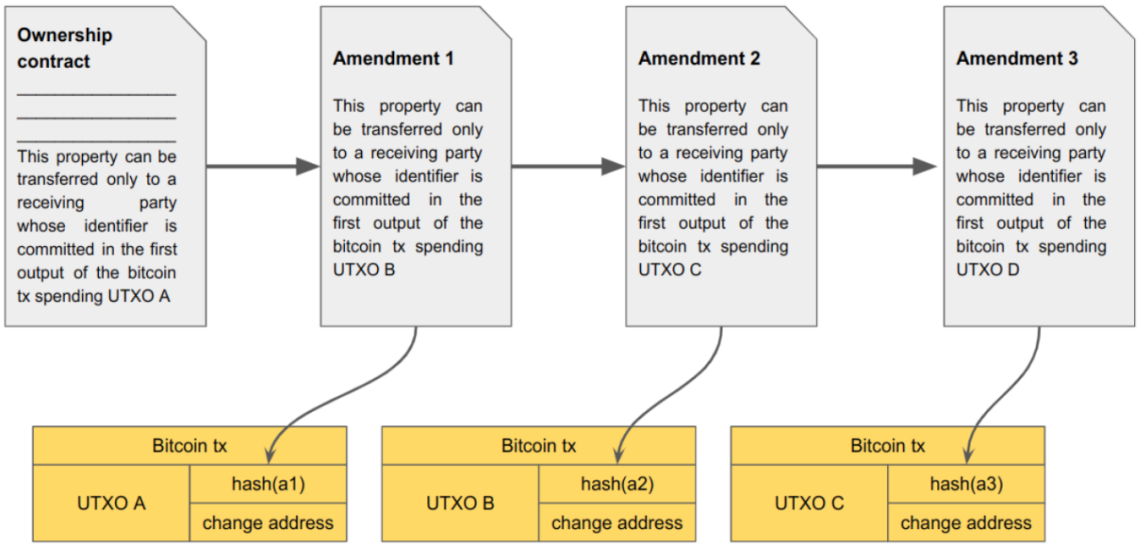


Figure 4.5: RGB Contract Transfer Chain (*Source*)

This implementation guarantees once again a fully trustless, privacy-oriented approach without intermediaries. The keys required to access and manage RGB contracts remain firmly in the possession of the car owner, and the contracts themselves are stored within the car's memory. This ensures that only the owner has complete control over their information and that operations on the contracts are performed securely and privately.

It is important to highlight that the implementation of the RGB wallet in the Car-Wallet provides the car owner with a convenient way to manage their contracts without depending on third parties or intermediaries. The entire process is based on trust in their private keys and the ability to communicate the necessary changes to the Bitcoin network when required.

It is emphasized that, given the evolving nature of the RGB protocol, it is essential to monitor future developments and accordingly adapt the implementation in the Car-Wallet to fully leverage the potential offered by this emerging protocol.

4.4 Automatic M2M Payments with Lightning Network

4.4.1 Automatic Payments on Bitcoin

The development of smart contracts to automate payments on Bitcoin and, specifically, through the Lightning Network (LN), is still in its early stages. However, some emerging solutions are laying the groundwork to make machine-to-machine (M2M) payments automation possible.

One possible solution could be the use of Bitcoin's on-chain scripting language. Although this language is relatively simple, it already allows for the creation of basic smart contracts, with Lightning Network channels being a concrete example. Nevertheless, implementing M2M solutions directly using Bitcoin's scripting language seems to be quite challenging.

However, to achieve a higher level of automation and to make M2M payment management fully programmable, solutions are starting to be explored that operate outside of the Bitcoin and Lightning Network.

A concrete example of a protocol that allows for M2M payment automation is "L402". This protocol, developed by Lightning Labs, uses the well-known HyperText Transfer Protocol (HTTP) for generating and automatically paying Lightning Network invoices. Through the use of "L402", machines can interact with the Lightning Network, send and receive LN payments without the need for human supervision. This opens the way to a new scenario in which IoT devices and services can manage their payments autonomously, without requiring external interventions, while ensuring greater efficiency and security in M2M transactions.

The creation of smart contracts and protocols like "L402" is just the beginning of an evolution towards automated M2M payments on the Lightning Network.

4.4.2 L402

Lightning API keys (L402) leverage the capabilities of Macaroons and Lightning Network features to create a mechanism that allows distributed systems to authenticate a user and a payment receipt without requiring access to a central user or invoice database.

A macaroon is an advanced authentication mechanism for distributed systems. It works like a cookie but can be cryptographically verified, enabling verification without a central database. It contains permissions and can be used for secure authorization delegation.

An L402 consists of a Macaroon along with a Lightning Network payment preimage. The Macaroon is transmitted to the user via HTTP along with a Lightning invoice and contains the invoice's payment hash as a condition.

To be a valid L402, the user must present two pieces of information:
1. The partial L402, which is the Macaroon including the payment hash.
2. The preimage, which can be obtained by fully paying the Lightning invoice.

Since the payment is a hash of the preimage and the preimage can only be obtained by fully paying the Lightning invoice, anyone with the root key that the Macaroon was created with can easily verify:
1. That the L402 was issued by the appropriate authority.
2. That the L402 contains the relevant capabilities.
3. That the Lightning invoice has been paid.

The L402 specification lists the necessary characteristics for a valid L402, including the version, user identifier, payment hash, and caveats that may include permissions for specific services, service capacities, and service constraints.

Verifying an L402 requires the root key with which the original Macaroon was created. This allows the server to verify, line by line, clause by clause, that the Macaroon was issued by the appropriate authority and that each clause has been properly modified.

Finally, the preimage is verified against the payment hash to ensure that all pending invoices have been paid.

Here's an example of an L402 Payment request:

```
HTTP/1.1 402 Payment Required

Date: Mon, 04 Feb 2014 16:50:53 GMT

WWW-Authenticate: L402
```

```
macaroon="AGIAJEemVQUTEyNCR0exk7ek90Cg==",
invoice="lnbc1500n1pw5kjhmpp5fu6xhthlt2vucmzkx6c7wtlh2r625r30cyjsfqhu8rsx4xpz5lwqdpa2
fjkzep6yptksct5yp5hxgrrv96hx6twvusycn3qv9jx7ur5d9hkugr5dusx6cqzpgxqr23s79ruapxc4j5usk
t4htly2salw4drq979d7rcela9wz02elhypmdzmzlnxuknpgfyfm86pntt8vvkvffma5qc9n50h4mvqhng
adqy3ngqjcym5a"
```

In this way, it is possible to implement scheduled and automated payments within web services in a completely trustless manner. L402s thus provide a fundamental basis for creating consumption APIs for the machine-to-machine economy.

4.4.3 Example of Usage for Our Use Case

In our case, to fully leverage the potential of L402, the implementation of the reverse proxy named "Aperture" within the car-wallet represents a significant step forward. This proxy allows us to create a secure solution for easy management of Machine-to-Machine (M2M) automatic payments.
The role of Aperture within the car-wallet is multifunctional and strategically crucial. Acting as a reverse proxy, Aperture serves as a gateway to the so-called "Lightning-Native Web," facilitating communication between the autonomous vehicle and a wide range of Lightning Network-based payment services. Aperture's ability to handle both gRPC requests through HTTP/2 and REST via HTTP/1 and HTTP/2 translates into a flexible and adaptable environment, allowing autonomous vehicles to take advantage of modern communication protocols.
Specifically, Aperture plays a key role in authentication and payment within the context of our car-wallet. Upon receiving incoming connections, Aperture verifies the authenticity of the L402 protocol, thus authorizing the autonomous vehicle to access the desired payment services. If the forwarded request involves a transaction, Aperture generates a Macaroon, a type of authorized access token, and combines it with a Lightning invoice. This combination is sent to the autonomous vehicle along with an HTTP 402 "Payment Required" status code, indicating the obligation to make the payment for access to the requested services. This approach synergistically combines authentication and payment, creating a seamless and secure experience for the autonomous vehicle without requiring a central third party for the transaction process.
The implementation of Aperture within the car-wallet marks a crucial step forward in realizing agile and secure automatic payments for autonomous vehicles. This innovative approach fully utilizes the L402 protocol, allowing vehicles to interact directly with the Lightning Network, managing real-time microtransactions, and greatly simplifying the payment process. The integration of Aperture represents a milestone in the Internet of Things (IoT) sector, paving the way for new interaction scenarios between autonomous vehicles and payment services, with significant implications for operational efficiency and transaction security.

4.3.4 Certification of Public Keys

4.5.1 Introduction to the Internet PKI Certification Authority (CA)

CAs are trusted and recognized entities that issue digital certificates for public keys used in cryptographic communication protocols. Entities seeking certification for their public keys must undergo a rigorous authentication and validation process by CAs to obtain a digital certificate. The CAs themselves must meet security and reliability criteria to become globally recognized and accepted.
The hierarchical structure of CAs involves the existence of a root CA (Root CA), representing the main authority in the system. The root CA issues certificates for other higher-level CAs, called Intermediate CAs, which can in turn issue certificates for further lower-level CAs or issuers directly. This CA hierarchy creates a chain of trust, where certificates issued by a CA are digitally signed by the higher-level CA, thus ensuring the authenticity and integrity of the certified public keys.

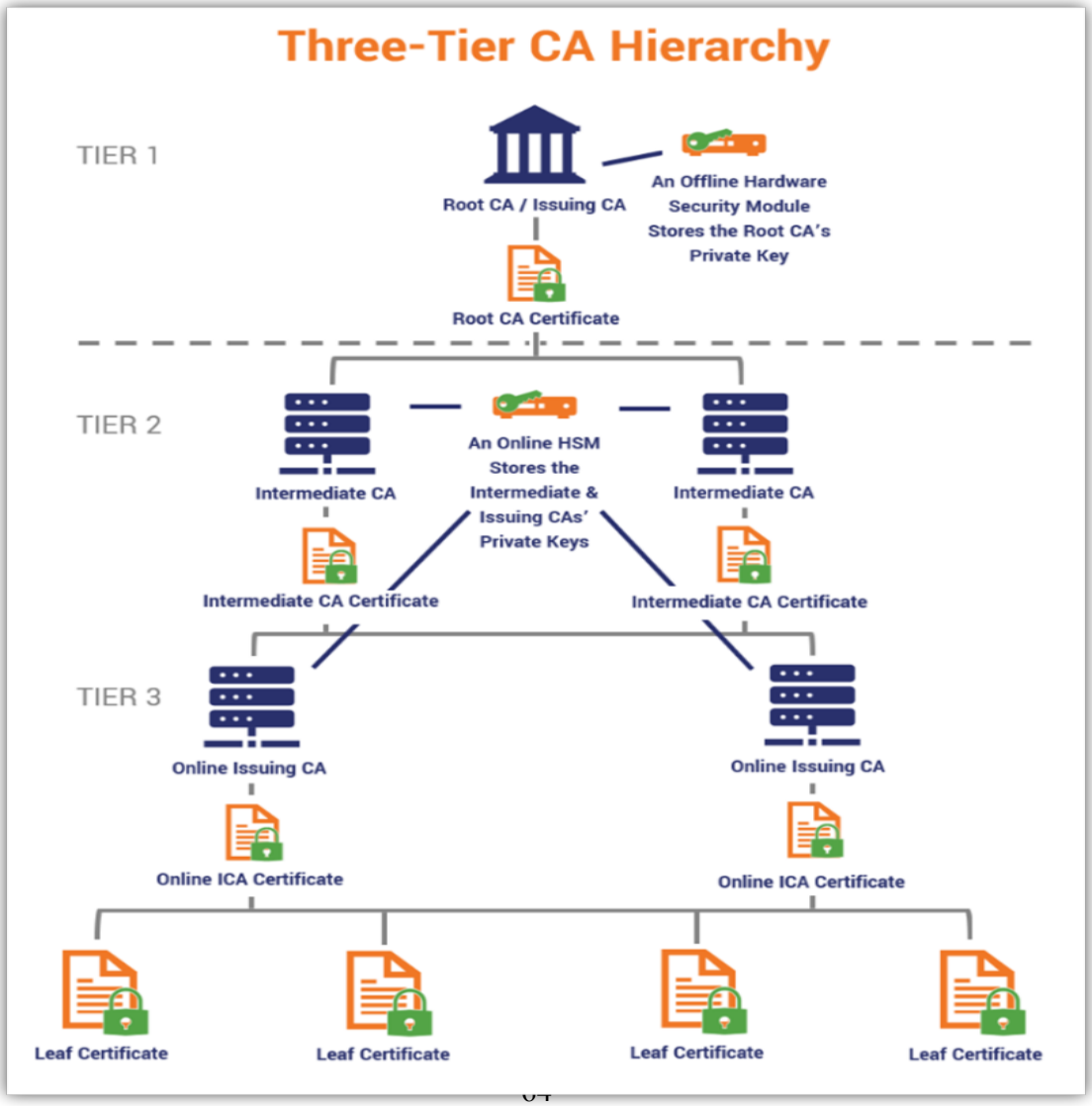


Figure 4.6: Public Key Infrastructure Example (*Source*)

An example of the use of certificates issued by CAs is in HTTPS connections. When a client connects to a website using HTTPS, the server sends its digital certificate, which contains its public key signed by the CA that issued it. The client can then verify the authenticity of the certificate by comparing the digital signature with the root CA's public key or a trusted CA's key previously stored in its system. If the verification is successful, the client can establish a secure connection with the server, ensuring that the exchanged information is protected and authenticated.

The authentication provided by certificates issued by CAs is of vital importance for the security of communication protocols. Without proper authentication, communications could be vulnerable to man-in-the-middle attacks, where an attacker positions themselves between the two parties and intercepts or modifies the exchanged information. The use of digital certificates and CAs ensures that the involved parties can verify each other's identity and establish a secure communication.

The most important property provided by this CA structure is non-repudiation. In essence, this property ensures that the owner of a certificate cannot deny any activities carried out through that certificate, even before a judge. This is possible due to the certificate revocation process, where the certificate holder must report the loss of the keys if it occurs. Therefore, if this request hasn't been made, the only one capable of using that certificate can be the legitimate holder.

4.5.2 Discussion on the Property of Non-Repudiation and the Importance of Public Key Certification

The property of non-repudiation represents a fundamental pillar in the car-wallet ecosystem where various actors interact to create, share, and authenticate digital contracts. This property assumes a central role as it confers reliability and integrity to contracts. As outlined earlier, a crucial aspect of the car wallet is the car owner's ability to establish contracts with third parties and store them within the vehicle's RGB archive in a trustless and secure manner.
A practical demonstration of this concept can be found in a use case where a ownership contract is established between the car manufacturer and its buyer. In this scenario, the attribute of non-repudiation plays a prominent role. To ensure that the ownership contract remains untouched and immune to future disputes, the car manufacturer should obtain a digital certificate from a reputable Certificate Authority (CA). Obtaining this certificate constitutes an official recognition of the manufacturer's identity by a reliable source, solidifying their position as the legitimate creator of the car.

The importance of this dynamic becomes evident when considering the transfer of car ownership from one owner to another. When the car is subsequently sold, the associated ownership contract cannot be contested. The confirmed identity of the manufacturer through the digital certificate lends unquestionable trust to the validity of the contract, laying the foundation for clear transaction traceability.

The use of digital certificates not only ties the manufacturer's identity but also their responsibility regarding the contracts generated and stored in the system. This structure creates a secure environment where contracting parties can interact without fear of legal disputes or fraudulent identities.

It's important to note that in this case, it's not essential for the car owner to have a certificate; in fact, it would be a dangerous violation of their privacy. The owner should be free to not disclose their identity and establish contracts with trusted parties, just as in TLS connections, where the server needs to identify itself through a certificate while client authentication is optional.

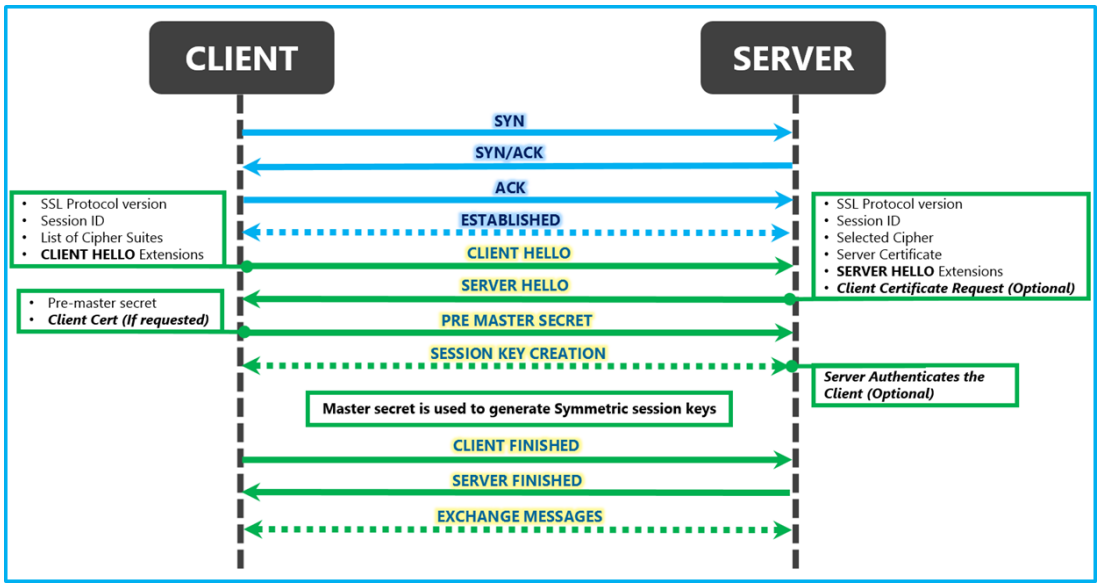


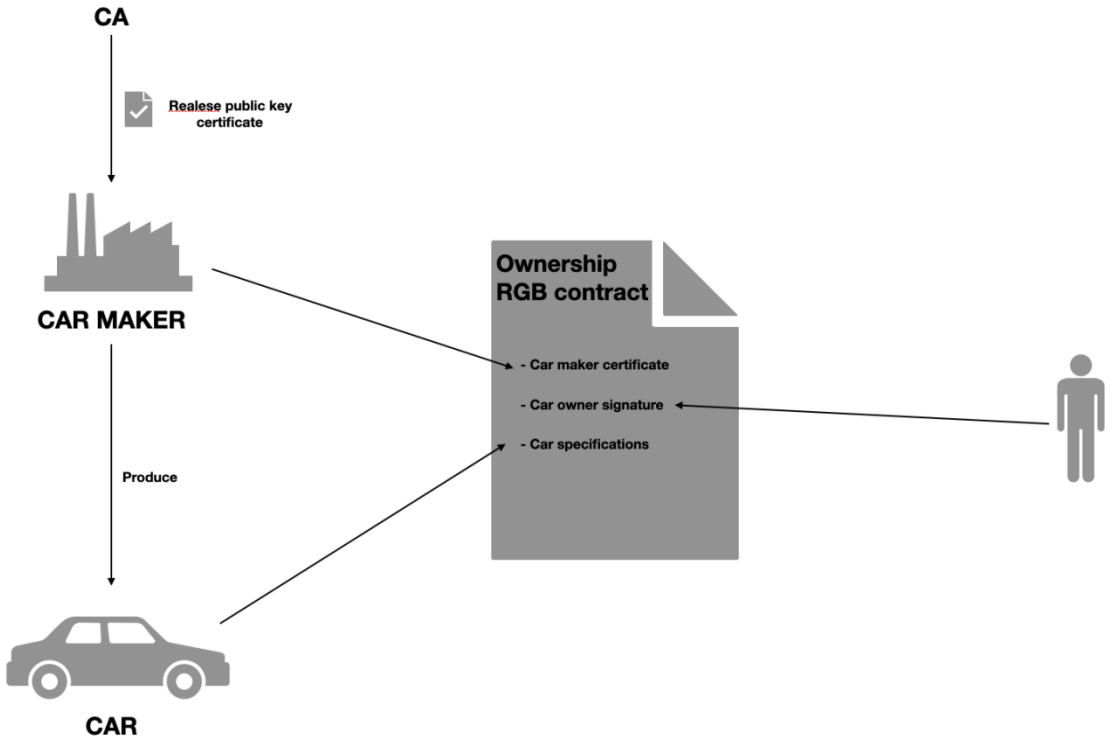
Figura 4.7: TLS Handshake Protocol (*Source*)

In summary, the property of non-repudiation and the careful implementation of digital certificates represent a crucial component in the architecture of the car-wallet. This feature guarantees the solidity of transactions and contracts within the system, ensuring the accuracy and reliability of digital interactions. The segregation of certification functions to the parties involved, in line with the TLS connection model, is essential to preserve the confidentiality of the end user while establishing robust foundations for contract management in the context of the car-wallet.

4.3.5 Use Case of Ownership Contracts: Car Passport

4.6.1 Contract Design

Here, we intend to present the design of an RGB contract that defines ownership rights over the car and declares its identity. This contract acts as a real "car passport," tracking and authenticating the ownership and identity of the



automobile.

Initially, the contract is created and signed by the Car Maker, who inserts all relevant information about the car components. In this initial phase, the Car Maker is recognized as the contract creator and the initial car owner. The contract contains detailed data about the car, such as the model, chassis number, technical specifications, and any other information relevant to uniquely identify the vehicle.

When a buyer wishes to purchase the car, they sign this contract, thereby becoming the new owner of the car.

The ownership contract work as a "Car-Wallet passport," tracing the history of car ownership and authenticating subsequent ownership changes. When the car needs to change ownership, the contract is signed both by the current owner and the new buyer. This digital signature ensures the validity of the transfer, ensuring that all transactions are authentic and secure.

This contract is defined using the LNPBP-0025 standard, enabling the creation of non-fungible tokens exchangeable through the RGB protocol. This way, ownership transfers are securely and immutably recorded on the Bitcoin blockchain through the RGB system.

In the next chapter, we will delve into the implementation details of this contract within the RGB environment of our Car-Wallet. We will use the proof

Figure 4.8: Car Ownership RGB contract design

of concept to demonstrate how the car ownership contract can be created, signed, and securely managed using the LNPBP-0025 standard.

4.6.2 Possible Uses of This Contract

The ownership contract presented in the previous chapter offers numerous opportunities for use, thanks to its advanced properties in managing car ownership and component traceability.

Due to its structure and detailed car information management, the contract allows precise monitoring of all parts and technical characteristics of the vehicle, providing a reliable foundation for detecting any unauthorized modifications or fraud in the car's components.

An additional and interesting use of the ownership contract is its potential integration as an attachment in other contracts, allowing the car's identity to be linked to various operations. For instance, a second use case could involve creating a contract that stores and traces the entire maintenance history of the car. Since the car will undergo routine and extraordinary maintenance over time after its purchase, such a contract could prove useful.

This new contract, linked to the ownership contract, would serve to track all maintenance operations. A possible scenario involves each authorized service center receiving a certificate issued directly by the Car Maker, and all maintenance operations on the car being recorded as updates to this contract, signed by various dealers. Here, the non-repudiation property offered by the certification of public keys of the service centers would ensure the authenticity of the recorded information.

Thus, the car ownership contract would not only contain the history of owners but also serve as a detailed record of all maintenance performed on the automobile. This would provide a comprehensive and reliable overview of the car's maintenance history, offering significant benefits for future buyers and more efficient post-sale service management.

Chapter 5

Proof of Concept

5.1 Implementation Specifications

5.1.1 Language

Before proceeding with the proof of concept of the Car-Wallet Bitcoin, I conducted an analysis of programming languages to evaluate the most suitable choice. Two main options were considered: Python and Rust.

Python is a language known for its simplicity and ease of learning. It offers a wide range of libraries and tools that facilitate rapid application development. Since I'm already familiar with Python, it allows me to implement the required functions more swiftly. However, it's important to recognize that Python might not be the optimal choice.

On the other hand, Rust is becoming increasingly popular in the Bitcoin world and offers significant advantages in terms of security and performance. Thanks to its strict memory management system and prevention of common errors, Rust is considered a reliable and robust language for Bitcoin application development.

After weighing the pros and cons of both languages, I decided to use Python for the demo portion I will be implementing. My choice was motivated by my familiarity with Python and the ability to rapidly develop the required functions. However, I fully acknowledge the security and performance benefits of Rust

and I’am confident that a future complete implementation of this wallet will be done in Rust.

5.1.2 Architecture

The Car-Wallet is designed to offer a convenient and secure user interface directly on the car's monitor, allowing users to manage Lightning Network transactions and RGB contracts. The Car-Wallet's architecture consists of three main areas, each responsible for specific system functionality.

5.2 Key Management

5.2.1 Seed Generation

For key management of a Bitcoin wallet, the most widely used standard is BIP32, or "Hierarchical Deterministic Wallets."

This standard is extensively used for managing Bitcoin wallet keys. It allows hierarchical generation of a series of keys starting from a single seed phrase or mnemonic. This deterministic approach ensures that all wallet keys are derived from a single root, greatly simplifying backup and fund restoration.

To achieve this, the Bitcoin Development Kit (BDK) library provides a versatile solution for generating seed phrases. Using the "GeneratableKey" trait and "GeneratableDefaultOptions," it's possible to generate a new BIP32::ExtendedPrivKey (extended key) or BIP39 mnemonic in a customized manner or utilizing a predefined entropy source.

For example, using the "PrivateKey::generate_default()" method generates a new private key with default options. Alternatively, with "PrivateKey::generate(PrivateKeyGenerateOptions { compressed: false })", you can generate a private key with custom options, such as without compression.

The library also provides a "ToDescriptorKey" trait, allowing the use of generic key types within descriptors. This trait is already implemented for rust-bitcoin's native key types like PrivateKey, PublicKey, bip32::ExtendedPrivKey, and bip32::ExtendedPubKey.

Furthermore, when enabling the "keys-bip39" feature, the implementation for BIP39 mnemonics and their seeds is also available. Finally, with the "DerivableKey" trait, it's possible to manage custom key types representing

xprv or xpub, simplifying descriptor use with such keys. These features enable flexible and secure generation of seed phrases, extended keys, and other cryptographic keys, adapting them to each user's specific needs.

In our case, we will implement a solution where the user can generate keys using an internal entropy source within the car or directly import their keys for use.

5.2.2 Transactions

Within the Car-Wallet, the updated API of the Bitcoin Development Kit (BDK) library will be utilized to manage transactions efficiently and securely. The use of TxBuilder will enable transaction creation, with careful configuration of each operation. By referencing the main wallet instance within TxBuilder, checks will be performed when adding information to the transaction, avoiding errors and allowing recovery in unforeseen situations.

Key functionalities to be used include specifying desired outputs, indicating the recipient and transaction amount. It will also be possible to set nLockTime, allowing transaction execution to be scheduled for a specific time. Additionally, control over the Unspent Transaction Outputs (UTXO) used in the transaction will optimize selection to reduce transaction fees.

With the enhanced API of the BDK library, enabling Replace-By-Fee (RBF) will allow dynamic fee increases, enabling transaction updates as needed or expediting execution in the presence of Bitcoin network congestion.

Thanks to these new features, the Car-Wallet will be capable of reliably managing transactions, optimizing costs, and ensuring greater recovery capabilities from any errors. These functionalities will not be directly used by the car user, as they will not see an actual Bitcoin wallet interface on the car's interface. Instead, they will be leveraged in the development of Lightning Network and RGB wallet features.

5.3 Lightning Network Wallet

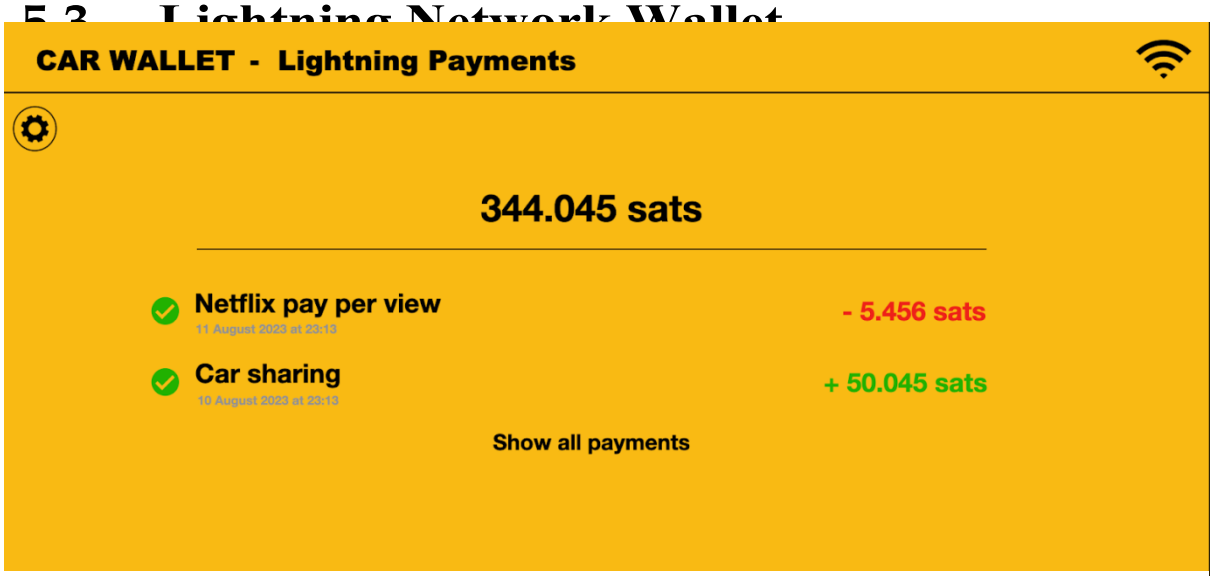


Figure 5.1: User Interface Draft for Lightning Wallet

As can be seen from this simple UI, the goal is to create a very clear and intuitive UX. Through the "send" button, you can send satoshis, and with "receive," you can receive them. The top-left button provides access to the settings page, where security settings, active "subscriptions," and personalized settings for linking your node can be configured, if needed.

From a technical perspective, as previously announced, interactions with the Lightning Network will be managed using the Breez SDK and a Greenlight server, allowing the user to have full control of their funds without efforts related to system management and understanding.

The choice to use the Breez SDK is based on its ability to provide a reduced learning curve and well-structured APIs that simplify the integration of Lightning functionalities into our system.

The architecture of the Car-Wallet will be designed with a modular approach, where the Breez SDK will play a central role.

The "Signer" module will handle all Lightning message signing operations, ensuring transaction security.

The "InputParser" will be responsible for parsing user input for sending and receiving payments, interpreting data and executing specific actions based on the protocol used (e.g., bolt11, lnurl-pay, Lightning address, etc.).

The "LightningNode" module will serve as an interface for interacting with the user's Lightning node, allowing for the creation of a new node or connection to an existing node, as well as managing payment operations and invoice creation.

Currently, Greenlight will be the utilized provider, but the architecture is designed to support the addition of other providers in the future.

The general architecture will be devised to fully leverage the features offered by the Breez SDK, efficiently integrating Lightning Network operations into our Car-Wallet. This will enable us to provide users with a fast, secure, and convenient payment experience, utilizing the power of the Lightning network to manage quick and reliable transactions directly on board the vehicle.

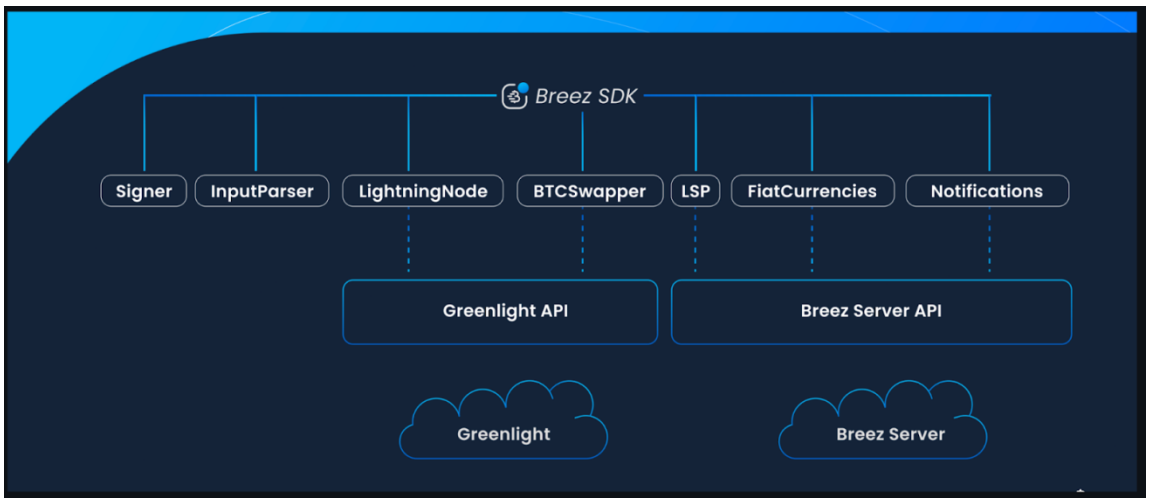


Figure 5.2: Flow chart of Greenlight Functions (*Source*)

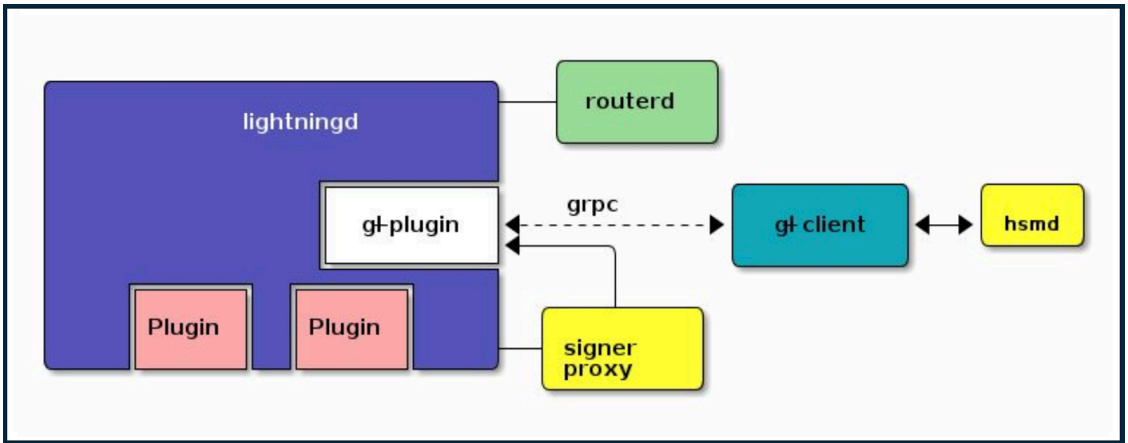
This covers the "client" part of our Lightning wallet. Now, let's delve into the section dedicated to managing the Lightning node and how the Greenlight service is handled.

Greenlight provides and manages the Core Lightning nodes, ensuring that the car wallet has access to a functioning node for Lightning operations. Utilizing the grpc interface exposed by the nodes, the car wallet can communicate with Greenlight and make use of its functionalities without any limitations.

Security is a key element in the implementation of Greenlight. mTLS authentication ensures that all communications with Greenlight are encrypted and authenticated, preventing unauthorized access and safeguarding sensitive data. The car wallet nodes will possess unique identities certified by Greenlight, enabling them to securely authenticate during interactions with the service.

The Signer module will manage private information by demonstrating node ownership during registration and recovery phases. The presence of the Signer will ensure that operations are secure and authorized, mitigating potential security threats and protecting the car wallet's assets.

The Scheduler will be responsible for monitoring the status of Greenlight nodes and initiating them when necessary. As a result, the car wallet will have access to an active node at all times, ensuring a seamless and uninterrupted user experience during Lightning transactions.



5.3.2 Implementation

For the implementation of the Car-Wallet Lightning Network, functions provided by the Breez SDK are used, as mentioned previously. This greatly simplifies payment management.

For instance, to receive Lightning payments, it's not necessary to open a channel and configure incoming liquidity, as the Breez SDK automatically connects the node to the Lightning Service Provider (LSP). This way, the Car-Wallet will be able to receive payments without further complications. The interaction for payment reception can be implemented as follows:

Figure 5.3: Flow chart of Greenlight Functions (*Source*)

```python
try:
  invoice = sdk_services.receive_payment(3000, "Invoice for 3000 sats")
except Exception as error:
  # Error handling
```

Similarly, to send Lightning payments, it's sufficient to use the Breez SDK with the recipient's bolt11 and the desired amount. Sending payments will be as simple as in the following example:

```python
bolt11 = "..."
try:
  sdk_services.send_payment(bolt11, 3000)
except Exception as error:
  # Error handling
```

The Car-Wallet Lightning Network is also capable of making spontaneous payments. This type of payment allows sending funds to a specific node without

the need for an initial invoice request. The implementation of spontaneous payments can be done as illustrated in the following example:

```python
node_id = "..."
try:
  sdk_services.send_spontaneous_payment(node_id, 3000)
except Exception as error:
  # Error handling
```

Thanks to the implementation of these examples using the Breez SDK, the Car-Wallet Lightning Network will be able to conduct payments swiftly, securely, and highly efficiently. However, to achieve this, it will need to connect to a node, which in our case will be managed through Greenlight. This way, users can utilize the service in a non-custodial manner without handling the complexity of node management.

For illustrative purposes, we will provide an explanation of how the registration of a node on Greenlight takes place.

To implement the node on Greenlight, we will start by creating a node identity, which consists of a node secret seed and the corresponding mTLS certificates. These will be used for authentication on Greenlight later. To ensure security, the secret seed will be generated from a BIP 39 seed phrase, allowing the user to perform a secure backup in a physical form such as paper, steel plate, or another creative storage medium.

Next, we will instantiate an mTLS identity for authentication with Greenlight. Since the node is not yet registered, we will use a dummy key that allows communication with the Scheduler but won't permit communication with other services. Once the node is registered, we will obtain a valid certificate for authentication.

After creating the mTLS identity, we will proceed with implementing the Signer, responsible for incoming signature requests. It will be used during node registration to demonstrate ownership of the private key.

Subsequently, we will proceed with registering the new node with the Greenlight Scheduler. Node registration creates node metadata on the Greenlight service, including node identity and public key, ensuring everything is set up for node initiation.

During registration, proving access to the corresponding private key will be necessary. Since the private key is solely managed by the Signer, we will need to pass the Signer to the Scheduler. The registration result will contain credentials we can use later to communicate with the Scheduler and the node itself.

Carefully preserving these credentials will be crucial, as anyone with access can control the node. Lastly, during development and gradual service expansion, it might be necessary to request an invitation code or partner certification to register a new node on the Greenlight service.
In summary, through this registration process, our Car-Wallet node will have a complete identity on Greenlight, enabling us to securely leverage Lightning network functionalities, providing a robust platform for executing Lightning transactions within the vehicle.

5.4 M2M Micro Payments

5.4.1 Introduction

This chapter focuses on the implementation of machine-to-machine (M2M) payments within the Car-Wallet using the L402 protocol. The goal is to demonstrate how it's possible to integrate micro payments between devices within the context of the Car-Wallet. This approach opens new opportunities and perks, enabling the vehicle to conduct automatic and instant transactions with other connected devices, such as charging stations, highway tolls, or parking services.

5.4.2 Implementation

To realize M2M payments in the car wallet, the L402 protocol will be used for payment authentication and authorization between the vehicle and other network devices. This protocol enables the car wallet to send payment requests containing Macaroon and invoice preimage to desired backend services.

The authentication procedure requires the client to pay the invoice provided by the server, thus revealing the invoice preimage. Subsequently, the client will construct the L402 token by concatenating the base64-encoded Macaroon with the hexadecimal-encoded preimage. This token will then be used for authorized requests to the backend service.

The security of M2M payments implemented through L402 relies on using a secure external system, such as TLS, for communication encryption, as Macaroon and preimage are transmitted in clear text over the network. Protecting L402 credentials is essential to prevent abuse or unauthorized access.

Here's an example of implementing a Macaroon in the Car-Wallet using the library provided by the L402 protocol. To build a Macaroon, a secret key and a condition need to be defined. In the following Python code, we'll use "super_secret_key" as the secret key and "user_id = 1234" as the Macaroon condition:

python Copy code

```python
from l402 import Macaroon, Version, Caveat

# Definition of keys and conditions
key = b"super_secret_key"  # Macaroon secret key
condition = b"user_id = 1234"  # Macaroon condition

# Creating the Macaroon
macaroon = Macaroon.create(key, identifier="Car-Wallet", version=Version.V2)
macaroon.add_first_party_caveat(condition)

# Serializing the Macaroon to base64 for transmission
macaroon_serialized = macaroon.serialize()
macaroon_base64 = macaroon_serialized.encode("base64")
print("Macaroon: ", macaroon_base64)
```

Once the Macaroon is created, the Car-Wallet will be able to use it to authenticate with other network services via the L402 protocol.

Now, let's see how to construct an L402 token for an authorized payment request. The L402 token is formed by concatenating the Macaroon and the invoice preimage. Below is an example of Python code to build an L402 token using the previously generated Macaroon and invoice preimage:

```python
# Macaroon and invoice preimage data
macaroon_base64 = "AGIAJEemVQUTEyNCROexk7ek90Cg=="  # Base64-encoded Macaroon
preimage_hex = "1234abcd1234abcd1234abcd"  # Hexadecimal-encoded invoice prei

# Constructing the L402 token by concatenating the Macaroon and invoice preim
token_l402 = f"{macaroon_base64}:{preimage_hex}"
print("Token L402: ", token_l402)
```

By using the L402 token, the Car-Wallet can conduct machine-to-machine payments with other devices within the network, ensuring authentication and authorization through the L402 protocol. This enables secure and reliable implementation of micro M2M payments within our Car-Wallet.

5.4.3. Future Developments

The use of the L402 protocol for M2M payments in the Car-Wallet opens new development prospects. The ability to manage instant payments between the vehicle and other devices could be extended to a wide range of services, such as electric charging, highway toll payments, or parking services.

The implementation of M2M payments through L402 represents a significant step towards greater integration of vehicles with the connected services ecosystem. This technology opens new service opportunities and simplifies transactions, turning the Car-Wallet into an advanced and secure hub for electronic payments within the transportation and mobility ecosystem.

5.5 Wallet RGB

5.5.1 Introduction

The Wallet RGB is built upon the "rgb-lib" Rust library, which provides tools for constructing RGB-compatible wallets across platforms, without the need to concern ourselves with the internal details of Bitcoin and RGB.

The library primarily leverages "bdk," a library already utilized within the Car-Wallet, to provide Bitcoin wallet functionalities and various RGB libraries such as "rgb-core" to offer RGB-specific capabilities.

5.5.2 Implementation

The Wallet RGB offers a straightforward and efficient interface for managing RGB-based assets.
Also here, the aim is to maintain a very simple and intuitive interface. On the

Figure 5.4: User Interface Draft for RGB Wallet

main page, a list of contracts signed by the car owner is displayed. Clicking on each contract opens a detailed view of the individual contract. Finally, through

the button at the bottom right, you can access into a new interface where a new contract can be signed.

Using the "rgb-lib" library, the RGB Wallet leverages the capabilities of "bdk" to manage the Bitcoin wallet and interact with the Bitcoin network. Simultaneously, it utilizes various RGB libraries such as "rgb-core" to allow the management of RGB-specific assets.

One of the key features offered by this library is the ability to function in an offline mode, enabling the use of APIs even without an internet connection. Additionally, it can be used in "watch-only" mode, where private keys are not needed for information consultation, but they will need to be used by other applications for signing operations.

The implementation of the RGB Wallet ensures secure management of the wallet's UTXOs, preventing the loss of RGB assets due to management errors. Whenever the wallet goes online, a consistency check is performed to verify that the set of UTXOs remains unchanged from the previous synchronization. In case of discrepancies, an error is returned, signaling any anomalies.

Here are some examples of how the rgb-lib library is seamlessly implemented within our car wallet:

Building Blinded UTXOs:

python Copy code

```python
# Initialize the car owner's wallet
recv_wallet = rgb_lib.Wallet(recv_wallet_data)
recv_online = recv_wallet.go_online(False, electrum_url)

# Create two Blinded UTXOs to receive assets
recv_blind_data_1 = recv_wallet.blind(None, None, None, transport_endpoints)
recv_blind_data_2 = recv_wallet.blind(None, None, None, transport_endpoints)

print(f'Blinded UTXO 1: {recv_blind_data_1.blinded_utxo}')
print(f'Blinded UTXO 2: {recv_blind_data_2.blinded_utxo}')
```

Building RGB25 Contracts:

```python
# RGB25 contract data
name = 'Ferrari Purosangue ownership contract'
precision = 0
amounts = [1]
description = ' [spec] '
parent_id = None
file_path = 'sample.png'

# Issuing the asset
rgb25_asset = send_wallet.issue_asset_rgb25(
    send_online, name, description, precision, amounts, file_path)
print(f'Issuing contract with ID: {rgb25_asset.asset_id}')
```

These examples demonstrate how the rgb-lib library simplifies interactions with RGB assets. With just a few lines of code, we can easily build Blinded UTXOs and issue RGB25 contracts. This allows us to securely and reliably manage assets, providing our car wallet with advanced and flexible capabilities for RGB-based asset management.

5.5.3 Use Case

In questo capitolo voglio descrivere il caso d'uso preso in esame: un contratto di ownership emesso da un car maker e firmato dall'acquirente dell'auto. Nello specifico questo contratto è stato realizzato con LNPBP-25 uno standard che definisce un token crittografico non fungibile. Nel nostro caso è stato utilissimo poichè non è stato necessario creare un modello da zero ma abbiamo potuto utilizzarne uno già esistente.
Questo contratto è costruito in maniera molto semplice.

5.5.3.1 Introduction

In this chapter, we present the realization of a Bitcoin Car-Wallet system that leverages the RGB protocol to issue ownership contracts for autonomous driving cars. This system facilitates secure ownership transfer and transaction validation between car makers and car owners.

5.5.3.2 Bitcoin Car-Wallet Architecture

The Bitcoin Car-Wallet is designed to provide a secure and decentralized platform for managing contracts and transactions for autonomous driving cars. It is built on top of the Lightning Network and utilizes the RGB protocol for issuing and transferring ownership contracts.

Figure 5.5: User Interface Draft to Show a Contract Details

The system involves two main participants: the car maker and the car owner.

Car Maker

The car maker is responsible for initiating the issuance of ownership contracts for the autonomous driving cars. The following steps outline the process:

1. Wallet Generation: A new wallet is generated for the car maker using the `rgb_lib.generate_keys()` function. This wallet will be used to issue ownership contracts.

2. Wallet Initialization: The generated mnemonic and xpub are used to initialize the wallet. This ensures consistency between wallet instances.

3. Funding the Wallet: The car maker's wallet is funded with regtest bitcoins using a designated address obtained from the wallet.
4. Creating UTXOs: Unspent transaction outputs (UTXOs) are created in the car maker's wallet to hold RGB allocations. These UTXOs will later be used to issue ownership contracts.

5. Issuing Ownership Contract: The car maker uses the wallet to issue an RGB25 ownership contract. The contract includes relevant information

such as the contract name, description, and an associated image. This contract represents the ownership of an autonomous driving car.

6. Transfer to Car Owner: The ownership contract is transferred to the car owner's wallet. This involves creating a partially signed Bitcoin transaction (PSBT) and transferring the RGB ownership contract as an RGB transfer.

Car Owner

The car owner receives the ownership contract and validates it before accepting ownership of the autonomous driving car. The process is as follows:

1. Wallet Generation: Similar to the car maker, the car owner generates a new wallet using the `rgb_lib.generate_keys()` function.

2. Wallet Initialization: The car owner's wallet is initialized using the generated mnemonic and xpub.

3. Funding the Wallet: The car owner's wallet is funded with regtest bitcoins using a designated address obtained from the wallet.

4. Creating UTXOs: UTXOs are created in the car owner's wallet to receive the blinded RGB ownership contract.

5. Receiving Ownership Contract: The car owner receives the RGB25 ownership contract from the car maker. This contract is in a blinded state to ensure privacy.

6. Validating and Accepting: The car owner validates the ownership contract and acknowledges its reception. The contract is then unblinded to reveal its contents, including the associated image.

7. Transaction Confirmation: To finalize the ownership transfer, the car owner mines a block to confirm the transaction.

5.5.3.3 System Implementation

The implementation of the Bitcoin Car-Wallet system involves the use of the `rgb-lib` library, which provides Python bindings for the RGB protocol. The following code snippets provide an overview of the key steps taken by both the car maker and the car owner.

Car Maker Implementation

The car maker initiates the process by generating its wallet, issuing the ownership contract, and transferring it to the car owner:

```
# Wallet Generation
send_keys = rgb_lib.generate_keys(bitcoin_network)

# Wallet Initialization
send_data_dir = './data/send_wallet'
send_mnemonic = send_keys.mnemonic
send_xpub = send_keys.xpub

# ... (Wallet initialization, funding, UTXO creation)

# Issuing Ownership Contract
rgb25_asset = send_wallet.issue_asset_rgb25(send_online, name, description, precision, amounts, file_path)

# Transferring to Car Owner
amount_rgb25 = 1
recipient_map_rgb25 = {
    rgb25_asset.asset_id: [
        rgb_lib.Recipient(recv_blind_data_2.blinded_utxo, amount_rgb25, transport_endpoints),
    ]
}
txid = send_wallet.send(send_online, recipient_map_rgb25, False, 1.5)
```

Car Owner Implementation

The car owner receives and validates the ownership contract before acknowledging and finalizing the transaction:

```
# Wallet Generation
recv_keys = rgb_lib.generate_keys(bitcoin_network)

# Wallet Initialization
recv_data_dir = './data/recv_wallet'
recv_mnemonic = recv_keys.mnemonic
recv_xpub = recv_keys.xpub

# ... (Wallet initialization, funding, UTXO creation)

# Receiving Ownership Contract
recv_blind_data_1 = recv_wallet.blind(None, None, None, transport_endpoints)
recv_blind_data_2 = recv_wallet.blind(None, None, None, transport_endpoints)

# Validating and Accepting
recv_wallet.refresh(recv_online, None, [])

# Transaction Confirmation
# (Mines a block to confirm the transaction)
```

5.5.3.4 Conclusion

The Bitcoin Car-Wallet system presented in this chapter demonstrates a practical application of the RGB protocol for issuing and transferring ownership contracts for autonomous driving cars. The system leverages the security and decentralization features of the Lightning Network and Bitcoin. By providing a step-by-step implementation guide, we have shown how car makers can issue ownership contracts and how car owners can securely receive, validate, and accept these contracts..

5.5.4 Future Developments

The RGB Wallet is still in the development and testing phase, and the current version is subject to changes that could break compatibility with APIs. We expect that new versions will address bugs and introduce new features to enhance user experience.

Additionally, the library envisions the implementation of native bindings to other languages through the "rgb-lib-ffi" project, thus providing the opportunity to use the RGB Wallet in various development environments.

Chapter 6

Results and Validation

6.1 Introduction

In this chapter, we delve into the analysis of the outcomes derived from the design study of the Car-Wallet, focusing on the methodologies employed. We place particular emphasis on the implementation strategy, the constructed testing environment, and a comprehensive comparison with existing solutions. This section is instrumental in validating the efficacy and innovation of the Car-Wallet concept, which integrates Bitcoin, the Lightning Network, and RGB technology into self-sovereign, trustless, and autonomous vehicles.

6.2 Implementation and Methodology

At the heart of the Car-Wallet's implementation and methodology lies an unwavering commitment to enabling users' complete self-sovereignty. This autonomy ensures that users can interact with the system in a trustless manner, free from the need to rely on intermediaries. To attain this ambitious goal, a two-fold approach was adopted. Initially, the foundational principles of the Bitcoin protocol were embraced. Subsequently, a constellation of open-source technologies, including BDK, Breez SDK, Greenlight, and RGB, were woven together to form the technological fabric of the Car-Wallet.

The methodology chosen not only safeguards against censorship and surveillance but also underscores the underlying principles of empowerment and security intrinsic to Bitcoin technology. While this thesis does not

encompass the exhaustive array of software necessary for the Car-Wallet's holistic functionality, it does present a pioneering prototype of an RGB contract. Specifically, the proof-of-concept involves a contract defining ownership of an autonomous vehicle—a tangible testament to the viability of the Car-Wallet's concept.

6.3 Experimental Setup

The validation process took shape through the implementation of an RGB contract governing the ownership dynamics between the manufacturer and the end customer. This experimental setup was facilitated by harnessing the capabilities of the RGB Python library. The library, combined with Docker containers, facilitated a seamless interaction within a Jupiter server environment. This dynamic environment allowed for Python script experimentation, thereby simulating the entire contract lifecycle—creation, signing, and exchange—within a controlled Bitcoin test network.

Crucially, this setup expedited experimentation without relying on real Bitcoin transactions or the time-consuming process of mining. Consequently, a practical testing platform emerged, permitting rapid prototyping. Within this context, an LNPBP-25 contract, manifesting as an NFT, was implemented, portraying the digital passport concept for the autonomous vehicle.

6.4 Comparison with Existing Solutions

Although the current iteration of the solution remains confined to the testing realm, developers caution against deploying it on the mainnet due to the ongoing testing phase. However, the implemented code demonstrates resilience and robustness. Importantly, the client-side validated approach employed here offers an unparalleled level of privacy compared to solutions built on the foundation of distributed virtual machines, such as those seen in certain NFT projects like Alfa Romeo Tonale. Such projects' transparency compromises user privacy and security, a drawback artfully sidestepped by the Car-Wallet's approach.

6.5 Discussion of Results

The result achieved is nothing short of remarkable, primarily for its simplicity of implementation. The asset generated not only boasts efficiency in creation but also underscores the resounding supremacy of the client-side validation approach over the distributed virtual machine-based approach of traditional smart contracts. As highlighted earlier, this contract's distinctiveness lies in its ability to ensure the security and incorruptibility intrinsic to Bitcoin while concurrently safeguarding an exceptionally high degree of privacy.

By seamlessly integrating Lightning Network and RGB technologies into the realm of autonomous vehicles, the Car-Wallet demonstrates the transformative potential of blockchain in redefining trust, autonomy, and security in future transportation ecosystems. This groundbreaking exploration paves the way for a new era of truly decentralized and self-governing vehicles, capable of fostering a trustless network for enhanced mobility and interaction.

Chapter 7

Conclusions

7.1 Recapitulation of Objectives

Within this section, a retrospective examination of the thesis objectives is presented, as initially outlined in the introductory chapter. These objectives encompassed the creation of a proof of concept for a "Car-Wallet," capable of managing digital payments and cryptographic assets. Beyond this, the objectives aimed to:

- Survey the contemporary landscape of automotive systems.
- Analyze the functional properties afforded by Bitcoin, the Lightning Network, and RGB.
- Develop a car wallet proof of concept for managing Lightning Network payments and RGB cryptographic contracts.
- Establish a cryptographic identity contract for the vehicle and its state traceability.
- Analyze results and simulate a use case.

An additional overarching goal was to provide a tool that harnesses autonomous driving technology while safeguarding against the risk of mass surveillance. This objective aligns with the pursuit of a trustless and uncensorable approach.

7.2 Implications of Findings

This section delves into the insights gleaned throughout the thesis and the conclusions drawn from them. A critical realization emerged: Bitcoin plays a pivotal role in preventing the technology from devolving into a tool for mass surveillance, similar to contemporary banking and credit card systems. Bitcoin's incorporation enables the creation of self-sovereign cars—autonomous entities capable of complete independence.

Throughout this exploration, the suitability of various tools was meticulously examined. The Bitcoin community's ceaseless efforts to develop trustless services were a captivating focal point. The likes of Breez SDK, Greenlight, and RGB all align with the overarching goal of enabling uncensorable, trustless interactions. The proof of concept attests to the effective utilization of these tools.

The implications of implementing such a service are far-reaching. A borderless payment solution like Bitcoin enables global car usage without the hindrance of payment circuitry. Privacy is robustly protected, eliminating the need for users to divulge their identities to every party they engage with. Moreover, the establishment of a microtransaction M2M infrastructure opens doors to novel business models and a plethora of potential applications.

7.3 Future Work and Extensions

A substantial portion of the technology employed in this proof of concept resides in experimental stages, with some components still in active development, such as RGB. Consequently, updates will be required to accommodate protocol upgrades and potentially integrate new technologies.

An additional pivotal consideration for the future lies in the integration with various automotive operating systems. As expounded upon in this study, each car maker has devised its unique operating system. Hence, the implementation of such technology demands a customized approach within distinct vehicle models. It's worth contemplating the potential convergence of existing operating systems into a more open and shared model. This evolution could empower users to curate their in-car experiences, mirroring the current paradigm of choice in the realm of mobile apps.

Moreover, the realm of RGB-based contracts presents fertile ground for expansion and innovation. While the initial spotlight has been on NFT-style ownership contracts for vehicles, an exciting array of possibilities unfurls within the broader automotive ecosystem. Beyond ownership, RGB-based contractual frameworks could encompass a multitude of domains, including supply agreements, maintenance contracts, real-time performance tracking, and

beyond. This multifaceted potential underscores the transformative impact of Bitcoin protocol, capable of revolutionizing not only how we interact with vehicles but also how the entire automotive industry operates.

As technology matures and the ecosystem expands, it's plausible that a novel paradigm emerges, marked by an intricate network of interconnected RGB-based contracts orchestrating seamless interactions across the automotive landscape. This evolution could transcend mere vehicles, engendering an ecosystem in which intelligent contracts play an integral role in redefining how we access, operate, and engage with vehicles—a truly revolutionary step forward.

7.4 Closing Remarks

In this closing segment, a reflection on the experience of collaborating with Turin Tech S.r.l. on this project is shared. The profound impact of technology in shaping a better world has been a lifelong aspiration. Bitcoin, as a fundamental piece in the mosaic of our civilization, offers the potential for an evolutionary leap, paving the way for the next stage of progress. The collaboration has not only realized concrete innovation but also further fueled the drive toward a future marked by technological empowerment and transformative change.

This collaborative journey has illuminated the immense potential of merging cutting-edge Bitcoin technologies with the automotive realm. As we steer toward a horizon characterized by self-sovereign cars, trustless interactions, and decentralized control, the groundwork laid within this endeavor serves as a testament to our unwavering dedication to reshaping the future of mobility. Through collaboration, innovation, and the harnessing of emergent technologies, we propel ourselves into an era where vehicles transcend their mechanical nature, morphing into dynamic entities that empower individuals and societies alike.

Bibliography

[1] Bitcoin developer guide. https://bitcoin.org/en/developer- reference.
[2] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf, 2008.
[3] Back, A. Hashcash - a denial of service counter-measure. Tech. rep., 2002.
[4] Wasabi coinjoin. https://docs.wasabiwallet.io/getting-started#coinjoin .
[5] Antonopoulos, A. M. Mastering Bitcoin. O'Reilly, 2017.
[6] Poon, J., and Dryja, T. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning- network-paper.pdf, 2016.
[7] William, J. (2016). Smart Contracts: The Essential Guide to Blockchain Smart Contracts for Developers and Blockchain Enthusiasts.
[8] Zheng, X. (2020). An overview on smart contracts: Challenges, advances and platforms.
[9] Buterin, Vitalik: Ethereum Whitepaper, https://ethereum.org/en/whitepaper/ .
[10] RGB website, https://rgb.info/ .
[11] Todd, Closed Seal Sets and Truth Lists for Better Privacy and Censorship Resistance, https://petertodd.org/2016/closed-seal-sets-and-truth-lists-for-privacy
[12] KPMG, CAR. Self-driving cars: the next revolution. Ann Arbor, MI, 2012.
[13] B. Hayes. "Leave the driving to it", Am. Sci., 99(2011), pp. 362-366.
[14] Jessica Shea Choksey and Christian Wardlaw, "Levels of Autonomous Driving, Explained", JDPower, https://www.jdpower.com/cars/shopping-guides/levels-of-autonomous-driving-explained .
[15] James Morris, "Self-Driving Cars Won't Go Mainstream Until We Solve This Problem", Forbes, https://www.forbes.com/sites/jamesmorris/2021/02/13/self-driving-cars- wont-go-mainstream-until-we-solve-this-problem/?sh=1221d3472f3b .
[16] Kersten Heineke, Ruth Heuss, Ani Kelkar, and Martin Kellner, "What's next for autonomous vehicles?", McKinsey & Company, aggiornato al 22/12/2021, https://www.mckinsey.com/features/mckinsey-center-for-future-mobility/our-insights/whats-next- for-autonomous-vehicles
[17] Nilesh Barla, "Self-Driving Cars With Convolutional Neural Networks (CNN)", Neptuneblog, https://neptune.ai/blog/self-driving-cars-with-convolutional-neural-networks-cnn

[18] "How Machine Learning in Automotive Makes Self-Driving Cars a Reality", Mindy Support, https://mindy-support.com/news-post/how-machine-learning-in-automotive-makes-self-driving-cars-a-reality/
[19] Enrique Dans, "The Future Of Autonomous Vehicles: Product Or Service?", Forbes, aggiornato al 11/06/2021, https://www.forbes.com/sites/enriquedans/2021/06/11/the-future-of- autonomous-vehicles-product-orservice/?sh=45167e6b5892
[20] Mario Herger, "Aurora-Uber: Set Up For Failure?", The Last Driver License Holder… , https://thelastdriverlicenseholder.com/2021/01/06/aurora-uber-set-up-for-failure/
[21] Matthew DeBoard, "Waymo could be worth as much $175 billion — here's a brief history of the Google Car project", Insider, aggiornato al 09/09/2018, https://www.businessinsider.com/google-car-project-history-2018-8?r=US&IR=T
[22] "Company Profile: Waymo", CTDT, https://www.carsthatdrivethemselves.com/waymo/
[23] Bernard Marr, "Key Milestones Of Waymo - Google's Self-Driving Cars", Forbes, https://www.forbes.com/sites/bernardmarr/2018/09/21/key-milestones-of-waymo-googles-self-driving-cars/?sh=7c0e87065369 .
[24] Zong, W.; Zhang, C.; Wang, Z.; Zhu, J.; Chen, Q. Architecture Design and Implementation of an Autonomous Vehicle. IEEE Access 2018, 6, 21956–21970.
[25] Cruise self-driving cars website, https://getcruise.com/ .
[26] Cuise self-driving wiki page https://en.wikipedia.org/wiki/Cruise_(autonomous_vehicle) .
[27] "Tesla (company)", Wikipedia, https://it.wikipedia.org/wiki/Tesla_(azienda) .
[28] "BMW is going all-in on in-car microtransactions", The Verge, https://www.theverge.com/2020/7/2/21311332/bmw-in-car-purchase-heated-seats-software-over-the-air-updates .
[29] "FinTech in Automotive: How to Implement In-Car Payments in Connected Vehicles
" , Intellias, https://intellias.com/fintech-in-automotive-how-to-implement-payment-systems-in-connected-cars/ .
[30] "In-Vehicle Payments System Market Size", Fortune, https://www.fortunebusinessinsights.com/in-vehicle-payment-system-market-103653 .
[31] "Waymo's self-driving cars will be available on Uber's app, starting in Phoenix ", Tech Crunch, https://techcrunch.com/2023/05/23/waymos-self-driving-cars-will-be-available-on-ubers-app-starting-in-phoenix/ .
[32] Lightning Network statistics website, https://1ml.com/statistics
[33] "M2M payments with Lightning Network", Medium, https://medium.com/@ABussutil/m2m-payments-with-lightning-network-a472562b181 .

[34] “How Machine-to-Machine, IoT, Blockchain, and In-Vehicle Payments are Revolutionizing the Payment Industry
”, Benedict Xavier, https://www.linkedin.com/pulse/how-machine-to-machine-iot-blockchain-in-vehicle-payments-xavier .
[35] “LSAT: Authentication and Payments for the Lightning-Native Web”, Olaoluwa Osuntokun, https://lightning.engineering/posts/2020-03-30-lsat/ .
[36] “Blockchain in the Automotive Sector: BMW's Innovation”, BMW website https://www.bmw.com/en/innovation/blockchain-automotive.html
[37] “Automakers Are Minting NFTs, But Is There a Strong Use Case?”, Cointelegraph, https://cointelegraph.com/news/automakers-are-minting-nfts-but-is-there-a-strong-use-case .
[38] “LG CNS, in decentralized identity project for autonomous cars using blockchain, 5G”, https://www.ledgerinsights.com/lg-cns-autonomous-cars-blockchain-5g-decentralized-identity/ .
[39] “DID, Decentralized identifiers”, W3C Reccomendation, https://www.w3.org/TR/did-core/ .
[40] “What is BIP 32?”, Trezor website, https://trezor.io/learn/a/what-is-bip32 .
[41] “What is BIP 39?”, Trezor website, https://trezor.io/learn/a/what-is-bip39 .
[42] Breez SDK website, https://breez.technology/sdk/ .
[43] Greenlight website, https://blockstream.com/lightning/greenlight/ .
[44] RGB design website, https://docs.rgb.info/design .
[45] “Public key infrastructure”, https://www.tutorialspoint.com/cryptography/public_key_infrastructure.htm .
[46] “An overview of the SSL Handshake”, Robert van Rijn, Medium, https://medium.com/@vanrijn/an-overview-of-the-ssl-handshake-3885c37c3e0f .
[47] Bitcoin Dev Kit source code, https://github.com/bitcoindevkit .
[48] L402 builders guide, https://docs.lightning.engineering/the-lightning-network/l402 .
[49] RGB python library source code, https://github.com/RGB-Tools/rgb-lib-python/blob/master/demo/README.md .