DEPARTMENT OF
INFORMATION SYSTEMS
FREIE UNIVERSITÄT BERLIN

Master Thesis

Zero-Knowledge Proof Algorithms: A Systematic Literature Review and Application in the Aviation Industry

Elina Beletski

Supervisor:	Univ.-Prof. Dr. rer. pol. Natalia Kliewer
Semester:	Winter term 2022/23
Author:	Elina Beletski
Student ID:	5504054
Address:	Nürnberger Str. 18, 10789 Berlin
Email:	elina.beletski@fu-berlin.de
Phone:	+4917620310530
Studies:	M.Sc. Information Systems

Date: **12 June 2023**

Abstract

The modern aviation industry, with its millions of aircraft parts and numerous stakeholders, faces the challenges of documenting and verifying the maintenance, repair, and overhaul (MRO) events throughout an aircraft's lifecycle. The paper-based documentation system lacks digital process automation, resulting in disconnected and scattered MRO event entries across various enterprise resource planning systems, compromising transparency and traceability. This situation impedes efficient secondary parts trading and poses challenges in verifying the documentation for airworthiness and traceability in the event of accidents or incidents. To address these challenges, the research project RAPADO aims to drive digitalization in the aviation industry by establishing an industry standard for seamless documentation and verification of aircraft spare parts. This research investigates the applicability of blockchain technology, precisely zero-knowledge proofs (ZKPs), to meet the verification and information confidentiality requirements for aircraft spare parts documentation. A systematic literature review surveys the current research on ZKPs, providing a comprehensive overview of the most practically used algorithms and their applications in various domains. However, their potential application in the aviation industry still needs to be explored. This master thesis bridges this gap by presenting practical implementations of ZKPs, starting with a step-by-step overview of the Groth16 algorithm through a simple polynomial example. Subsequently, a zero-knowledge decentralized application (zk-DApp) is proposed and evaluated for MRO data attestation and verification to balance transparency and data confidentiality. A fraud-preventive zero-knowledge data structure based on Merkle trees and ZKPs is also introduced to facilitate authenticity checks of spare part certificates. The findings of this research have implications for both scholars and practitioners. The systematic literature review provides an overview of ZKPs. It highlights their potential applications across various fields, while the practical implementations offer concrete solutions for the aviation industry. The proposed zk-DApp and zero-knowledge data structure contribute to securing and efficiently verifying aircraft spare parts documentation, promoting transparency, traceability, and data integrity. Future research directions include broader adoptions of zk-rollups and recursive ZKPs and re-assessing quantum-resistant algorithms' development state and applicability.

Contents

List of Figures **v**

List of Tables **vi**

List of Abbreviations **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Contribution Outline 2
1.3 Structure 3

2 Related Work **4**
2.1 Topic-related Work 4
2.2 Project-related Work 5
2.3 Related Systems 7

3 Methodology **9**
3.1 Systematic Literature Review 9
3.2 Implementation and Evaluation 12

4 Zero-Knowledge Proofs **14**
4.1 From Proof Systems towards Argument Systems 15
4.2 Zero-Knowledge Succinct, Non-interactive Arguments of Knowledge 23
4.3 Application Domains and Use Cases 32
4.4 Evaluation and Challenges 41

5 Implementation and Results **45**
5.1 Summary of Requirements 45
5.2 Groth16 Proof and Verification 46
5.3 PLONK-based zk-DApp 63
5.4 Evaluation 69

6 Conclusion **74**
6.1 Discussion of Results 75
6.2 Outlook 75

Bibliography **77**

Appendix **88**

List of Figures

3.1 Concept Map - Compression of Main Topics and Relations as Basis for Final Search String 10
3.2 Concept Matrix - Outcome Summary of the Systematic Literature Review 11

4.1 Illustration of Generic Fiat Shamir Transformation (based on Thaler (2023)) 18
4.2 PLONK Construction Steps 26
4.3 Example Polynomial $Q_L(x)$ evaluated at the Points in (4.5). Lagrange Interpolation yields Coefficient Form $Q_L(x) = -0.6667x^3 + 3x^2 - 3.3333x + 1$ 28
4.4 FRI-STARKs Construction Steps 30

5.1 $y^2 = x^3 + 7$ 56
5.2 $y^2 = x^3$ 56
5.3 PLONK-based zk-DApp Interaction Process Flow 64
5.4 Initiating the Deployment of PLONK in circom, snarkjs in the zk-DApp Backend 65
5.5 zk-DApp Landing Page 66
5.6 zk-DApp Fill Form (Initial) 67
5.7 zk-DApp Fill Form (Follow-Up after Evaluation) 70
5.8 Zero-Knowledge Data Structure Architecture 72

List of Tables

3.1 Summary of Artifacts and Implementation Approaches 12

4.1 ZKPs Application Poblem Domains and Literature Selection 33
4.2 Complexity Comparison of zk-SNARKs, zk-STARKs, and Bulletproofs . . 43

5.1 Groth16 - Parameter Summary 59

List of Abbreviations

AIR	Algebraic Intermediate Representation
ALI	Algebraic Linking Protocol
APR	Algebraic Placement and Routing
COVID-19	Coronavirus Disease 2019
CRS	Common Reference String
DApp	Decentralized Application
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DLT	Distributed Ledger Technology
EASA	European Union Aviation Safety Agency
FAA	Federal Aviation Administration
FFT	Fast Fourier Transform
FRI	Fast Reed-Solomon Interactive Oracle Proofs of Proximity
FST	Fiat-Shamir Transformation
IATA	International Air Transport Association
IOP	Interactive Oracle Proof
IP	Interactive Proof
KEA	Knowledge of Exponent Assumption
KPI	Key Performance Indicator
LIP	Linear Interactive Proof

MIP	Multi-Prover Interactive Proof
MPC	Multi-Party Ceremony
MRO	Maintenance, Repair, and Overhaul
MSP	Membership Service Provider
NFT	Non-fungible Token
PCP	Probabilistically Checkable Proof
PLONK	Permutations over Lagrange-bases for Oecumenical Non-interactive Arguments of Knowledge
QAP	Quadratic Arithmetic Program
R1CS	Rank-1 Constraint System
ROM	Random Oracle Model
RS	Reed-Solomon
SLR	Systematic Literature Review
SNARK	Succinct, Non-interactive Argument of Knowledge
SRS	Structured Reference String
VC	Verifiable Computation
zk-DApp	Zero-Knowledge Decentralized Application
zk-SNARK	Zero-Knowledge, Succinct, Non-interactive Argument of Knowledge
zk-STARK	Zero-Knowledge, Scalable, Transparent Argument of Knowledge
ZKP	Zero-Knowledge Proof

1 Introduction

A modern civil aircraft consists of approximately three million parts, of which thousands of components must be maintained, repaired, and overhauled (MRO) throughout its life cycle. These MRO events have to be documented. If aircraft and accompanying components demonstrate safe technical conditions, aviation authorities declare them airworthy and allow them to operate. Considering there are 25,000 commercial aircraft in operation and approximately 20,000 suppliers in the industry, hundreds of millions of events must be documented and verified (SITA 2022).

1.1 Motivation

The aviation industry is characterized by high competition and mistrust among the stakeholders (Chatzi 2019), creating conflicting expectations when digitalizing and automating its processes. The aircraft part lifecycle starts with initial manufacturing documentation and continues with buying, selling, leasing, repair, overhaul, and disposal documentation. To this day, back-to-birth documentation of aircraft and components is still largely paper-based and lacks digital process automation (Efthymiou et al. 2022). Transparency and traceability are compromised through disconnected, scattered MRO event documentation entries across various enterprise resource planning systems operated by the industry's participants (SITA 2022). Aircraft components without verified back-to-birth documentation history have no value, i.e., more sustainable secondary parts trading is impossible. This overall situation has an immediate effect on the verification of aircraft parts documentation by aviation authorities: verifying for airworthiness and traceability in the event of an accident is a cumbersome, inquiry-based process, which leaves room for withholding of information and counterfeit parts (Insider Inc. 2019). The research project RAPADO drives digitalization in the aviation industry by creating an industry standard for seamless documentation and verification of aircraft spare parts. One of the research goals is to accumulate knowledge about blockchain technology and investigate its applicability to the MRO industry. Research is conducted by the Information Systems Department at Freie Universität Berlin in

collaboration with project partner Opremic Solutions GmbH.

Current work on blockchain in the MRO industry focuses on empirical analyses of the benefits of the technology in general and the willingness to adopt (Efthymiou et al. 2022). At the same time, further technical understanding and prototypical implementations must be performed. Project-related work resulted in further research into the application of blockchain technology to target the requirements of document storage and traceability, excluding the scope of state-of-the-art cryptography. Previously, a blockchain-based file storage system was proposed, prototypically developed as a decentralized application (DApp), storing aircraft part certificates in a distributed file storage system. Certificate data is traceable through a file hash, which is persisted on-chain. This preliminary work concludes with zero-knowledge proofs as the next step in project-related research (Zedel & Kliewer 2022). Aircraft parts documentation history should always be verifiable and tamper-proof while preserving data confidentiality and protecting the identity and ownership of industry participants (Wickboldt & Kliewer 2019). The aspect of decentralized verification with the requirement of data privacy preservation still needs to be addressed.

Zero-knowledge proofs (ZKPs) can facilitate such a verification process without revealing sensitive information. Due to the possibility of combining confidentiality and transparency, and scaling blockchain throughput, interest in zero-knowledge proofs increased. However, their potential application in the aviation industry needs further assessment. This master thesis focuses on verification and information confidentiality requirements for aircraft spare parts documentation. Suitable use cases and their implementation through zero-knowledge proofs are investigated. The research question is formulated as follows:

How can zero-knowledge proofs be utilized to satisfy requirements and implement identified use cases for the aircraft spare parts documentation?

1.2 Contribution Outline

In this sub-chapter, the following aspects of contribution are outlined. The topic of zero-knowledge proofs is complex and needs to be discussed separately, as concluded in preliminary research. A systematic literature review, based on the methodology in Vom Brocke et al. (2020) and Webster & Watson (2002), surveys current research results and accumulates expertise in the field. As a more practical result, a ZKP protocol, i.e., Groth16, is applied step by step to provide an overview based on a simple polynomial example. Specific details

for understanding the protocol support the mathematical example, representing a reference document as the first artifact of the thesis.

Current research requirements in the project RAPADO are revisited to derive implementation requirements and select use cases. Taking up the use case of MRO data attestation and verification, the developed zero-knowledge decentralized application (zk-DApp) is a proposal to balance transparency and data confidentiality. The zk-DApp is evaluated based on performance aspects and feedback within the project.

For all existing software implementations within the project, high-level assumptions about a digital data structure for aircraft spare part documentation were necessary to continue with prototype development. The question of a practical, blockchain-compatible data structure was not yet addressed and reoccurred as a feedback point during discussions in the project. Motivated by the use case of authenticity checks of spare part certificates and as an evaluation result of the zk-DApp mentioned above, a ZKP and Merkle-tree-based data structure based on Sedlmeir, Völter, & Strüker (2022) is introduced. Concluding with the data structure proposal, a direction of future research is suggested, considering existing software artifacts and results.

1.3 Structure

This master thesis is structured as follows. Chapter 2 summarizes related work, separated into project-related work, zero-knowledge proof topic-related work, and related systems. Chapter 3 describes the execution of the systematic literature review methodology, the artifacts' implementation approach, and summarizes key findings. Chapter 4 presents the outcomes of the systematic literature review, structured from the perspective of historical and design-oriented classification of zero-knowledge proofs. Chapter 5 summarizes implementations and results. First, the requirements are summarized. Second, the three artifacts are described, each satisfying a requirement. The first artifact is the example calculation and step-wise computation of the Groth16 algorithm. The second artifact is the zk-DApp for MRO data attestation and verification. Subsequently, the zk-DApp is evaluated, resulting in the third artifact's design. The third artifact is the fraud-preventive zero-knowledge data structure for data authenticity and integrity verification. Chapter 6 concludes this master thesis by summarizing key findings, describing identified limitations, and providing a future outlook for research.

2 Related Work

Zero-knowledge argument systems are algorithms resulting from specific designs of proof systems combined with cryptographic primitives and mathematical tools. This chapter summarizes content-related and project-related work and systems associated with this master thesis. Revisiting the project RAPADO, current and previous research and implementation efforts at the Department of Information Systems at Freie Universität Berlin are described. The selective systematic literature review in Chapter 5 provides a primary knowledge survey and references for further information. The results of this master thesis are a step-by-step example computation of the Groth16 protocol, a zero-knowledge DApp software artifact, and a data structure conceptual artifact. The related systems utilized for artifact development are presented in this chapter.

2.1 Topic-related Work

In Thaler (2023), an in-depth survey is provided on the history of verifying computation and content classification of probabilistic proofs. Furthermore, more practical argument systems are described, and their composition is studied, while the reader is provisioned with a bird's eye view of the proof systems and cryptographic primitives. First, interactive, multi-prover interactive, interactive oracle, probabilistically checkable proof systems, and variants are described in more detail. Second, concrete example proof systems resulting from historical breakthroughs in the field are highlighted. Lastly, relevant cryptographic primitives and mathematical tools are introduced that enable non-interactivity, efficiency, and zero knowledge. The reader is presented with a taxonomy of succinct non-interactive arguments of knowledge (SNARKs).

The work of Chen et al. (2022) surveys zk-SNARKs from a technical perspective. First, the historical development of zk-SNARKs is described. Second, the first state-of-the-art zk-SNARKs, the Pinocchio protocol, is analyzed in detail and compared to its successor Groth16. Third, two prominent use cases of zk-SNARKs are highlighted, financial and rollups-based applications. Lastly, novel circuits are introduced and applied in private

auctions and decentralized card games, and the implementation code is provisioned. A future research outlook is portrayed by introducing the current research status on zero-knowledge, scalable, transparent arguments of knowledge (zk-STARKs) and recursive zk-SNARKs.

A survey on verifiable computation is provided by Ahmad et al. (2018), focusing on the chronological summary of theoretical and practical advances. Approaches are summarized according to their functionalities and analyzed according to their contributions. The authors comment on open challenges in verifying computation and give an outlook for research efforts.

Zk-SNARKs are the main focus in Nitulescu (2019). First, properties are defined, and tools for designing zk-SNARKs are described. Second, SNARKs from probabilistically checkable proofs, quadratic arithmetic programs, linear interactive, polynomial interactive oracle proof systems, and variants are introduced, highlighting more practical use cases.

2.2 Project-related Work

Project RAPADO is part of the aviation research program in the German Federal Ministry of Economic Affairs and Climate Action, which, among other things, supports research and development of disruptive technologies to be implemented in the aviation industry within the next decade. It is funded by the German Federal Ministry of Economic Affairs and Climate Action and the German Aerospace Center (DLR) within the Federal Aviation Research Programme (LuFo Klima VI-1, European Commission State Aid SA.55829 (2019/N)). This project's primary goal is to research seamless, complete, and safe documentation and certification of aircraft spare parts materials using current developments in blockchain technology. The common practice of reusing, repairing, and trading aircraft spare parts is strengthened and permanently transformed. The MRO industry operates in an error-prone and non-digital manner: If a spare part is repaired and ready to be returned, the corresponding certificates and receipts are sent via mail, i.e., purchasing complex spare parts results in paper-based documentation being delivered on pallets. Hence, MRO providers are not motivated to reuse spare parts and instead buy new ones to prevent risks and liabilities due to incomplete documentation. Spare parts are only used for civil transportation with complete documentation history. The project consortium, consisting of representatives from Freie Universität Berlin and industry partner Opremic Solutions GmbH, aims at creating a digital process to reduce the production of new spare parts, enable trade, and provide increased security through persistent documentation while contributing to the funding target

of productive and efficient aerospace. The industry standard is distributed as open-source, driven by a critical mass of users and the International Air Transportation Association (IATA) as a partner.

Preliminary research at the Department of Information Systems was focused on the architecture of suitable blockchain-based platforms for aviation industry MRO documentation. Wickboldt et al. (2020) proposes a framework to use a private blockchain-based architecture in HyperLedger Fabric, whereby node registration is managed through trusted Membership Service Providers (MSP). This first approach resulted from initial core requirements of data persistence, selective data access, data integrity, and transparency of back-to-birth documentation histories (Wickboldt & Kliewer 2018). Subsequently, these core requirements were further specified through extensive information exchange with previously identified stakeholder groups of airline companies, MRO full-service providers, and MRO parts merchants (Zedel & Kliewer 2022). Further research proposed a public blockchain-based platform with smart contracts and a decentralized file storage system (Zedel & Kliewer 2022). MRO documentation is stored in the decentralized file storage system, which produces a unique file hash. Documents are implemented as a non-fungible token (NFT) in a smart contract.

Following the identified process, the validation status set by aviation authorities is captured. In a DApp software artifact, aviation authorities get full document access using threshold encryption. They receive their key shares from the smart contract and can decrypt using their private key. Combining their key shares allows the corresponding document to be decrypted, accessed, and verified. Access management is separate because various nodes in the network have to combine their key shares. However, once access is granted, all data is exposed. Documents contain competitive information and must be treated confidentially. Zero-knowledge proofs were identified as promising technology to meet data confidentiality requirements while verifying MRO documents on a blockchain-based platform (Zedel & Kliewer 2022). This master thesis extends previous research by focusing on zero-knowledge proofs and applications for sensitive data verification.

The current project status and outlook for the remaining 16 months of the project run time were discussed in the workshop on May 8, 2023, in interaction with the DLR. The following project parties attended: experts from the aviation industry on the business product side (Opremic GmbH), experts from the aviation industry on the technical product side and data privacy aspects (Opremic GmbH), researchers on the topic of mobility systems, blockchain, and zero-knowledge proofs (Department of Information Systems at Freie Universität Berlin).

In workshop discussions, it is suggested that adopting DLT-based solutions for MRO documentation depends on full compliance with requirements set by the operating model in the aviation industry, whereby the main compatibility aspects combine the benefits of data persistence and privacy while capturing spare part documentation. Data persistence describes the constant accessibility of spare part documentation and MRO event history. It is often called tamper-proof data storage and retention in an industry context. Decentralized networks increase data availability, which needs further assessment regarding DLT-based solutions in the project. Data availability is seen as a result of a high number of nodes operated in the network, which also bears the risk of redundancy and efficiency loss.

Further research about off-chain possibilities and the degree of decentralization can help counteract this risk. The research topic of zero-knowledge proofs is identified as an opportunity during the project, whereby this thesis serves as a knowledge accumulation and implementation proposal for the project requirements of data privacy compliance and transparent validation of MRO spare parts. Furthermore, efficient implementation and consolidation into existing prototypes facilitate potential cost reductions and performance advances. The research project requirements are summarized to achieve data integrity, persistence, and transparency. Through achieving this goal, the RAPADO project contributes to creating an industry standard and paradigm change for the aviation business.

2.3 Related Systems

Based on the smart contract for zkDocs created by Andreessen Horowitz a16z (2023), the zk-DApp for MRO data input verification is implemented and described. For this use case, additional operators and a new schema are added, and changes to the frontend are implemented. Chapter 5.3 describes the implementation. In contrast, the non-zero-knowledge DApp implementation developed during the RAPADO semester project uses encrypted and decrypted key shares via the secrets.js library, which are combined to view and validate an MRO certificate for a specific part. The focus of the previously developed DApp is to create a first structure to upload MRO certificates. The validation implementation using only Shamir's secret sharing is vulnerable. The zk-DApp implementation focuses on privacy and transparency requirements for the attestation and verification of specific MRO data and digitization concepts. Requirements are satisfied through examining the technicalities of zero-knowledge proofs and their implementation as per current research. The second prototypical artifact is an architecture proposal for MRO data digitization to preserve data

confidentiality while offering requirement-sufficient transparency via zero-knowledge proofs and Merkle trees. A similar first attempt is presented in Sedlmeir, Völter, & Strüker (2022) for trading green energy certificates.

3 Methodology

The research concept of the thesis is divided as follows. First, a systematic literature review (SLR) will be conducted according to Vom Brocke et al. (2020); Webster & Watson (2002). Second, knowledge from the first part will be applied to the project RAPADO. The implementation part of this master thesis follows an agile development approach. RAPADO use cases for zero-knowledge proof protocols are investigated, conceptualized, and evaluated, considering aspects found in previously examined literature and preliminary work at the Department of Information Systems at Freie Universität Berlin. The application of acquired technical and theoretical knowledge is at focus, while the use cases implemented can be exchanged in the future during further project research.

3.1 Systematic Literature Review

The initial and current status and results of project RAPADO are discussed. From this analysis, potential use cases and design requirements for the application of zero-knowledge proofs are derived. Following the requirement to accumulate knowledge within the project to build expertise in blockchain-based development for the aviation industry, the literature survey focuses on designing zero-knowledge proofs and theoretical foundations. Opportunities and challenges are displayed considering practical examples. The second part of the research concept is implementing a zero-knowledge application for data attestation and verification, focusing on the demonstration of technical mechanisms.

Previous research at the Department of Information Systems concludes with conceptual solutions and a decentralized application for aviation industry documentation, centering storage, and traceability (Zedel & Kliewer 2022). As a result, use cases of uploading, storing, and trading aircraft spare parts certificates were given. This research extends previous findings. However, it takes a new perspective by further investigating possible use cases for ZKPs to automate verification processes, preserve data confidentiality, and suggest suitable data formats. The results are expected to represent the broader research project, i.e., beyond previously used software.

The SLR focuses on zero-knowledge proofs with the scope of classification, opportunities, challenges, evaluation methods, and examples in practice. The design of zero-knowledge proofs needs to be examined from a theoretical perspective. Application domains and use cases applicable to the research project are at focus, which excludes topics of cryptocurrencies and embedded systems. This survey aims to provide an extensive overview of the design and implementation of zero-knowledge proofs and to bring out practical and critical implications. The final search string is derived from a concept map (Figure 3.1), and the selected literature is shown in a concept matrix, with headers displayed in Figure 3.2 and the entire matrix in Appendix 6.2.

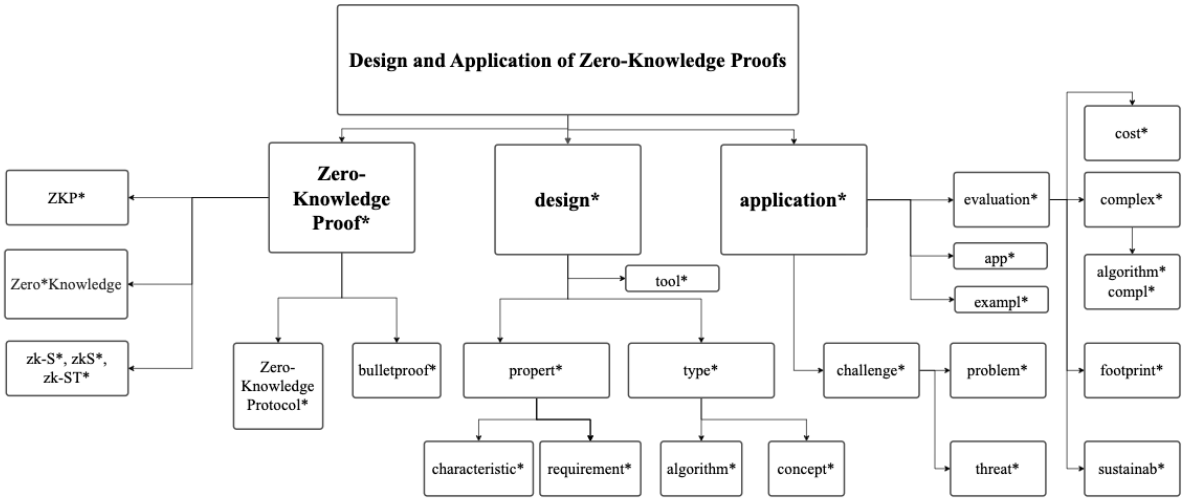


Figure 3.1: Concept Map - Compression of Main Topics and Relations as Basis for Final Search String

The research question concerns topics of computer science, mathematics, and information systems. For the search, the databases Web of Science (WoS), Association of Computing Machinery (ACM), zbMATH, arXiV, Association of Information Systems (AIS), and Institute of Electrical and Electronics Engineers (IEEE) were queried. Articles were searched in Title, Keywords, and Abstract, with date restrictions back to 2018 (WoS) and 2006 (ACM). The following search string was used: (("ZKP*" OR "Zero-Knowledge*" OR "zero*knowledge*algorithm*" OR "zero*knowledge*protocol*" OR "zero*knowledge*proof*" OR "zkS*" OR "zk*SNA*" OR "zk*STA*" OR "zk*STO" OR "bulletproof*") AND ("application*" OR "example*" OR "app*") AND ("carbon*footprint*" OR "*complexity*" OR "evaluation*" OR "cost*" OR "sustainab*" OR "environment*") AND ("challenge*" OR "threat*" OR "problem*")). After deduplication, the search resulted in 580 hits further

condensed according to the following approach. First, the result was filtered by English or German resources with more than or equal to five citations three times or higher past 180 days of usage (560 hits). The next exclusion criteria were applied in Title and Keywords (396 hits), Abstract (319 hits), and Full-text (80 hits), of which backward/forward search added another 20 hits.

Verifiable Computation	Proof Systems and Argument Systems					Mathematical & Cryptographic Tools											zk-SNARKs						Applications						Evaluation				
Survey/Overview	Introduction Theory/Overview	Interactive Proofs	Linear PCPs	Constant-round IOPs	Polynomial IOPs	Complexity Theory	FFT	Elliptic Curves & Pairing	Commitment Schemes	Lagrange Interpolation	R1CS	QSPs and QAPs	Random Oracle Model	Fiat Shamir Transformation	Zero Knowledge	Hashing	MPC	Schwartz-Sippel Lemma	Introduction Theory/Overview	Circuit-specific zk-SNARKs	Universal Setup zk-SNARKs	FRI-STARK	Tools and Libraries	Bulletproofs	Electronic Voting and Government	Electronic Auctions	Data Queries and Traceability	Electronic Healthcare	Cloud Security	Scaling	Cost	Trust	Transparency Challenge	Quantum Computing	Algorithmic Complexity/Performance

Figure 3.2: Concept Matrix - Outcome Summary of the Systematic Literature Review

The results of the survey are categorized as follows.

- Algorithm design topics: Zero-knowledge proofs are designed by combining proof systems and cryptographic means to achieve specific characteristics. Proof systems are examined and defined, linking and presenting cryptographic tools within the scope of building zero-knowledge argument systems in the area of applicable blockchain-based solutions.
- Widely used and performative zero-knowledge proofs are described in more detail, i.e., algorithms of specific zk-SNARKs, zk-STARKs, and Bulletproofs.
- Selective literature is reviewed according to the identified problem domains: Electronic Voting and Government, Electronic Auctions, Data Queries and Traceability, Electronic Healthcare, Cloud Security, and Rollups.
- Opportunities and challenges are discussed and aligned by evaluating and comparing the different algorithms qualitatively through literature review and quantitatively through complexity analysis.

3.2 Implementation and Evaluation

The results of this master thesis are three artifacts, each satisfying a requirement in project RAPADO. The requirements are derived from previous work at the Department of Information Systems at Freie Universität Berlin and workshops held within the project consortium. The development of artifacts is conducted agilely, focusing on the demonstration of the technical application of zero-knowledge proofs via prototyping (Wilde & Hess 2007). Table 3.1 summarizes the three artifacts and their implementation approach.

Table 3.1: Summary of Artifacts and Implementation Approaches

	Artifact	**Short Description**	**Implementation**
1	Groth16 Example Calculation	Step-wise computation of Groth16 protocol by taking a simple polynomial as an example. The goal is to create a reference document to underline zk-SNARKs functionality, corresponding to the requirement of accumulating expertise on the topic within the project.	Developed during the research phase on zk-SNARKs theory and functionality by taking a simple proof example and following the Groth16 protocol steps (Groth 2016).
2	zk-DApp Attestation	Zero-knowledge decentralized application to attest to data of parts and create a trusted, verified process.	Agile development focuses on demonstrating the PLONK zk-SNARKs functionality and practical implementation with circom and snarkjs. First, feedback is collected and implemented.
3	Zero-Knowledge Data Structure	Motivated by recurring comments during workshops and project consortium meetings, this is the first architecture prototype to enable further discussion of effective data formats and digitization of spare parts and corresponding documents.	Analyzing similar efforts in other alternative fields of research (Sedlmeir, Völter, & Strüker 2022), key insights are combined with results from the literature survey to implement first ideas and approaches to utilize zero-knowledge proofs to create effective data formats and standards for the future of aviation.

The algorithms studied in Chapter 4 are evaluated according to their complexity. Artifact 1 represents a reference document to understand zk-SNARK functionalities. Artifact 2 is

evaluated through the first feedback collection from the project partner. Enhancements are implemented and summarized in Chapter 5. Artifact 3 is evaluated by revisiting project and design requirements to reassess current and future project needs.

4 Zero-Knowledge Proofs

This chapter surveys selective literature about zero-knowledge proofs for practical design applications. The goal is to familiarize the reader with the content classification of zero-knowledge proofs in cryptography and to give an introduction and comparison of the central argument systems widely used today.

Zero-knowledge proof systems belong to the domain of verifiable computation (VC) (Šimunić et al. 2021; Ahmad et al. 2018). VC uses cryptographic protocols and arguments to verify that a computation was performed correctly. The introduction of interactive proof systems (IPs) by Goldwasser et al. (1985) and Babai (1985) shows that only correct proofs are valid and malicious proving strategies cannot be verified. Traditional proofs are static and are made for easy, step-wise computation verification, whereas IPs require interaction between the prover and verifier. In computational complexity theory, interactive proof systems are abstract machines exchanging messages between the prover and verifier to convince the verifier that these strings belong to a language L. Here, the formal language L defines a decision problem, i.e., a computational problem with binary classification, yes and no. Examples of sets of strings in L can be that a specific Bitcoin transaction is valid, $x = 7$ for a given $f(x)$, or a particular object belonging to a specific Merkle tree. The untrusted prover has unlimited computational power, and the verifier is honest and computationally restricted.

Advances in computational complexity theory during that time showed that IPs are more efficient and belong to a broader class than the traditional NP, i.e., problems solvable in deterministic polynomial time by reading proof strings of polynomial length (Ben-Sasson et al. 2016). In 1987, it has been shown that every language belonging to NP has zero-knowledge proof systems (Goldreich et al. 1987). Later, it has been proven that the class of IP, i.e., problems solvable by interactive proof systems, lies in PSPACE, i.e., problems solvable in polynomial space (Shamir 1992) and that every language L in polynomial time has an interactive proof system (Lund et al. 1992). The complexity class of IP describes prover and verifier interaction in a polynomial number of rounds. Other important advances in computational complexity theory are MIP=NEXP (Babai et al. 1991) and the PCP

theorem (Arora et al. 1998). These works resulted in a set of proof and argument systems that will be introduced in this chapter. Thaler (2023) gives an exhaustive overview of all systems and in-depth protocol descriptions.

The following examined proof systems are secure against computationally unrestricted provers: Interactive Proofs (IPs), linear Probabilistically Checkable Proofs (PCPs), and Interactive Oracle Proofs (IOPs).

Combining the systems above with cryptographic tools to force specific behavior in the proof generation will create an argument system. Argument systems are considered zero-knowledge if the proof reveals nothing but its validity (Goldwasser et al. 1985). Adding certain properties, e.g., non-interactivity, succinctness, and zero-knowledge, will design a specific argument of knowledge, e.g., zk-SNARK (zero-knowledge succinct non-interactive argument of knowledge). Different zk-SNARKs and notions will be examined in more detail.

The following sub-chapters are structured according to the design-oriented approach described above:

- IPs, PCPs, and IOPs are defined. Through the combination of polynomial commitment schemes or cryptographic tools, argument systems can be designed. Non-interactivity is achieved through the Fiat-Shamir transformation.
- Properties and mathematical tools are introduced to describe different arguments of knowledge: zk-SNARKs, FRI-STARKs, and bulletproofs. Real-world applications of zero-knowledge proof systems are summarized.
- The different argument systems are evaluated according to computational complexity, communication, and security.

4.1 From Proof Systems towards Argument Systems

A mathematical proof in the context of computer science and cryptography is any object that convinces a verifier that a statement is correct. Mathematical proofs embody what is defined in the complexity class NP. A proof system is a structured scheme that decides whether a statement is correct or incorrect. Three properties are desirable for proof systems (Goldwasser et al. 1985), which will be introduced shortly. Throughout this chapter, these properties are revisited more exhaustively.

- The procedure to create and verify proofs should be **efficient and fast**.
- **Completeness**: True statements should have convincing proof of their validity.
- **Soundness**: If a statement is incorrect, there is no possibility of it having convincing proof.

Unlike argument systems, proof systems do not limit the malicious prover in its computational power (statistical soundness). Using cryptographic primitives and restricting the prover, e.g., probabilistic polynomial time proof, so that it cannot break the primitives, describes the design towards argument systems, which are computationally sound (Brassard et al. 1988; Micali 2000). Each proof system presented makes assumptions about the prover. However, only with cryptographic tools and zero knowledge can these proof systems be extended with additional properties to yield zero-knowledge argument systems of various kinds.

4.1.1 Interactive Proofs

In interactive proof systems, the prover with unlimited computational resources interacts with a computationally bound verifier to convince the verifier of the correctness of a statement. The verifier randomly challenges the prover, which is called coin tosses (Goldwasser & Sipser 1986). These challenges happen in rounds until sufficient tests are run, and the verifier is convinced. Goldwasser et al. (1985) defined an interactive proof system that is private, i.e., the verifier's challenges are not publicly accessible, whereas the interactive proof system of Babai (1985) allows the coin tosses to be publicly accessible by the prover.

Interactive Proof System. *L is a language over $\{0,1\}^n$, with n representing the input size domain of n-bit strings. An interactive protocol is an interactive proof system if, after k rounds, the probabilistic verifier in polynomial time exchanged k messages with the computationally unrestricted prover and has to either accept or decline the correctness of the prover's proposition. IP is the complexity class of problems solvable by a k-round interactive proof system.*

The transcript is the order in which messages are exchanged. The prover and verifier are functions $P(x), V(x)$ with common input x. The overall distribution of all transcriptions between the prover and verifier is called $View_V(P(x), V(x))$, which is bound by the number of rounds between P and V. P provides a result that satisfies the proposition, e.g., y to a function $f(x) = y$. P and V exchange a transcript of messages $(m_1, m_2, m_3, ..., m_k)$, whereby both parties take turns, and the prover sends the last message. Note that V is probabilistic

with internal randomness r. Hence, the output depends on (V, x, r, P) and is $\{0, 1\}$, i.e., 1 if the statement is correct, and 0 if it is incorrect (Goldwasser et al. 1985; Babai 1985). Interactive proof systems have completeness error δ_c and soundness error δ_s . For any input x, there must be a convincing proof that $f(x)$ is correct. Incorrect statements for $y \neq f(x)$ cannot result in a convincing proof, i.e., a malicious prover does not exist. IP systems are valid if $\delta_c, \delta_s \leq 1/3$ (Thaler 2023). IPs with k provers are multi-prover interactive proofs (MIPs), introduced by Ben-Or et al. (1988) and extended by Setty (2020) to allow for the design of succinct arguments. MIPs are characterized by no information sharing and non-adaptivity of provers. The probabilistic polynomial time verifier acts similarly as in IPs, but the challenges sent are not shared among the different provers. The prover's response i does not depend on the $i - 1$ interactions in the transcript, i.e., the prover cannot react based on the previous messages, because there is at least another prover p_j whose messages are not known to p_i for $i \neq j$.

The Fiat-Shamir transformation transforms any interactive proof system based on public coin tosses into a non-interactive, publicly verifiable system. This transformation can be described effectively with the use of an ideal cryptographic assumption, the random oracle model (ROM).

Random Oracle Model

The methodology of ROM was introduced in cryptographic theory to satisfy the goal of designing secure cryptographic protocols. First, an ideal system is designed to give all parties access to an oracle, i.e., a random public function. Once the security of the protocol is proven, the random oracle function is replaced by a cryptographic hashing function to implement the system in practice (Bellare & Rogaway 1993). In IPs, the random oracle function is part of the view, i.e., random choices of challenges simulator output. It is assumed that the prover and verifier can query the random oracle function f_R by sending an input x so that the random oracle returns $f_R(x)$. In ROM, the efficiency is dependent on x, i.e., for every input in the problem size domain D, $f_R(x)$ is captured. Since it is only practical in theory, because $|D|$ is large to ensure security, e.g., 2^{256}, hash functions are used in practice (Bellare & Rogaway 1993; Thaler 2023), e.g., POSEIDON or SHA-3. The ROM is controversially discussed in retrospect: On the one hand, it is argued that there exist protocols that are only secure in the ROM, and all implementation efforts in practice have led to insecure protocols, whereby the security of the ideal scheme is not maintained. The area of work at focus is digital signatures and public-key encryption, showing that the

ROM is not sound (Canetti et al. 2004). On the other hand, this conclusion is discussed to indicate there is no evidence of practical security weaknesses if there is a need for random oracle systems in a protocol. Protocol design examples support this statement, e.g., Elliptic Curve Digital Signature Algorithm (ECDSA), leading to cryptographic insecurity if not using a random oracle (Koblitz & Menezes 2015).

Fiat-Shamir Transformation

The Fiat-Shamir transformation (FST) (Fiat & Shamir 1986) removes the need for prover-verifier interaction in a public coin interactive proof system with the help of the random oracle function. The result is a non-interactive, publicly verifiable system. IPs send messages between the prover and verifier, whereby the verifier challenges the prover randomly. Every message the verifier sends in the IP is replaced by values obtained by querying the random oracle. The query always depends on the previous message sent by the prover to prevent soundness attacks.

In summary, the prover only sends one message containing the transcript, i.e., the list of all messages sent by the prover and the query results of the random oracle. The input x has to be appended to the list every round. The verifier's coin tosses are no longer needed, and the verifier does not need to send messages to the prover (Figure 4.1). Argument systems are

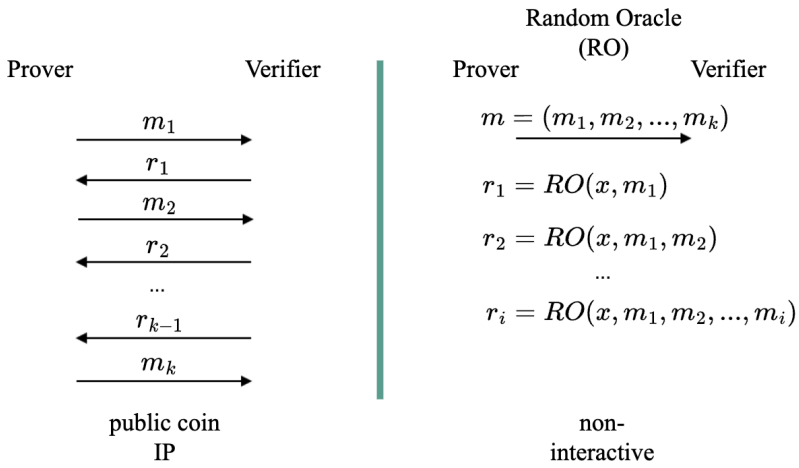


Figure 4.1: Illustration of Generic Fiat Shamir Transformation (based on Thaler (2023))

obtained by applying polynomial commitment schemes to proof systems, while the prover commits to a low-degree polynomial. It allows for polynomial evaluation verification without possessing all the information of this polynomial. Polynomial commitment schemes are used

to prove that a polynomial, evaluated at a specific input, results in a specific output. The prover commits to the polynomial, acting as some object hiding the polynomial, e.g., similar to a hash. The verifier challenges the commitment with a random value. The committer then creates a proof that the polynomial evaluates at that random value at a specific point. The polynomial itself is not revealed. In interactive proof systems with an honest prover running in polynomial time, any arithmetic circuit can be evaluated (Goldwasser et al. 2008). Arithmetic circuits have gates that operate on two types, i.e., addition and multiplication. Besides gates, some wires carry integer variables. Retrospectively, it is a significant achievement since arbitrary computer programs can be expressed via arithmetic circuits, the base layer of many zero-knowledge protocols today. More mathematical tools for understanding the functionality of argument systems are covered in Chapter 4.2.

Zero-Knowledge

If the verifier knows nothing about the statement and the witness but its validity, the algorithm will run with zero knowledge. This property can be described by using a simulator notion (Thaler 2023): A prover and probabilistic polynomial time verifier strategy exists under the honest verifier assumption. For any probabilistic polynomial time verifier strategy, there is a probabilistic polynomial time algorithm (simulator) that can depend on the verifier strategy, which produces outputs that are indistinguishable from the distribution of transcripts generated through the prover and verifier strategy interactions. The simulator ensures that the verifier only learns that $x \in L$. Under the honest verifier assumption, there are at least six types of zero-knowledge protocols, whereby the notion of simulator output and transcript distribution being indistinguishable is at focus (Birrell & Vadhan 2009):

- **Perfect Zero-Knowledge**: The output of the simulator $S(x)$ and the distribution of the transcript created through prover and verifier strategy interaction $View_V(P(x), V(x))$ within the protocol is identical.
- **Statistical Zero-Knowledge**: $S(x)$ and $View_V(P(x), V(x))$ have a negligible statistical distance, given a polynomial number of samples from the distributions.
- **Computational Zero-Knowledge**: $S(x)$ and $View_V(P(x), V(x))$ can be distinguished with negligible probability only if a polynomial number of samples is given.

Soundness is divided into statistical (for proof systems) and computational (for argument systems) soundness. Argument systems are computationally sound, i.e., allow for applying cryptographic primitives. The computationally bound prover cannot break these primitives,

unlike the unbound prover in proof systems, which are statistically sound (Thaler 2023). The verifier learns that a statement is true and has to be convinced that the prover knows a correct witness. The simulator is able to extract that witness information through interaction with the algorithm (knowledge soundness) (Nitulescu 2019).

4.1.2 Linear Probabilistically Checkable Proofs

In probabilistically checkable proofs (PCPs), the prover does not need to answer queries based on the current or previous query content posted by the verifier. The polynomial time verifier is provided with oracle access to a static proof string π, whereby the proof is queried, and the result of it only depends on the currently processed query (Fortnow et al. 1994). The breakthrough of robust, pairing-based schemes is attributable to Gennaro et al. (2012), who introduced Quadratic Arithmetic Programs (QAPs) as a variant of quadratic span programs. Quadratic span programs are efficient because they only satisfy boolean circuits. Therefore, introducing QAPs is essential to represent more practical computations, e.g., multiplication gates, whereby the efficiency lies in the usability for effectively solving more natural problems. Any arithmetic circuit instance can be transformed into instances of a rank-1 constraint system (R1CS).

R1CS. *A R1CS is an intermediate representation of the computational problem, which is used to perform the application of argument systems. Given a set of $n \times m$ matrices A, B, C, with values derived from a finite field $\mathbb{F}$. An R1CS instance is called satisfiable if there exists a solution vector $z \in \mathbb{F}^n$ with $z_1 = 1$, so that*

$$(A \cdot z) \circ (B \cdot z) = C \cdot z.$$

For every ith row of each of the three matrices belonging to the finite field circuit size, the following equation must hold:

$$\langle a_i, z \rangle \cdot \langle b_i, z \rangle - \langle c_i, z \rangle = 0 \tag{4.1}$$

In practice, z is known by the prover. The solution vector also has a public input, namely the computation result (refer to Chapter 5.2 for an example calculations). The goal is to arrive at a univariate polynomial $t(x)$ when divided by the minimal polynomial, represents a secret polynomial $h(x)$ without remainder. The minimal polynomial is always known if

the number of constraints is known and is a multiple of $t(x)$. The linear PCP is evaluated in linear, constant time. The QAP is obtained by taking each value of each row of the R1CS matrices as an output of a polynomial, which is to be calculated. The polynomial results to that specific value in the R1CS, when evaluated at $X = \{1, 2, ..., \text{number of constraints}\}$. Given these constraints, the three sets of polynomials are calculated via the sum of Lagrange Interpolation.

Lagrange Interpolation. *The polynomial obtained through Lagrange Interpolation is the polynomial $P(x)$ of degree at most $\leq (n-1)$ and passes through the n points $((x_1, y_1 = f(x_1)), (x_2, y_2 = f(x_2), ..., (x_n, y_n = f(x_n))$, that are given. It is denoted by*

$$P(x) = \sum_{j=1}^{n} P_j(x) \tag{4.2}$$

$$P_j(x) = y_j * \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

Example. *Let $n = 2$ with the three given points $[1, 0], [2, 1], [3, 0]$. The polynomial obtained through Lagrange Interpolation is:*

$$\begin{aligned} P(x) = \sum_{j=1}^{2} \left(y_j * \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} \right) &= 0 * \frac{(x-2)(x-3)}{(1-2)(1-3)} + 1 * \frac{(x-1)(x-3)}{(2-1)(2-3)} \\ &= 0 * \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{x^2 - 4x + 3}{-1} \\ &= -x^2 + 4x - 3 \end{aligned}$$

Each gate is the x value, and the values of the R1CS matrix column are the corresponding y values for the computation to obtain the QAP. It consists of three sets of polynomials (Chapter 5.2). Each interpolation calculation receives n points (number of constraints), and the result is a polynomial of degree $n - 1$. Multiplying each polynomial in the matrices

with the solution vector yields the final polynomials $A(x), B(x), C(x)$ and

$$\frac{A(x) * B(x) - C_(x)}{Z(x)} = H(x). \tag{4.3}$$

In a linear PCP, the proof consists of evaluations of linear functions created by the prover. Soundness is guaranteed against a dishonest prover by ensuring the proofs rely on a specific structure. The verifier is querying the proof by transforming linear PCPs into non-interactive argument systems. The trusted set-up issues proving and verifying keys, which are used to perform checks using pairing-based cryptography. Many systems rely on this feature, e.g., Groth16, whose mathematical tools and general protocol procedures are presented in Chapter 4.2.1.

4.1.3 Interactive Oracle Proofs

Interactive oracle proofs (IOPs) can generalize PCPs and IPs, which decreases the large proof size, e.g., resulting from PCPs. Introduced by Ben-Sasson et al. (2016), IOPs are interactive proofs with a verifier receiving query access to the proof in every round instead of the full proof size. The verifier chooses the element of the message in every round and only pays the cost of the query, which results in a verifier time that is sub-linear to the total proof length. IOPs were made non-interactive by introducing Merkle hashing and the FST in the random oracle model (Ben-Sasson et al. 2016). The prover message is not sent in every round. Instead, the Merkle commitment of the prover's message is sent. The verifier determines which element of the message will be queried through simulation. The prover reveals the relevant element by provisioning the verifier with authentication paths in the Merkle tree. The interactive argument is then transformed into a non-interactive argument via FST, while soundness is preserved (Thaler 2023).

SNARKs from constant-round IOPs are slower than IP or MIP-based IOPs because constant-round IOPs require provers to commit to many polynomials, while in IP and MIP-based argument systems, provers commit only to a single polynomial. Also, the single polynomial is not larger than any of the many polynomials in constant-round IOPs (Reingold et al. 2016). IOPs derived from IPs and MIPs need recasting to effectively compare the different IOPs because classic IPs and MIPs use multivariate polynomials. In standard IOPs, each message is sent as a string, and the verifier has to query each string for a specific element. In polynomial IOPs, each message is a polynomial over a finite field $\mathbb{F}$ with a degree at most some upper bound specified. Polynomial IOPs for R1CS-satisfiability use univariate

polynomials. These polynomials can have many coefficients, sometimes the size of the R1CS. If the verifier had read access to the full polynomial description, verifier time would explode to check the proofs (Bünz et al. 2019). The verifier chooses any input for query access to its evaluation in the specified polynomial. Polynomial commitment schemes are used to obtain succinct arguments so the verifier can confirm that the specified input evaluates to the output received, i.e., that a specific polynomial with a certain degree is correctly specified. Replacing each message and associated evaluation queries with polynomial commitment schemes will make the protocol standard IOP. The transformation steps in Ben-Sasson et al. (2016) will yield succinct arguments for further ZKP design. All SNARKs, except for linear-PCP-based, are designed according to this approach, e.g., the Fast Reed-Solomon Interactive Oracle Proofs of Proximity (FRI) IOP, an IOP-based polynomial commitment scheme with polylogarithmic proof length (Chapter 4.2.3).

The proof systems introduced in this chapter form the base layer of zero-knowledge succinct, non-interactive arguments of knowledge. The proof is convincing, and the witness is valid. At the same time, the verifier learns nothing about it except its validity; hence cannot be convinced by an invalid witness (completeness, soundness, and zero knowledge).

4.2 Zero-Knowledge Succinct, Non-interactive Arguments of Knowledge

Following the first zero-knowledge IPs (Goldwasser et al. 1985), Blum et al. (1991) introduce the common reference string (CRS) and create non-interactive zero-knowledge proofs with only one message to be shared, i.e., the proof. Subsequently, research on reducing the proof size increased, Micali (2000) introduces the first non-interactive zero-knowledge proof with sublinear proof size. Building on these findings, the first zk-SNARKs for circuit satisfiability uses proofs that are constant-size and uses pairings to efficiently check the polynomial equations without revealing any information, e.g., coefficients (Groth 2010). Gennaro et al. (2012) continue increasing the efficiency of polynomial equation verification and introducing QAPs. Parno et al. (2016) makes use of it and develops the Pinocchio protocol, which utilizes QAPs and the knowledge of coefficients to efficiently verify polynomial equations through eight pairing checks without revealing any decodings. This protocol is enhanced by Groth (2016), reducing the proof size again and decreasing the number of pairing checks to three. It is widely used today, and the characteristics of succinctness and non-interactivity make it particularly useful in blockchain, e.g., in Zcash and circom (Chen et al. 2022). In

the following, different zk-SNARK designs will be introduced.

4.2.1 Circuit-specific zk-SNARK from QAP

With the introduction of QAP by Gennaro et al. (2012) as starting point, Groth (2016) introduced essential and performative zk-SNARKs for circuit satisfiability. The linear interactive proof (LIP provided reduces the number of verifier queries to 3 and the number of group elements from 10 to 6. Previous security relied on the knowledge of exponent assumption, which was enhanced by establishing security in the generic group model. Practically, the Fast Fourier Transform (FFT) is utilized. Changing the circuit to a small degree still requires a complete restart of the trusted set-up (Thaler 2023). The following introduces preliminary assumptions and definitions for designing circuit-specific zk-SNARKs from QAP, providing a starting point for the functionalities in the Groth16 protocol applied in Chapter 5.2.

Linear Interactive Proof (LIP). *While linear PCPs ensure that the prover answers any verifier query through a linear function, LIPs ensure soundness against provers that do not use the same linear function to answer queries. Any linear PCP can be transformed to LIP (Bitansky et al. 2022). The Knowledge of Exponent Assumption (KEA) guarantees, under the hard discrete logarithm problem, that the prover is knowledgeable about these linear functions, i.e., can prove that the same coefficients are used for all polynomials resulting from the QAP.*

SNARKs from QAP consist of a generation algorithm, prover, and verifier (Groth 2016; Guo et al. 2022; Benamara 2022), introduced informally. The generation algorithm executes the trusted set-up by taking a random security parameter and the arithmetic circuit as input. The QAP is generated over a finite field, which forms the basis for the trusted set-up run. With secret states and parameters not to be known by anyone and deleted after (toxic waste), the CRS is created as an output of the generation algorithm (for a detailed structure, see Chapter 5.2). The prover algorithm uses the CRS and public statements to compute a witness so that the target polynomial, i.e., the result of encoded computation from QAP, and the minimal polynomial vanish on some quotient polynomial. After it has been ensured, the proof is generated. The proof consists of elements from the encoded computation generated by two group elements. The verifier algorithm uses the proof and performs pairing checks to output either a true or false. In Groth16, the pairing checks are minimal, and the parameters and generated values are mainly taken from the proving and

verification keys. Perfect completeness is achieved if the prover knows a true statement and an honest verifier will be convinced (Guo et al. 2022). Zero-knowledge is achieved by randomizing the polynomials and uniformly distributing the proof terms (Groth 2016, 2010).

The combination of linear interactive proofs with pairing-based cryptography is beneficial due to the functionality of bilinear maps (Chapter 5.2). Through the use of bilinear maps, one homomorphic multiplication operation can be executed:

Bilinear Maps and Multiplicative Homomorphism. *Given the commitments a_1, a_2, a_3 and the values b_1, b_2, b_3 in the commitments, and the bilinearity of the following map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$, $e(g^{b_1}, g^{b_2}) = e(g^{b_3}, g)$, if and only if $b_3 = b_1 * b_2$.*

In practice, $\mathbb{G}$ is an elliptic curve group defined over a prime finite field $\mathbb{F}_p$, and $\mathbb{G}_t$ is a subgroup of the extension field of $\mathbb{F}_p$, namely $\mathbb{F}_{p^k}$. k is a positive integer and defines the embedding degree of the elliptic curve group, and must be low to apply pairings efficiently. If k is too large, the mapped elements to $\mathbb{G}_t$ will be more expensive to operate on (Thaler 2023).

Decisional Diffie-Helman Assumption (DDH). *Given a cyclic group $\mathbb{G}$ with generator g and g^a, g^b for a, b, chosen uniformly and independently from $\mathbb{G}$, g^{ab} is computationally indistinguishable from a random group element of the cyclic group.*

Instead of KEA, the FFT computes the Decisional Diffie-Helman problem in $O(nlogn)$. It is achieved by making use of the roots of unity. The element w is a root of unity in finite field $\mathbb{F}$, whereby $w^n = 1$ with n being the n^{th} primitive root of unity for all positive integers s smaller than n, $w^s \neq 1$. Instead of evaluating a polynomial at n point pairs $\{(x_0, y_0), (x_1, y_1), ..., (x_{n-1}, y_{n-1})\}$, the values $1, w, w^2, ..., w^{n-1}$ are used, halving the values to $\frac{n}{2}$ (Groth 2016).

4.2.2 Permutation Argument zk-SNARK

Permutations over Lagrange-bases for oecumenical non-interactive arguments of knowledge (PLONK) is a zk-SNARK with a universal trusted set-up, which produces a structured reference string (SRS). The SRS is of size d, used for circuits of up to $\leq d$ gates. This universal SNARK has fully succinct verification and low prover run time (Gabizon et al. 2019), compared to its predecessor Sonic (Maller et al. 2019), which was the first universal and fully succinct SNARK. PLONK is based on constant-round polynomial IOP and uses

the polynomial commitment scheme based on Kate et al. (2010). It is presented as a non-interactive protocol obtained through Fiat-Shamir transformation (Gabizon et al. 2019). The trusted set-up is universal and updatable: it can be used for the entire scheme and does not have to be produced for every problem (circuit), e.g., in Groth16. Also, the Kate commitment scheme can be replaced by any other polynomial commitment scheme, e.g., FRI (Chapter 4.2.3). Kate commitments use the elliptic curve generated points published in the public key after trusted set-up, similar to Groth16 (Chapter 5.1), to commit to a polynomial of degree d. The first $d+1$ points are used to evaluate the polynomial at the respective coefficient. The underlying assumptions can be attributed to the Schwartz-Zippel lemma.

Schwartz-Zippel Lemma. *Let $f(x)$ be a non-zero polynomial with degree d over $\mathbb{F}^n$, then, for a randomly chosen r, the probability of $f(r) = 0$ is at most $\frac{d}{n}$.*

The Schwartz-Zippel lemma proves that the polynomial evaluates to 0 at any point with high probability if it evaluates to 0 at a given random r. If two polynomials evaluate equally at r, they are equal at every point with high probability. In a polynomial commitment scheme, this suggestion is beneficial. The prover evaluates the polynomial at the random r chosen by the verifier and sends it along with a proof. If the proof is valid, the verifier concludes that the result of the prover is also valid (Kate et al. 2010).

The computation is first converted into an arithmetic circuit. Then, the arithmetic circuit is used to obtain a constraint system similar to the R1CS from the previous chapter. Both have only one multiplication allowed per gate. However, if it is not a constant, PLONK only allows for one addition per gate. This constraint system also comes with copy constraints, transforming the system into polynomials. The verification uses a polynomial commitment scheme (Figure 4.2).

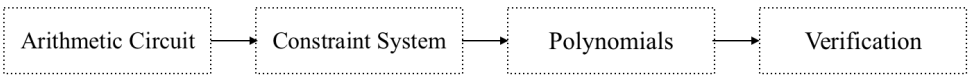


Figure 4.2: PLONK Construction Steps

Like circuit-specific zk-SNARKs, e.g., Groth16, the computation must be flattened for the protocol to process it. The problem is represented in an arithmetic circuit consisting of gates that can represent an addition or a multiplication. Then, the arithmetic circuit is transformed into a constraint system representing the circuit's wires. In analogy to

circuit-specific zk-SNARKs, the constraint system depends on the number of gates (Chapter 5.2). In PLONK, the constraint system is normalized into a specific form, which will be presented shortly, and is referred to as part of the public input in Gabizon et al. (2019). It is distinguished between constraints per gate and across gates within arithmetic circuits. The formalization system of gate constraints is the following:

$$Q_L a + Q_R b + Q_O c + Q_M ab + Q_C = 0 \tag{4.4}$$

L, R, O represent the left, right, and output gate wires. M, C stand for multiplication and constant. This standardized form allows the representation of addition and multiplication. Setting $Q_M ab = 0,\ Q_C = 0,\ Q_O c = -1$ and the rest to 1 will represent $a + b - c = 0$. Setting $Q_L a = 1,\ Q_L b = 1,\ Q_O c = -1$ and the rest to 0 will represent multiplication. Each gate is represented in the form presented in 5.1. Similar to the R1CS previously described, $Q_L a, Q_L b, Q_O c, Q_M ab, Q_C$ can be expressed as vectors that hold the circuit structure. Again, in analogy with the R1CS, a, b, c can also be expressed as vectors, which are the witness assignments. In correlation to Chapter 4.2.1 on circuit-specific zk-SNARKs, these witness assignments might be private and only known to the prover. In this constraint system, there are vectors $Q_L, Q_O, Q_M, Q_C, a, b, c$. Using the indices of these vectors as x, they can be transformed into polynomials of evaluation format. For example, let us define one of the vectors in a circuit with 4 gates to illustrate the procedure.

$$Q_L = (1, 0, 1, 0) \text{ converts into the set of points } (0, 1), (1, 0), (2, 1), (3, 0). \tag{4.5}$$

The set of points matches a degree 3 polynomial, which can be shown in a coordinate system (Figure 4.3). Through Lagrange Interpolation, the concrete polynomial in coefficient form can be calculated.

This procedure is applied to all Q-vectors and the vectors $a,\ b$, and c to transform them from constants into polynomials. The corresponding function obtained is

$$f(x) = Q_L(x)a(x) + Q_R(x)b(x) + Q_O(x)c(x) + Q_M(x)a(x)b(x) + Q_C(x) = 0 \tag{4.6}$$

$$f(x) = Z(x)H(x) \tag{4.7}$$

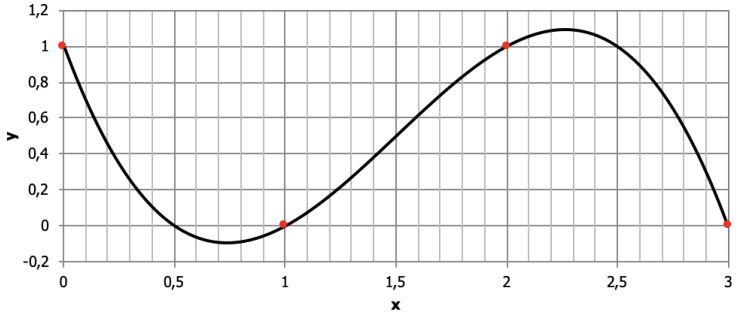


Figure 4.3: Example Polynomial $Q_L(x)$ evaluated at the Points in (4.5). Lagrange Interpolation yields Coefficient Form $Q_L(x) = -0.6667x^3 + 3x^2 - 3.3333x + 1$

Now, as much information as possible can be compressed into a single source, i.e., a polynomial.

Besides the constraints as per gate, some constraints hold across gates, e.g., the output of a gate can be equal to the input on another gate. It is necessary to represent them, too, in order to translate the whole problem into the scheme. These copy constraints can lie within one vector, e.g., if $b0 = b2$, or between multiple vectors, e.g., $c0 = a3 = b1$. In the first case, the indices of the vectors are exchanged to create a permutation function $\sigma(i)$. In the second case, the vectors are combined into one long vector before obtaining the permutation function. Once all copy constraints are generated as lists of permuted indices, they are transformed into polynomials of the permuted gate indices. The resulting polynomials are

$$\sigma_a(x), \sigma_b(x), \sigma_c(x) \tag{4.8}$$

A proof is generated by using these permutations and computing the values of all the gates to create the polynomials of $a(x), b(x), c(x)$. Now, an accumulator $p(x)$ will be designed in order to represent all coordinates in the set of points between the vectors a, b, c. It proves the copy constraints as follows.

$$p(x+1) = p(x) * (v_1 + X(x) + v_2 * Y(X)) \tag{4.9}$$

v_1, v_2 are random values and X, Y are polynomials per vector representing the x and y coordinates. For every copy constraint, e.g., $a1 = a3$ or $b4 = c1$, there will be a $X(x)$ representing the x coordinates a, b, c. $X'(x)$ will be a polynomial representing the index flips in the copy constraint. Every vector will be represented in $p_a(n), p_b(n), p_c(n)$ and $p'_a(n), p'_b(n), p'_c(n)$. Instead of checking each copy constraint within the same vector

individually, the following multiplication check is also performed to check copy constraints across vectors at the same time:

$$p_a(n) * p_b(n) * p_c(n) = p'_a(n) * p'_b(n) * p'_c(n) \tag{4.10}$$

In practice, the permutation accumulators are not expressed in dependency of vector size n but through high-order roots of unity, whereby the field elements satisfy $\omega^n = 1$. Also, all values are expressed as elements within a finite field of prime order p, similar to the circuit-specific zk-SNARKs. The X coordinates are not expressed as indices dependent on vector size n but with ω and some random element g in the field.

Summary

After transforming all the constraints into sets of polynomials, the following checks need to be performed to verify the proof (Gabizon et al. 2019; Buterin 2019; Chen et al. 2022). These checks can be verified through the polynomial commitment scheme based on (Kate et al. 2010). Elliptic curve points are generated randomly, similar to circuit-based zk-SNARKs, and used to evaluate the polynomials. Elliptic curve pairings allow checking whether the equations hold without revealing any generated points or polynomials. The equation in (4.3) is the main equation of the circuit that needs to be checked. Then, six permutation accumulator functions are used for the witness assignment vectors and their copy constraints. The equation check in (4.4) results in six rounds:

$$P_a(\omega x) - P_a(x)(v_1 + x + v_2 a(x)) = Z(x)H_1(x) \tag{4.11}$$
$$P_{a'}(\omega x) - P_{a'}(x)(v_1 + \sigma_a(x) + v_2 a(x)) = Z(x)H_2(x) \tag{4.12}$$
$$P_b(\omega x) - P_b(x)(v_1 + gx + v_2 b(x)) = Z(x)H_3(x) \tag{4.13}$$
$$P_{b'}(\omega x) - P_{b'}(x)(v_1 + \sigma_b(x) + v_2 b(x)) = Z(x)H_4(x) \tag{4.14}$$
$$P_c(\omega x) - P_c(x)(v_1 + g^2 x + v_2 c(x)) = Z(x)H_5(x) \tag{4.15}$$
$$P_{c'}(\omega x) - P_{c'}(x)(v_1 + \sigma_c(x) + v_2 c(x)) = Z(x)H_6(x) \tag{4.16}$$

There are constraints for the accumulator to be checked, which result from (4.7):

$$P_a(1) = P_b(1) = P_c(1) = P_{a'}(1) = P_{b'}(1) = P_{c'}(1) = 1 \tag{4.17}$$
$$P_a(\omega^n)P_b(\omega^n)P_c(\omega^n) = P_{a'}(\omega^n)P_{b'}(\omega^n)P_{c'}(\omega^n) \tag{4.18}$$

The only program-specific polynomials that must be computed upfront are the Q-polynomials from the circuit and the σ-permutation polynomials. The verifier algorithm only stores commitments to these polynomials. The user inputs are the witness assignments $a(x), b(x), c(x)$, the accumulators P from above, and the different H for every round. Although the verification is efficient, the proof size is still an area of improvement. For the implementation of PLONK in circom and snarkjs, see Chapter 5.3.

4.2.3 FRI-STARKs

This chapter describes the general functionalities and properties of zk-STARK and FRI. There are various options to parameterize and design FRI-STARKs with different properties, e.g., concerning the presence of non-interactivity, soundness, succinctness, and zero-knowledge. Hence, FRI-STARKs are perceived as a group of transparent argument systems (Ben-Sasson et al. 2018). In FRI-STARKs, a computation statement is represented using an algebraic intermediate representation (AIR) and then transformed into a set of Reed-Solomon (RS) proximity problems, which can be efficiently verified using the FRI protocol. FRI-STARKs consist of two main parts, i.e., an algebraic linking protocol (ALI) and a Fast Reed-Solomon proximity testing protocol (FRI). The ALI is the last step of the algebraic intermediate representation of the computational problem. The FRI protocol is an effective proximity test method to conclude that a set of points mostly lies on a polynomial with a degree less than a certain value provided, achieving linear proof complexity and logarithmic verification complexity (see Table 4.2). The FRI-STARKs construction involves

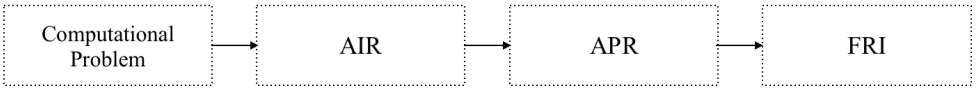


Figure 4.4: FRI-STARKs Construction Steps

several steps. The first step is the (AIR), which represents a computation statement as a pair (instance, witness) $= (x_{AIR}, w_{AIR})$. The instance x_{AIR} defines the algebraic constraints of the problem, represented by a set of polynomials $P = (P1(X,Y), P2(X,Y), ..., Ps(X,Y))$, where X represents the current state and Y represents the next state of the computation. The algebraic execution trace gives the witness w_{AIR} and must satisfy both the boundary and transition constraints defined by the polynomials. The next step in the construction of the FRI-STARKs is algebraic placement and routing (APR). APR arithmetizes the computational statement further by mapping the instance and witness (x_{AIR}, w_{AIR}) to a

set of RS proximity problems. The instance reduction is performed on the verifier's side, while the prover performs the witness reduction. The instance x_{APR} is characterized by various parameters, including a binary field $\mathbb{F}$, a set of indices T, mappings corresponding to transition constraints, witness evaluation space and composition evaluation space, and rates. The witness w_{APR} is a sequence of functions evaluated on the witness evaluation space and must satisfy specific RS membership conditions. The final step in FRI-STARKs is the algebraic linking IOP (ALI), which transforms the verification of functions belonging to RS codes into RS proximity problem instances that can be solved using the FRI protocol. Non-interactivity is achieved through applying the FST. The ALI protocol ensures perfect completeness, soundness error $< 1/|\mathbb{F}|$, and knowledge extraction. If certain conditions related to the random sampling of functions are satisfied, ALI achieves the zero-knowledge property.

Overall, FRI-STARKs provide an efficient and secure way to verify computation statements using algebraic representations, Reed-Solomon codes, and interactive protocols. The construction of FRI-STARKs provisions the system with perfect completeness, zero knowledge, and good soundness. The only cryptographic assumptions are public randomness and the existence of a one-way hash function. These assumptions remove the need to use not quantum-resistant elliptic curves and a trusted setup. Improvements of FRI-STARKs are referred to as DEEP-APR and DEEP-FRI (Ben-Sasson et al. 2019). The construction is future-proof and enables transparent and non-interactive arguments of knowledge, making it suitable for various cryptographic applications.

4.2.4 Bulletproofs

The third IOP-based SNARKs make use of discrete logarithmic polynomial commitment approaches. The discrete logarithm problem is a one-way computation making it infeasible for non-quantum computers to compute any input given only the output. Introduced by Bünz et al. (2018), bulletproofs were designed to address the limitations of earlier ZKP systems, e.g., non-interactive zero-knowledge proofs and zk-SNARKs. The main advantages of bulletproofs are efficiency and scalability. They offer logarithmic-sized proofs, i.e., the proof size grows logarithmically with the number of inputs, making them smaller than previously introduced SNARKs. Bulletproofs are suitable for a wide range of applications, including confidential transactions, anonymous credentials, and secure multi-party computation.

Bulletproofs rely on a cryptographic primitive called a Pedersen commitment, which allows

for efficient range proofs. These commitments allow for zero-knowledge verification by only revealing that the sum of inputs is greater than the sum of outputs without disclosing underlying values (Deng et al. 2022). A range proof proves that a secret value lies within a specific range without revealing the exact value (Chung et al. 2022). Bulletproofs leverage these commitments to construct efficient and secure zero-knowledge proofs. The interactive protocol operates in multiple rounds, resembling the FRI polynomial commitment scheme. In each round, the verifier sends a random field element to the prover, which is used to halve the length of the committed vector. After multiple rounds, the prover claims to know a vector that satisfies an inner product relationship, and its length is reduced to 1 (Thaler 2023; Godden et al. 2022). At this point, the prover can send the vector to the verifier as proof. The protocol achieves zero knowledge by sending Pedersen commitments instead of the vectors. Non-interactivity is realized via FST. No trusted setup is required; however, verifying time is more consuming than in the other SNARKs introduced (see Table 4.2).

Bulletproofs have gained significant attention in cryptography due to their practicality, efficiency, and strong security guarantees. Unlike regular range proofs, which grow linearly with the number of transactions, this type of range proof exhibits logarithmic growth relative to the verified data size. As a result, bulletproofs are more succinct and smaller in size.

Chapter 4.1 and 4.2 survey a selective SNARKs taxonomy from a design perspective. Except for linear-PCP-based SNARKs, all SNARKs covered are obtained by combining IPs, MIPs, or constant-round IOPs with polynomial commitment schemes. IPs and MIPs require a combination for multilinear polynomials, while constant-round IOPs need one for univariate polynomials. SNARKs using IOP-based polynomial commitment schemes are quantum-secure (Thaler 2023).

4.3 Application Domains and Use Cases

Zero-knowledge proofs have found broader implementation due to increasing interest in decentralized applications in recent years. However, despite successful research efforts in distributed ledger technology, there is only limited adoption of blockchain-based solutions outside the financial sector. One of the main challenges, pointed out by Sedlmeir, Lautenschlager, et al. (2022), is the need to make sensitive data visible. The high transparency of blockchain collides with preserving privacy and allowing for restricted visibility. Godden et al. (2022) describe it as increasing consciousness to preserve data confidentiality and ownership, which leads to the development of privacy-enhancing technologies. The literature

review (SLR approach described in Chapter 3) shows that general-purpose zero-knowledge proofs find essential implementation in domains with enhanced privacy-preserving efforts. ZKP implementations can be categorized into the following application domains: identity management, data sharing and traceability, and system scaling (Zhang et al. 2021; Chen et al. 2022; Morais et al. 2019). Use cases from these application domains are clustered according to the problem domain they are attempting to solve (Table 4.1): (1) Electronic Voting and Government, (2) Electronic Auctions, (3) Data Queries and Traceability, (4) Electronic Healthcare, (5) Cloud Security, and (6) Rollups.

Table 4.1: ZKPs Application Poblem Domains and Literature Selection

	Problem Domain	**Literature**
1	Electronic Voting and Government	Bansod & Ragha (2022); Guo et al. (2022); Querejeta-Azurmendi et al. (2020)
2	Electronic Auctions	Li & Xue (2021); D. Wang et al. (2021)
3	Data Queries and Traceability	Godden et al. (2022); Xue & Wang (2022)
4	Electronic Healthcare	Luong & Park (2022); Zheng et al. (2022); H. Wang et al. (2022); Huang et al. (2020)
5	Cloud Security	W. Liu et al. (2019); Major et al. (2020); Munivel & Kannammal (2019); Kanagamani & Karuppiah (2021)
6	Rollups	Chen et al. (2022); Ethereum Foundation (2023a); Matter Labs (2022); Buterin (2021)

4.3.1 Electronic Voting and Government

Motivated by recent developments in personal data protection laws, Bansod & Ragha (2022) present a governmental architecture based on self-sovereign identity to cope with the increasing demand to protect personal online information transactions. In this generalized scenario, users request a decentralized identifier from their digital governmental issuer, which

will ask for personally identifiable information needed for administrative services, e.g., birth date, address, and tax identifier. The user only provides a zero-knowledge-proof-enabled identity, e.g., the proof of being the person the user claims to be, and receives identity certificates from the e-government issuer. These certificates are encrypted and stored in a database and on-chain. When in need for a service, e.g., renewing a driver's license, the user can request it from service providers and be provisioned with the required certificates. The service provider can verify the certifications' hash values and digital signatures and provide the service once it holds.

Current privacy-preserving efforts in governmental biometric identification did not leave traditional cryptography yet. Guo et al. (2022) propose a novel approach to decrease governmental expenses and enable the scaling of the systems. A zk-SNARKs-based approach is presented, which reduces efficiency and eliminates fingerprint template disclosure. First, the fingerprint features are extracted by calculating the Euclidean distance of points collected. If the sizes of the distances match a certain threshold, the authentication is passed. This computation is being converted into a zk-SNARK friendly, polynomial computation with specific constraints that resulted from the first step. Then, this computation is transferred into a circuit, then to an R1CS, and later into a QAP with three polynomial matrices. The trusted set-up generates proving and verifying keys via CRS. The prover algorithm creates the proof with proving key and private witness, and the verifier algorithm can perform the verification through elliptic curve pairing. The Groth16 algorithm was used to apply it to the biometric authentication example, which leaves improvement for disposing of the trusted set-up requirement and QAP computational time.

There are various implementations of electronic voting schemes using zero-knowledge proofs. Querejeta-Azurmendi et al. (2020) introduce a first verifiable re-voting scheme with linear complexity. Voters are authenticated and can vote multiple times from any device with zero knowledge. Only the latest vote is counted. Adversaries can only obtain further information about the vote as the final result used for the election, even if they possess unbounded computational power. In the pre-election phase, the voting server generates voter identities at random, which are encrypted using the public key of the voting server and published to the bulletin board. The voters can receive such an anonymous identity and prove they are legitimate. Every time a vote needs to be made, the voting server must create the voting identities and the voter signs with their key so multiple votes can be allocated to a single voter. The voting server verifies the vote and publishes it as a ballot to the bulletin board. The voter can verify that the vote was put forward. Querejeta-Azurmendi et al.

(2020) included dummy votes into the scheme to prevent adversary attacks. The voting server casts dummy votes to hide the actual number of votes in the election. Finally, in the tallying phase, the tellers proceed and decrypt the votes in a verifiable and private manner. Interactive zero-knowledge proofs under the random oracle assumption are used to simulate proofs and turned into non-interactive proofs via Fiat-Shamir transformation.

4.3.2 Electronic Auctions

Zero-knowledge proof implementation efforts for bid e-auction schemes aim to prevent the exposure of bid information and price and protect the bidder's identity. Li & Xue (2021) show that the goal can be met while using an auctioneer no longer becomes necessary. The latest bid price is hidden through Pedersen commitment scheme, and the ZKP algorithm uses bulletproofs so the bidder can prove that the new bidding price is higher than the previous price. Every participating bidder can verify it. The sealed-bid auction smart contract interacts with the blockchain by publishing the commitments and initializing the bidding process for the owner. The information about goods, the price, and the winning bidder is also encrypted in the process. All participants verify the price and the winner without obtaining knowledge about them. This decentralized e-auction scheme ensures sealing and fairness while protecting the participants' privacy. Although there is no cost involved in engaging a third-party auctioneer, the productive cost of running such a platform remain to be explored.

Similar implementations without the use of public blockchain structures are found in D. Wang et al. (2021), which makes use of Hyperledger. Consortium members can use private channels for secure communication while effectively realizing the overall use of smart contracts and transaction privacy. In the endorsement process, the bidder initiates the transaction, and the client submits it to the endorsing peer. The endorsing peer simulates the transaction proposal and verifies it. The transaction is initiated in the ledger and returned to the platform. In the ordering process, the platform performs a combination of transactions and endorsements to be sent to the ordering peer. This peer puts the transactions into transaction blocks for every channel and delivers them to the committing peer. In the verification process, the committing peer verifies the transactions in the blocks, submits them to the ledger, and sends a notification to the platform. Zero-knowledge proofs manage the participants' identity and enable only authorized users to send requests to the client. Despite the promising architecture of Hyperledger Fabric, the practical adoptions can get very complex, depending on the use cases. This implementation requires more

computational resources, in combination with the use of zero-knowledge proofs in particular.

4.3.3 Data Queries and Traceability

Protected data sharing, securing data queries, and corresponding outputs have become increasingly important during the COVID-19 pandemic. Godden et al. (2022) propose Circuitree, a ZKP-based tool that can exchange verified information in zero-knowledge and implement the example of digital COVID-19 certificates. The underlying query language is Datalog. A Datalog verifier is presented, which verifies each logical querying step in the reasoning process. This zero-knowledge Datalog engine is fed with domain-specific sets of rules, e.g., vaccination, test, and recovery data. The verifier, e.g., the restaurant with specific entry rules during the COVID-19 pandemic, creates this rule set in Circuitree. The prover, e.g., a vaccinated person, declares their facts to Circuitree and signs them. It builds the tree-like R1CS structure, converted into a bulletproof-based system. The prover can create the proof by querying the Datalog engine and, e.g., providing it to the restaurant for admission.

The problem of missing privacy preserving traceability for product development and supply chain data is being addressed in Xue & Wang (2022). Parties in the industrial product development sphere can track product development history without trusting each other. There are three main layers to the newly-proposed process architecture. The traceability application layer authenticates data owners and interacts with a third party, the traceability agent. The data privacy layer, which contains the zero-knowledge proof implementation, generates traceability features in zero-knowledge and interacts with the traceability data providers, e.g., another partner in the production process inquiring about tracing some data on a specific product. The data traceability request is processed to create proof via a smart contract. The data owner acts as a verifier of this proof. The traceability features and use of the smart contract happen in the physical data layer and are mainly performed by a third party. The traceability process can be entirely publicly monitored, which solves the transparency problem in the industry while preserving the traceability inquiries and adapting to the low-trust environment of the participants.

4.3.4 Electronic Healthcare

Patient monitoring nowadays is increasingly remote, i.e., health data collection happens via medical devices. Luong & Park (2022) uses zk-SNARKs to enable patient medical

data sharing between medical devices and health service providers. The main interactions happen between the patient, medical device, and health service provider. The health service provider is responsible for collecting patient data from the medical device, analyzing it, and responding with adapted features and enhanced functions in the medical device provided. The patients initialize the process using their public address in the blockchain system and creating signed hashes. The provider uses these hashes to create an arithmetic circuit and initiate the zk-SNARKs protocol. Parameters and the zk-proof are created via a smart contract. Patients can use the smart contract to authenticate themselves and add additional information, e.g., the device. Through a secure key exchange algorithm, the health service provider can share encrypted patient data with medical devices via a secure, zero-knowledge communication channel. The underlying zk-SNARKs used is Pinocchio (Parno et al. 2016), implemented via Zokrates (Eberhardt & Tai 2018). Even though the proof generation takes a long time and the system is not suitable for mobile devices due to the nature of the tools used, it solves current problems due to the lack of anonymous and secure patient data sharing between medical devices and health service providers, e.g., adversary attacks and unauthorized disclosure of health data.

Efforts for patient health data privacy protection also reach the medical insurance purchase and claim process, as underlined by Zheng et al. (2022) and H. Wang et al. (2022). Health data is shared securely, and the patient's identity is disclosed to a minimum using non-interactive zero-knowledge proofs. Patients provide their health data in a regulated and authenticated manner via smart contracts. Hereby, the hospital acts as a fully trusted data generator, interacting with the smart contract. Patients can obtain medical data from the hospital, providing a unique identity. The insurance companies publish their restrictions and requirements, e.g., for purchasing medical insurance, via smart contracts on-chain. The hospital uses these requirements to build the constraint system and circuit and produce a proof alongside the encrypted medical data and patient identity. The medical insurance company can verify this proof, and the smart contract can initiate purchasing or claiming between the patient and insurance company without compromising patient identity and sensitive health data insights. Other peers in the blockchain verify the validity of the payment transaction.

In contrast to the increasing consciousness about data protection and excessive amounts of patient data available, there is a need to process medical data effectively to enable research and healthcare. Implementations of Huang et al. (2020) try to leverage these opposites by implementing secure medical data sharing between patients, healthcare providers, research

institutions, and semi-trusted servers on the cloud. It is achieved using a private blockchain in Hyperledger Fabric combined with zk-SNARKs. Research institutions put their requirements for medical data, e.g., study inclusion criteria, into an arithmetic circuit and publish a proof. Patients encrypt their medical data independently or can authorize their healthcare provider, e.g., hospitals. The encrypted and signed medical data is sent to the semi-trusted cloud server, broadcasting the encrypted data on-chain. Whenever patients decide to share their medical data with research institutions, another proof has to be created to show that the medical data matches the research institution's research criteria. It is achieved via smart contracts and the proof algorithm specified there. It also verifies this proof, and upon success, the patient can generate the re-encryption key used in conversion to the public key of the specific research institution. The cloud server receives this re-encryption key and signs it on top with its public key. This enables the research institution to decrypt the medical data, while the semi-trusted cloud server cannot obtain further knowledge. It is captured in a transaction so that participating nodes in the network can verify via consensus.

4.3.5 Cloud Security

Robust authentication schemes for client and user authentication on cloud servers experience enhancement due to the increasing practical applicability of zero-knowledge proofs. Implementation efforts of W. Liu et al. (2019) show how a center-less and biometric-based single sign-on across cloud services can work. The user gets registered in the registration center, which does not participate another time in the authentication procedure, which removes centralization vulnerabilities. A token service provider generates a zero-knowledge token for the user, which is used across multiple cloud services. The underlying technology is based on circuit-specific zk-SNARKs, e.g., Groth16. An elliptic curve over a finite field with generator points is used, and a common reference string is issued, whereby the token service provider performs the set-up phase. The user adds secret values, i.e., user identity, password, and biometric information, as secret values. The token service provider registers the user and delivers a token without learning the decrypted user's secret values. The cloud service provider also registers via the token service provider. Each registration ends with a zero-knowledge token provision. The user and cloud service provider verify each other, whereby a specific session key is generated. This session key authenticates the user on other cloud service providers.

In Major et al. (2020), a prototype applies zero-knowledge proofs for lightweight and private

client-server authentication based on port knocking. Port knocking is widely used as an authentication mechanism between clients and firewalls, allowing for a channel between them within an untrusted network, e.g., the Internet. The host authenticates the client without open ports, and attacks are difficult because the machine's function as a server is hidden. Non-interactive zero-knowledge proofs are used in the prototype to work towards the goal of hiding any further knowledge from sniffing traffic or eavesdropping. In the set-up phase, profile files for the client and server are created, whereby only the client has a secret private key in the file, which is randomly selected. Furthermore, the files contain the parameters for the ZKP, a private hash key, the server port number, and the command to be run upon successful authentication, e.g., to mount an app service. The client creates the proof, treats it like the knock, and transmits it to the port-knocking server. The server parses the traffic, inspects it, and checks whether ZKP criteria are met, e.g., that the server port specified by the client matches. If the checks are met, the protocol is executed further to perform the computational verification through bilinear pairings. If the verification passes successfully, the client-specific command can be executed.

Password attacks have increased, especially observing the rising usage of cloud storage services for mobile devices. This threat is analyzed in Munivel & Kannammal (2019), and a new authentication scheme is proposed using zero-knowledge proofs for mobile cloud storage authentication. The client server provides a unique in-browser mask for entering user identifier and password. In the background, these values are hashed and do not leave the browser as entered with the user's public key and a random value, which is an element of a cyclic group of prime order. The user's algorithm calculates a proof consisting of a random token, the password hash hidden by some generator from the cyclic group and the random value, and the user's public key. The server calculates the verifying key, and, i.e., similar to the circuit-specific zk-SNARKs verify algorithm, can check whether the proving information sent by the user matches. The server can do this because the server has access to the random token, user public key, and the group element generator, which are public elements. Neither the server nor an adversary can obtain the user password and receive access to the cloud storage.

Apart from research on cloud authentication, there are implementation efforts to use zero-knowledge proofs for storing only one single copy of the same data on cloud servers, i.e., data deduplication efforts. Motivated by mass data storage outsourcing to third-party cloud computing providers, data deduplication and dynamic ownership are promising areas of development. On the assumption that the cloud server is honest but curious, Kanagamani

& Karuppiah (2021) propose a data deduplication scheme using in-line block matching and interactive zero-knowledge proofs. The cloud server performs the deduplication check by verifying the proof of ownership of the file and checking whether a copy is already stored while learning no further information about the ownership and the file. Before the initial upload, the file should have been hashed already. The server registers users and provides them with public and secret keys. The encryption key is obtained by applying another hashing algorithm to the file hash. Further, this key is hashed once more to obtain the tag. The cloud server refers to the tag to check whether a subsequently uploaded file already exists. The user chooses a random encryption key, which is different, and encrypts the encryption key to obtain a ciphertext. The tag, ciphertext, user identifier, and the proof are stored on the cloud server. The proof is obtained by combining in-line block matching with zero knowledge proof generation. The file is divided into blocks, and each block is used to perform an exponential equation with some primes a and calculated in modular arithmetic with another prime $mod\ b$. All computations are performed using a multiplicative cyclic group $\mathbb{G}$ with prime order t. The result of each calculation forms a sequence of individual block proofs, which are the file proof. The server receives the proof and the data described above and checks if the tag matches any other tag of a file stored previously. Group keys are generated for verification and used to perform bilinear pairing checks, i.e., similar to zk-SNARKs, e.g., Groth16. The public key of the file is taken alongside random prime numbers to obtain the group key. The ciphertext is encrypted with the group key by the server and stored with the ownership information. Each time the ownership changes, the server uses the existing proofs to send challenges to the allegedly new owner. The new owner responds by creating secret values from the challenge received. Once the server can verify the proof, secret, and response, another group key is generated and used to encrypt the ciphertext of the file. The ownership information can be changed. The server learned nothing more than a change in ownership and that the file only existed once.

4.3.6 Rollups

Implementing zero-knowledge rollups enhances the throughput of blockchain transactions through outsourcing computation off-chain and only performing the validation part on-chain. Often referred to as validity rollups, they do not show the property of perfect zero-knowledge. In this reflection, only application-specific rollups are discussed. General rollups, e.g., Zcash and Tornado Cash, are not in scope (Chapter 3). In application-specific rollups, the most expensive part of the blockchain application is deployed via rollups. All rollups in practice

can be categorized into validity-style rollups and validium-style rollups (Chen et al. 2022).

Along with the scope of application-specific implementations, Ethereum is the most widely used Turing-complete blockchain to build apps. However, the block validation time of 15 seconds per block results in significantly high gas cost, and with increasing volume, Ethereum can barely serve the number of users (Ethereum Foundation 2023a). The general idea of rollups is to solve this via off-chain computation of transaction states (Chen et al. 2022). In order to realize it, another blockchain layer (L2) is utilized. The main layer (L1) deploys a smart contract, which interchanges tokens with L2 and verifies that computations and transactions on L2 are performed correctly. In an Optimistic Rollup, the L2 scaling is approached with the assumption that every transaction is valid until proven invalid. It depends on users to submit proofs to claim that some transaction was forged to ensure transaction security (Matter Labs 2022). Zero-knowledge rollups are different because the rollups smart contract verifies each transaction state transition before it becomes effective.

Validity-style rollups have the following architecture (Buterin 2021): validators on the mainnet bridge between L1 and L2, receiving submitted transactions by users who signed their transaction activities on L2. These validators perform the following scaling procedure: all submitted transactions are scaled, i.e., aggregated, into a single batch and submitted to the L1 smart contract for further processing. In essence, the L2 state root, a zk-SNARK to prove L2 state root correctness, every transaction header, and the Merkle root of the transaction batch are submitted to the smart contract. In L1, the validation of the entire batch and the suggested update of the Merkle state root can be performed. First, batch verification is more inexpensive. Second, the off-chain storage of the state root increases the performance on the mainnet, boosts its capacity, and decreases transaction fees (Chen et al. 2022). In validium-style rollups, the transaction header is not stored and provided; instead, the SNARK proves the validity of the state transition. Because of this, validium-style rollups are additionally perceived as proof of knowledge, whereas validity-style rollups only serve as proof of computation (Ethereum Foundation 2023b).

4.4 Evaluation and Challenges

As presented above, zk-SNARKs are successfully implemented in various application scenarios, but weaknesses remain to be discussed. The zero-knowledge proofs presented earlier are contrasted to condense differences and similarities. Ultimately, the main challenges of trust assumption, cost, and quantum computing are summarized in this chapter, providing an

outlook on future solution approaches currently discussed in the research.

Zero-knowledge proof evaluation approaches can be grouped into computational complexity analysis (W. Liu et al. 2019; Maller et al. 2019; Zhang et al. 2021), security analysis (Huang et al. 2020), and communication cost (Zheng et al. 2022; X. Liu et al. 2022; Gong et al. 2022). This subchapter compares specific aspects of these categorizations. First, highlighted characteristics of zk-SNARKs, zk-STARKs, and bulletproofs are revisited. Second, the analysis aspects of trusted set-up, communication complexity, and quantum threat are emphasized.

The cryptographic assumption of circuit-specific zk-SNARKs is secure bilinear pairings with a short proof length and smaller memory consumption on-chain. However, a trusted set-up with a CRS is needed. A third party generates the proving and verifying key. A multi-party ceremony (MPC) can diffuse the centralization of trust by letting hundreds of participants generate the CRS. Zk-STARKS have a faster verification and proof speed. The larger proof and circuit size (bytes versus several hundred kilobytes) make zk-STARKs more complex than zk-SNARKs (X. Liu et al. 2022). There is no necessity for a trusted set-up and CRS because of the hash collision-based symmetric encryption, which is free from arguments of knowledge; thus, zk-STARKS are not vulnerable to quantum computers. Bulletproofs do not need a trusted set-up and possess a logarithmically increasing proof size. There is good applicability with range proofs and short proof sizes for general arithmetic circuits, although bulletproofs are faster to be verified than range proofs (Gong et al. 2022). Unlike all other ZKPs discussed in this chapter, bulletproofs enable efficient on-chain proof storage.

The CRS generated in zk-SNARKs relies on a third party. Zk-STARKs utilize verifiable randomness, which makes a CRS obsolete. Bulletproofs can avoid a trusted set-up because of the discrete logarithm assumption and a completed 128-bit security in untrusted environments (Huang et al. 2020). The communication complexity in zk-SNARKs grows nearly linear in $O(1)$ because the prover hands over the proof to the verifier while the computational workload increases. Zk-STARKs have a larger run time with poly logarithmic functional expressions, with N being the input size of the circuit (Table 4.2). In Groth16, the prover complexity can also be described as $m + 3n - l * E$, with m circuit wires, n multiplication gates, l elements in the statements, and E exponentiation operations (Groth 2016). Zk-STARKs possess the highest computational efficiency in proof and verifier complexity (Gong et al. 2022).

Even though quantum computers are not fully mature yet, it is a considerable threat to

Table 4.2: Complexity Comparison of zk-SNARKs, zk-STARKs, and Bulletproofs

	zk-SNARKs	**zk-STARKs**	**Bulletproofs**
Prover complexity	$O(N * log(N)$	$O(N * log(N)^k)$	$O(N * log(N))$
Verifier complexity	$O(1)$	$O(log(N)^k)$	$O(N)$
Proof size	$O(1)$	$O(log(N)^k)$	$O(log(N))$
Post-quantum security	not given	given	not given

be discussed when evaluating zero-knowledge proof algorithms. The quantum threat to ZKPs comes from the ability of quantum adversaries to break encryption algorithms, which threatens blockchain security.

Zk-SNARKs are easy to be broken by quantum computers due to the ability of quantum machines to break polynomial evaluation. It has yet to be discovered how zk-STARKs can be broken by quantum computing due to collision-resistant hashing functions (for IP and ROM). Bulletproofs rely on discrete logarithm assumptions for problem-solving, which makes it particularly easy for quantum computers to break. The commitment values are exposed by adversaries (Gong et al. 2022).

Recalling from the definition of quantum zero-knowledge (QZK) introduced by Watrous (2003), various approaches enhance classic zero-knowledge proofs to become quantum secure. In QZK, the verifier V could be quantum, i.e., for any polynomial-time quantum verifier, there exists a quantum simulator S, such that any mappings induced between them are quantum computationally indistinguishable. Even though there could be a quantum verifier, it should not obtain knowledge. All approaches to obtain quantum resistance focus on classic properties of zero-knowledge proofs and transform them to be suitable against quantum adversaries. For example, security notions of witness indistinguishability (WI) and witness hiding (WH) are studied to construct the quantum variants of those, and ultimately, propose a quantum secure signature scheme to construct the CRS (Xie & Yang 2019; Katz et al. 2018). WI refers to a proof system in which the verifier cannot know which witness was used by the prover. WH describes the verifier's inability to compute the unknown witness through interaction with the prover. Vidick & Zhang (2020) construct a zero-knowledge

protocol sound against polynomial-time quantum provers and both classic and quantum polynomial-time verifiers. It uses a perfect binding, quantum computationally concealing commitment protocol with extended trapdoor functionalities. Another approach to construct QZK focuses on the MPC, providing quantum secure digital signature algorithms based on asymmetric key primitives, e.g., ZKBoo and ZKB++ (Gong et al. 2022).

Recursive zk-SNARKs is a promising research field when discussing the future of ZKPs. With zk-SNARKs, proving the correct execution of a specific multi-step function at a given step requires the computation of each step, which can be improved. For example, not all function steps are known, computing all these steps is computationally too expensive for the use case, or it is not efficiently feasible to process all proofs if the amount of participating provers is high. Recursive zk-SNARKs act as aggregators so that zk-SNARKs can be applied for each iteration to prove the correctness of a previous proof (Chen et al. 2022).

5 Implementation and Results

The following sub-chapters present the results of this master thesis. First, the requirements are summarized. Subsequently, the example computation of a Groth16 proof and verification is presented in detail. The software artifact, the zero-knowledge decentralized application (zk-DApp), is demonstrated. The prototypical artifact, an architecture proposal for the zero-knowledge data structure of spare part certification and metadata information, is introduced as an outcome of the zk-DApp evaluation.

5.1 Summary of Requirements

The requirements introduced in Chapter 2 and 3 are summarized as follows.

1. Zero-knowledge proof systems are a disruptive technology in the domain of blockchain-based research and development. One project requirement is to **accumulate knowledge and current research outcomes** in this field to extend expertise within the project RAPADO (Zedel & Kliewer 2022). The Groth16 algorithm is introduced in detail using an example calculation problem (Chapter 5.2). The problem is split and transformed into an arithmetic circuit in the step-wise computation, arriving at an R1CS. From this, the QAP is calculated. Tools, e.g, homomorphic hiding and elliptic curve pairing, are also covered. Ultimately, following the Groth16 protocol, the key and proof generation and verification mechanisms are illustrated.

2. The requirement of **constructing a trade-off between privacy, confidentiality, and transparency** arises from preliminary work in the project and at the Department of Information Systems (Zedel & Kliewer 2022). The use of blockchain-based solutions enables transparency at the cost of confidentiality. With the constantly increasing awareness for data privacy, combined with trust issues and regulatory data confidentiality requirements in the industry, zero-knowledge proof systems have to be explored and utilized to secure the adoption of project results and products in the future. The first minimum viable product (MVP) of the zk-DApp for MRO data attestations contributes to this goal and satisfies this requirement. The zk-DApp is organized

as follows: the backend directory stores the JSON input schema, the solidity smart contracts, the circom circuit, and corresponding generated files. The lib directory stores shared files between the backend, frontend, and hashing functions used, e.g., in the circuit code. The user interface (UI) directory finds the correct schema and stores a copy of the final protocol transcript key and the witness-generating file. It also stores the frontend code for the landing page and all corresponding pages to submit, attest to and verify MRO data.

3. Current paper-based MRO documentation of aviation spare parts primarily sets the requirement for **fraud-preventive document authenticity checks, enabled through practical data formats and digitization of spare part documentation**. Fraud-preventive verification is covered via the zk-DApp. However, current data digitization efforts for spare part documentation need to be further addressed: the zero-knowledge data structure architecture enables consistent and temper-proof data storage via Merkle trees, and proves specific memberships, e.g., mechanics who worked at a specific part at a given time, with zero-knowledge.

5.2 Groth16 Proof and Verification

Let us use an example to illustrate the underlying mathematical methods applied in zk-SNARKs. The example calculation will use the knowledge of the coefficient assumption for simplification. In practice, the FFT is applied. First, the arithmetic circuit is transformed into an R1CS. The R1CS is used to obtain the QAP. Homomorphic hiding, elliptic curves, and pairing-based cryptography are introduced in more detail and put in context for the next steps of the calculation. Finally, the Groth16 protocol is introduced: First, the key generation steps are explained. Second, the proof is generated. Lastly, the verification steps are illustrated. Formal definitions are found in Chapter 4.2.

Say we want to prove we know a secret x so that

$$x^3 + x + 5 = 35$$

In this case, our secret is x = 3. In practice, we would use hiding and modular arithmetic instead of real numbers and calculations since these are easy to forge and find solutions to, making the proof useless. For the R1CS and QAP, we will proceed with real numbers to show the underlying mechanisms. The following will demonstrate how any computation

that needs to be proven can be converted into polynomial format.

Arriving at a R1CS

A rank-1 constraint system is a mathematical format to help us reduce our problem into a less complex computational problem. First, we flatten the equation by writing a short program to break down the different steps to solve the equation.

1. $sum1 = x * x$
2. $y = sum1 * x$
3. $sum2 = y + x$
4. $out = sum2 + 5$

As shown above, we arrive at an arithmetic circuit with 4 gates and the solution variables

$$x = 3, y = 27, sum1 = 9, sum2 = 30, out = 35.$$

From this, we can construct the solution vector s, starting with a dummy variable of value 1, which we call *one*. Now, the solution vector s is

$$\vec{s} = \begin{pmatrix} one \\ x \\ out \\ sum1 \\ y \\ sum2 \end{pmatrix} \tag{5.1}$$

Each gate will be represented so that

$$\vec{s} \cdot \vec{a_i} * \vec{s} \cdot \vec{b_i} - \vec{s} \cdot \vec{c_i} = 0 \tag{5.2}$$

Let us go through every gate and assign the values for a, b, and c. For the first gate

$sum1 = x * x$, the values of a, b, and c are assigned as follows:

$$a_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$c_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The above is correct because the dot product of s and a, multiplied by the dot product of a and b, subtracted by the dot product of s and c, is 0.

This procedure is applied to every gate. Let us show more complex gates to underline the calculation. For example, the third gate and the fourth gate. The third gate $sum2 = y + x$ could be approached as the first gate by setting the variables in the equation to 1. However, this would not fulfill the equation shown in (5.2). Therefore, the correct values for a_3, b_3 and c_3 are

$$a_3 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$b_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$c_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, we make use of the dummy vector *one* so that we can arrive at

$$30 * 1 - 30 = 0.$$

The fourth gate $out = sum2 + 5$ must also be approached by holding to the dot product equation in (5.2). We have to make use of the dummy vector once again. Setting the values

of $one, sum2$ and out to 1 will give us the following incorrect solution:

$$\vec{s} \cdot \vec{a_4} * \vec{s} \cdot \vec{b_4} - \vec{s} \cdot \vec{c_4} \neq 0.$$

The calculation shows $30 * 1 - 35 \neq 0$, which means we need to add 5 so that the dot product of vector s and a adds up to 35. Therefore, the values of a_4, b_4 and c_4 are as follows:

$$a = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$c = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

By combining our results into matrices, we can set up the corresponding R1CS:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

$$B = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

From R1CS to QAP

The R1CS shows three matrices A, B and C representing the four gates of length six. It is transformed into a QAP by expressing polynomials as sums of Lagrange Interpolation (Chapter 4.2.1). This results in three sets of polynomials $A_i(X), B_i(X)$ and $C_i(X)$, each consisting of six polynomials of degree three. In the following, we will summarize the reason for setting up a QAP instead of continuing with the R1CS. Lagrange Interpolation allows us to develop polynomial coefficients representing each gate when evaluated at an X in the range of the number of constraints (gates). With $X = 1$, the Lagrange Interpolation can be explained quite well because it means that we can add up the coefficients of the polynomials in $A_i(X), B_i(X)$ and $C_i(X)$. Each set of polynomials is built so that evaluated at gate X, whereby X has to be in the range of the number of gates (constraints), will deliver the specific value of X and 0 for the other values in that specific range.
The QAP for our example:

$$\begin{matrix} & & & A_i(X) \end{matrix}$$
$$\begin{bmatrix} -5.0 & 9.166 & -5.0 & 0.833 \\ 8.0 & -11.33 & 5.0 & -0.666 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ -6.0 & 0.5 & -4.0 & 0.5 \\ 4.0 & -7.0 & 3.5 & -0.5 \\ -1.0 & 1.833 & -1.0 & 0.166 \end{bmatrix}$$

$$B_i(X)$$

$$\begin{bmatrix} 3.0 & -5.166 & 2.5 & -0.333 \\ -2.0 & 5.166 & -2.5 & 0.333 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

$$C_i(X)$$

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ -1.0 & 1.833 & -1.0 & 0.166 \\ 4.0 & -4.833 & 1.5 & -0.166 \\ -6.0 & 9.5 & -4.0 & 0.5 \\ 4.0 & -7.0 & 3.5 & -0.5 \end{bmatrix}$$

The corresponding values in the matrices represent polynomial coefficients and shall be read from right to left, e.g., $A1(X) = 0.833x^3 - 5x^2 + 9.166x - 5$. For example:

$$X = 1 \tag{5.4}$$

$$A1(1) = 0, A2(1) = 1, A3(1) = 0, A4(1) = 0, A5(1) = 0, A6(1) = 0 \tag{5.5}$$

Comparing the results with the first vector a of the first gate, we see that the results represent the first gate. For example, evaluating A, B and C at $X = 1$ means adding up the coefficients of the first polynomial of A, which will result in a value matching to the vector value in the first gate. Then, the next polynomial in A, etc. It will result in a vector of length six and be correct if the values match vector a from the first gate in our R1CS. The same is done by evaluating the polynomials with X, whereby X starts at 1 and ends

at the number of gates, in our case $X = \{1, 2, 3, 4\}$. However, it would be cumbersome to evaluate each constraint individually. This is why we can make use of the QAP to check whether the dot product equation of the polynomials will hold:

$$A_i(X) \cdot \vec{s} * B_i(X) \cdot \vec{s} - C_i(X) \cdot \vec{s} = H(X) * Z(X) \tag{5.6}$$

Interestingly, the left side of the equation is our target polynomial $T(X)$, which we want to prove. Now, look at the equation's right side in (5.6). $Z(X)$ is known if we know the number of constraints. In Groth16, it is made available in the trusted set-up. In this case, we have four gates, so we arrive at

$$Z(X) = (x - 1)(x - 2)(x - 3)(x - 4) \tag{5.7}$$

$H(X)$ is the hiding of our initial minimal example, those inputs we do not want to share but prove we know the solution. What role hiding plays will be explained shortly. We want to finish looking at the equation in (5.6). In essence, we want to prove that we know a polynomial and its solution so that

$$T(X)/Z(X) = H(X) \tag{5.8}$$

In our example, $H(X)$ is also a polynomial. The QAP is correct if $H(X)$ is a polynomial without remainder. In this example, the resulting

$$H(X) = -0.44x^3 + 17.055x^2 - 3.666x.$$

Practically, the coefficients of each polynomial in $A_i(X), B_i(X)$ and $C_i(X)$ are publicly known. The same can be said for $Z(X)$ through knowing the number of constraints (in this example, we have four constraints). The prover can calculate the coefficients of $H(X)$ by dividing T(X) / $Z(X)$. However, there is no zero-knowledge yet, since the prover has to prove knowledge of vector s and $H(X)$ without revealing it.

Hiding

zk-SNARKs are dependent on a trusted set-up releasing these parameters. The goal is to prove knowledge of the polynomial $H(X)$ with all its coefficients without disclosing this information. Therefore, the trusted set-up also provides a random secret point P. Note that P is hashed, calculated once, and deleted from memory. Depending on the number of constraints, a certain amount of P values are needed. In our example, we have four constraints, which need $P = 1, P, P^2, P^3$, whereby the value of P^3 corresponds to the value of x^3 when the polynomials are evaluated. The following values of P are provided

$$hh(1), hh(P), hh(P^2), ..., hh(P^{(\text{no. of constraints - 1})}) \tag{5.9}$$

The trusted set-up makes these values publicly available in the CRS.

With our previous knowledge, we know that the proof will consist of

$$\frac{hh[A(P)] * hh[B(P)] - hh[C(P)])}{hh[Z(P)]} = hh[H(P)] \tag{5.10}$$

The hidings of our polynomials are numbers that currently can just be forged. The following will show how it can be proven that these numbers are hidings of the polynomials $A(X)$, $B(X)$ and $C(X)$ in P which is not known to anybody. Furthermore, we need to prove that in order to arrive at $A(X)$, $B(X)$, and $C(X)$, the same solution vector s was used (5.6).

In order to approach the first problem, proving that the hidings of $A(X)$, $B(X)$ and $C(X)$ were calculated in P, we need to "extend" P by the same number, namely u. The CRS also consists of $hh(u * P), hh(u * P^2), hh(u * P^3)$, i.e., it consists of two sets of hidings. We know that $A(X)$ is a linear combination of the values of vector s inserted into the polynomials A1, A2, A3, etc. of A. B calculating $hh[A(P)]$ and $hh[A(u * P)]$ and looking if $hh[A(P)] = u * hh[A(u * P]$ holds, shows that the hiding of $A(X)$ calculated in P is indeed a result of the linear combination of A1, A2, A3, etc. and the values of the vector s (5.6). All we did was prove that the same sets of hidings of P were used to arrive at these numbers. The same is applied to the hidings of $B(X)$, $C(X)$ in P.

For the second problem, to prove the same values of vector s were used to arrive at the hidings of $A(X)$, $B(X)$ and C(X) in P a similar approach can be used. In our example,

vector s has six solution variables. We use a new variable, K as

$$K = K1 + K2 + K3 + K4 + K5 + K6 \tag{5.11}$$

$$K1 = A1(P) + B1(P) + C1(P)$$
$$K2 = A2(P) + B2(P) + C2(P)$$

...

$$K6 = A6(P) + B6(P) + C6(P)$$

By checking that

$$hh[K(P)] = one * hh[K1] + x * hh[K2] + out * hh[K3] + ... + sum2 * hh[K6], \tag{5.12}$$

we can prove that the same coefficients of vector s were used. It is nearly impossible to come up with numbers that hold for another P and to create proofs with the knowledge of the coefficients.

Homomorphic Hiding

$y = hh(x)$ is a hashing function. It is collision-resistant, i.e., one cannot guess anything of x from y. For zero-knowledge proofs, more than this property is required. The hashing function should also preserve algebraic structures so the checks in, e.g., (5.10), can be performed. Let us divide the term Homomorphic Hiding into two sections to explain in more detail. A function $y = hh(x) = e^x$ is homomorphic if

$$hh(a * x1 + b * x2) = e^{\mathrm{a*x1+b*x2}} = e^{\mathrm{a*x1}} * e^{\mathrm{b*x2}} = hh(x1)^a * hh(x2)^b \tag{5.13}$$

As seen in (5.13), the basic exponential laws hold. However, this function is not hiding because one could calculate the logarithmic base e of x because of working with only real numbers $\mathbb{R}$ so far.

We must express variables in a finite field as modulo p, with p being a large prime. The finite field consists only of integer inputs in the range of 1 and some value $p - 2$. This way,

expressing values in modular arithmetic, nobody can guess or calculate our base e anymore. Now,

$$y = hh(x) = G^x, \tag{5.14}$$

where $\mathbb{G}$ is a value in the finite field $\mathbb{F}_p$ and y will always be expressed as modulo p.

With homomorphic hiding being introduced, we know all the tools to prove that we can calculate the equation in (5.8) with the polynomials from the QAP and the same values of vector s without knowing any P and $u * P$. We know how the proof is calculated without revealing our solution vector s. The following deals with verifying that the above equations hold without revealing the solution vector s.

Elliptic Curve Pairing

The goal of PCP and pairing-based zero-knowledge algorithms is to create a succinct proof that a defined computation with given inputs produces specifically known outputs without revealing any information about them and to show that the constraints of that computation hold. Eventually, we want to check if the following equality holds, i.e., that after transforming our problem into a polynomial structure, we know some polynomials so that

$$\frac{A(x) * B(x)}{Z(x)} = H(x) + C(x) \tag{5.15}$$

We have the polynomials A, B and C, not expressed in real numbers but mapped to a finite field with a large prime number. We can calculate $H(X)$ as in (5.8). Now, we will use generators for each of our polynomials to produce points on an elliptic curve. This is necessary to use pairing, which allows us to check if equations, e.g., (5.13), hold without knowing the actual variable values in these equations. In the following, some preliminaries will be introduced to create a basis for the Groth16 CRS generation, proving, and verification mechanism.

Elliptic curves define collision-resistant one-way functions, i.e., homomorphic hiding functions. An elliptic curve is a polynomial, e.g., the elliptic curve used in Bitcoin (Figure 5.1).

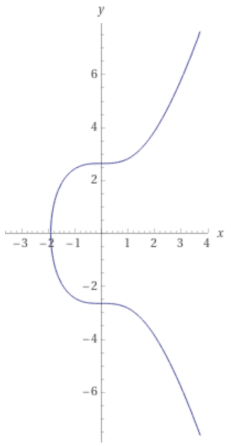


Figure 5.1: $y^2 = x^3 + 7$

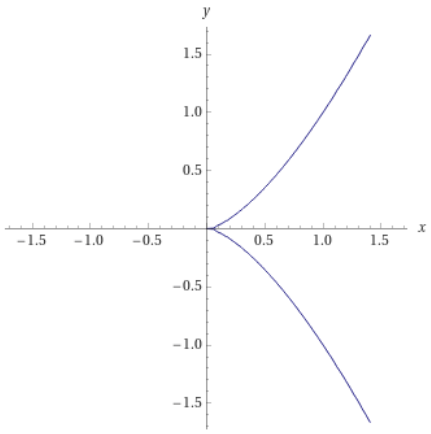


Figure 5.2: $y^2 = x^3$

Elliptic curves are useful for zk-SNARKs because of the discrete logarithm problem, which is believed to be very hard to solve. Given a point g on the elliptic curve and a multiple of that point, $n * g$, it is impossible to solve n, even if g and $n * g$ are given. In order to choose an elliptic curve that offers homomorphic hiding, we need to implement a mapping between our known numbers of the finite field $\mathbb{F}_p$ and a set of points on the elliptic curve (hidden space). Let $\mathbb{F}_p$ be a finite field of order p, whereby p is a large prime, e.g., if $p = 97$, then $\mathbb{F}_p = \{0, 1, 2, 3, ..., 96\}$ For this, we are going to take a generator point $g = (x1, y1)$ that lies on the elliptic curve and multiply it with every element $element_i$ in $\mathbb{F}_p$. For example, $g + g = 2 * g$ is calculated by putting a tangent line on g, and wherever the line crosses the elliptic curve, we receive the result by using the opposite signs of that point. To arrive at $2 * g + g = 3 * g$, the point $2 * g$ is used to draw a line to g, see where the line further intersects with the elliptic curve, and use the opposite signs of that point to arrive at $3 * g$. This is repeated for every *element* in $\mathbb{F}_p$. As a result, we have our finite field mapped to a hidden space on the elliptic curve. In summary, every *element* is hidden by

$$hh(element) = element * g \tag{5.16}$$

Additionally, we have to define what 0 and 1 are. The element 0 results from subtracting a point on the elliptic curve, i.e., when point g goes to infinite. 1 is the point g itself. Now, we have achieved homomorphic addition:

$$(A + B) \longrightarrow (A + B) * g = A * g + B * g \tag{5.17}$$

In order to use elliptic curve pairing to verify zk-SNARKs proofs, e.g., Groth16, we need to achieve a limited homomorphic multiplication operator. The hidden space is a group of points generated by the finite field elements and the generation point, $g1$, on the elliptic curve. Now, we want to choose a subgroup $\mathbb{G}_1$ from that group. We choose that subgroup so that the number of elements we chose, r, is a prime number too. Having found r, we can continue to choose the embedding degree of the elliptic curve. In Groth16, the proving key and verification key consist of $\mathbb{G}_1$ and $\mathbb{G}_2$ element. The embedding degree k has to be found so that $p^k - 1 \mid r$, i.e., is a multiple of it. Let us use a minimal example to show how to arrive at $\mathbb{G}_1$ and $\mathbb{G}_2$. Let us define an example base field $\mathbb{F}_p = \{0, 1, 2\}$ with $p = 3$. We have found an embedding degree $k = 2$. In order to achieve our goal of creating a subgroup $\mathbb{G}_2$, we need to extend our base field by a defining polynomial. This polynomial is of degree k, and no element of our base field evaluates it to 0. In summary, we have the following:

$$\mathbb{F}_p = \{0, 1, 2\}, p = 3, k = 2 \tag{5.18}$$

Defining polynomial for field extension $\mathbb{F}_p^k$:

$$\begin{aligned} f(x) &= z^2 + 1 \\ f(0) &= 1 \\ f(1) &= 2 \\ f(2) &= 5 mod 3 = 2 \end{aligned}$$

As shown in (5.18), none of the base field elements make f(x) evaluate to 0. In order to create the elements of the field extension $\mathbb{F}_p^k$, we have to create all possible degree 2 polynomials out of the combinations of our base field $\mathbb{F}_p$. For example, one possible polynomial with the coefficients from our base field is:

$$1 * z^2 + 2 * z + 0 \tag{5.19}$$

$$(z^2 + 2 * z) \mod (z^2 + 1) = 2 * z - 1 \mod 3 = 2 * z + 2$$

$2 * z + 2$ is one element of the extension field $\mathbb{F}_p^k$. In total, $\mathbb{F}_p^k$ has 9 elements, all calculated as in (5.18). In Summary, the elements of our extension field $\mathbb{F}_p^k$ are

$$\{0, 1, 2, z, z+1, z+2, 2z, 2z+1, 2z+2\} \tag{5.20}$$

As shown in (5.20), the elements of the extension field are polynomials of degree up to $k-1$. Addition and multiplication are defined in the way that coefficients are calculated $\mod 3$ and polynomials $\mod z^2+1$, the defining polynomial $f(x)$ from (5.18).

Now, having our extension field, we can use it to create $\mathbb{G}_2$, a subgroup of points of the same elliptic curve used for $\mathbb{G}_1$, but with elements of $\mathbb{F}_p^k$, instead of base field $\mathbb{F}_p$. For this, we have to define points, whereby x and y coordinates are polynomials from $\mathbb{F}_p^k$. $\mathbb{G}_2$ will consist of combinations from $\mathbb{F}_p^k$ in the form of (x, y), which satisfy the elliptic curve.

Pairings are bilinear maps that combine elements of two spaces to receive an element of a third space, e.g., matrix multiplication. In Groth16, the following pairing notation is used:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T \tag{5.21}$$

The result of all steps performed previously is an incomplete homomorphic multiplication that enables us to check that the correct polynomial coefficients were used for $A(x), B(X),$ and $C(x)$, as well as the same solution vector s. It is incomplete because not more than two elements can be multiplied. However, this satisfies the use case for zk-SNARKs.

Groth16

Preliminaries to understand the Groth16 protocol have been covered. In the following, we will describe the setup, proof, and verification steps in Groth16. The parameters are summarized in Table 5.1.

Key generation

The proving and verification keys are obtained from the Common Reference String (CRS) via multi-party computation. From $\mathbb{F}_p$, a set of random values is generated. This toxic

Table 5.1: Groth16 - Parameter Summary

Parameter	**Definition**
n, m	number of constraints, number of variables
$\mathbb{F}_p$	finite field of prime order p
$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	groups of points of prime order p satisfying an elliptic curve
$\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$	bilinear pairing
$g_T = e(g_1, g_2)$	generators with mapping
$\left\{A_i(X), B_i(X), C_i(X)\right\}_{i=0}^{m}$	encoded computation as result of R1CS and QAP: three sets of polynomials of degree $n-1$
$Z(x) = (x-1) * (x-2) * (x-3)...(x-(n-1))$	minimal polynomial, known because n is known
l	number of public inputs
$(s_1, ..., s_l)$	elements of witness whose inputs are public (e.g., out = 35 in our example)
$(s_{l+1}, s_{l+2}, ...s_m)$	elements of witness for secret input x, with $s_0 = 1$

waste (tw), or trapdoor, must be secret and forgotten from memory because knowledge of it enables forged proofs. Note that τ is the random point P from our examples. From the toxic waste, polynomial $L_i(x)$ is defined:

$$tw = (\alpha, \beta, \gamma, \delta, \tau) \tag{5.22}$$

$$L_i(x) = \beta * A_i(X) + \alpha * B_i(X) + C_i(X)$$

The CRS consist of $\sigma = ([\sigma_1]_1, [\sigma_2]_2)$, which are elements of $\mathbb{G}_1, \mathbb{G}_2$.

$$[\sigma_1]_1 = [(\alpha, \beta, \gamma, \delta, \tag{5.23}$$
$$1, \tau, \tau^2, \tau^3, ..., \tau^{n-1}, \tag{5.24}$$
$$\frac{L_0(\tau)}{\gamma}, ..., \frac{L_l(\tau)}{\gamma}, \tag{5.25}$$
$$\frac{L_{l+1}(\tau)}{\delta}, ..., \frac{L_m(\tau)}{\delta})]_1 \tag{5.26}$$

$$[\sigma_2]_2 = [(\beta, \gamma, \delta,\ 1, \tau, \tau^2, \tau^3, ..., \tau^{n-1})]_2$$

- (5.23): elements of the toxic waste
- (5.24): powers of τ of degree up to $n-1$
- (5.25): The polynomial is chosen from the set of polynomials of $A(X), B(X), C(X)$, which corresponds to the place of the public input of the solution vector. In our starting example, $out = 35$ is the public input (since this is our only public input, $l = 1$). The public input is in third place in s. Hence, $A_3(X), B_3(X), C_3(X)$ are chosen, evaluated at τ and multiplied by α, β.
- (5.26): Same as (5.25), but for the non-public inputs of s. All elements of $[\sigma_1]_1$ are $\mathbb{G}_1$ elements, e.g., $\alpha_1 = g_1 * \alpha$.

The proving key consists of the following elements:

- $[(\alpha, \beta, 1, \tau, \tau^2, \tau^3, ..., \tau^{n-1}, \frac{L_{l+1}(\tau)}{\delta}, ..., \frac{L_m(\tau)}{\delta})]_1$
- $[(1, \gamma, \delta)]_2$
- circuit information about the polynomials:

  $A_0(X), A_1(X), ..., A_m(X)$

  $B_0(X), B_1(X), ..., B_m(X)$

  $C_0(X), C_1(X), ..., C_m(X)$

  $Z(x) = (x-1)(x-2)(x-3)...(x-(n-1))$

The verification key consists of the following elements:

- $[(1, \frac{L_0(\tau)}{\gamma}, ..., \frac{L_l(\tau)}{\gamma})]_1$

- $[(1, \gamma, \delta)]_2$
- precomputed pairing $[\alpha * \beta]_T$, which is a $\mathbb{G}_T$ element

Generating the proof

Two random numbers r, t are generated from $\mathbb{F}_p$, that are used to compute

1. $A = \alpha + s_0 * A_0(\tau) + s_1 * A_1(\tau) + ... + s_m * A_m(\tau) + r\delta$
2. $B = \beta + s_0 * B_0(\tau) + s_1 * B_1(\tau) + ... + s_m * B_m(\tau) + t\delta$
3. $C = \frac{s_{l+1}L_{l+1}(\tau)}{\delta} + \frac{s_{l+2}L_{l+2}(\tau)}{+}... + \frac{s_{lm}L_{lm}(\tau)}{\delta} + \frac{H(\tau)Z(\tau)}{\delta} + At + Br - rt\delta$

The proof π consists of two elements from $\mathbb{G}_1$ and one element from $\mathbb{G}_2$:

$$\pi = ([A]_1, [B]_2, [C]_1) \tag{5.27}$$

Verification

In Groth16, three pairings are checked during verification. $[\alpha * \beta]_T$ is a precomputed pairing and is made available in the setup phase. The verification computation receives proof π and accepts it only if the following equation holds:

$$[A]_1 * [B]_2 = [\alpha]_1[\beta]_2 + \Big[\frac{s_0L_0(\tau)}{\gamma} + \frac{s_1L_1(\tau)}{\gamma} + ... + \frac{s_lL_l(\tau)}{\gamma}\Big]_1 * [\gamma]_2 + [C]_1 * [\delta]_2 \tag{5.28}$$

As shown in (5.28), the following three pairings are needed to be checked:

- $e([A]_1, [B]_2)$
- $e(\Big[\frac{s_0L_0(\tau)}{\gamma} + \frac{s_1L_1(\tau)}{\gamma} + ... + \frac{s_lL_l(\tau)}{\gamma}\Big]_1, [\gamma]_2)$
- $e([C]_1, [\delta]_2)$

Let us evaluate the verification equation in (5.28). The left-hand side evaluates as follows:

$$
\begin{aligned}
[A]_1 * [B]_2 = [A * B]_T &= [\alpha + s_0 * A_0(\tau) + s_1 * A_1(\tau) + ... + s_m * A_m(\tau) + r\delta]_1 * \\
&\quad [\beta + s_0 * B_0(\tau) + s_1 * B_1(\tau) + ... + s_m * B_m(\tau) + t\delta]_2 \\
&= [(\alpha + A(\tau) + r\delta) * (\beta + B(\tau) + t\delta)]_T \\
&= [\alpha * \beta]_T + [\alpha * B(\tau)]_T + [\alpha * t\delta]_T \ + [A(\tau) * \beta]_T + \\
&\quad [A(\tau) * B(\tau)]_T + [A(\tau) * t\delta]_T + [r\delta * \beta]_T + [r\delta * B(\tau)]_T + \\
&\quad [r\delta * t\delta]_T \\
\\
&= [A(\tau) * B(\tau)]_T + [\alpha * \beta]_T + [\alpha * B(\tau)]_T + [\alpha * t\delta]_T \\
&\quad + [A(\tau) * \beta]_T \ + [A(\tau) * t\delta]_T + [r\delta * \beta]_T + [r\delta * B(\tau)]_T \\
&\quad + [r\delta * t\delta]_T
\end{aligned}
$$

The right-hand side evaluates to:

$$
\begin{aligned}
&= [\alpha]_1[\beta]_2 + \Big[\frac{s_0 L_0(\tau)}{\gamma} + \frac{s_1 L_1(\tau)}{\gamma} + ... + \frac{s_l L_l(\tau)}{\gamma}\Big]_1 * [\gamma]_2 + [C]_1 * [\delta]_2 \\
&= [\alpha * \beta]_T + [(s_0 L_0(\tau) + s_1 L_1(\tau) + ... + s_l L_l(\tau))]_T + [(s_{l+1} L_{l+1}(\tau) + s_{l+2} L_{l+2}(\tau) + ... \\
&\quad + s_{lm} L_{lm}(\tau)) + H(\tau)Z(\tau) + At\delta + Br\delta - rt\delta\delta]_T
\end{aligned}
$$

Now, we can replace A and B. We also see that the middle of the equation is $L_i(\tau)$.

$$
\begin{aligned}
&= [H(\tau) * Z(\tau) + C(\tau)]_T + [\alpha * \beta]_T + [\alpha * B(\tau)]_T + [\alpha * t\delta]_T + [A(\tau) * \beta]_T \\
&\quad + [A(\tau) * t\delta]_T + [r\delta * \beta]_T + [r\delta * B(\tau)]_T + [r\delta * t\delta]_T
\end{aligned}
$$

Eventually, we got the equality check we wanted to achieve (5.15). The use of the secret encoded values α, β of the toxic waste (5.22) forces the prover algorithm to use the same coefficients of the solution vector (witness) to compute $A(X), B(X), C(X)$. γ, δ ensure that the public inputs of order l are independent from the solution vector (witness). In order to achieve the zero-knowledge aspect, r, s are used to shift the proof randomly.

5.3 PLONK-based zk-DApp

The decentralized application for zero-knowledge MRO data attestation (zk-DApp) facilitates data entries and data sharing between MRO data owners (submitters) and aviation authorities (attesters) with zero-knowledge. This software artifact attempts to ensure the compatibility between the lack of trust among industry participants and the need for data verification and transparency (a trade-off between requirements within the project). MRO data need to be transparent for potential secondary market trading of aviation spare parts without giving full access to the documentation, e.g., to the entire shop report.

5.3.1 Implementation

The zero-knowledge part of the zk-DApp is implemented via circom and snarkjs, developed by iden3. iden3 is an open-source project focused on scalable and distributed identity systems on zero-knowledge proofs, i.e., providing protocols, data structures, and modules. The current objectives include achieving self-sovereign identities free of charge, minimizing on-chain transactions, re-designing privacy with non-reusable proofs, creating complete products, e.g., user wallets, and focusing on community standardization (iden3 2019). Circom is Rust-based and allows one to write and compile arithmetic circuits. In circom, there is a data type called field elements, which per default represents values modulo a large prime number based on the BN128 elliptic curve (iden3 2023a). R1CS files are generated by compiling the circuit: the constraint system has binary format, a symbols file for debugging, and files that are necessary to generate the witness. In order to compute the witness, an input file in JSON format needs to be specified that contains the witness information.

In the zk-DApp, the submitter enters the information to be verified, which is processed according to an input schema. The witness is computed and saved in a file format accepted by snarkjs. Snarkjs is a JavaScript library that continues with the outputs from circom and executes a specified zero-knowledge proof protocol, e.g., currently, Groth16, PLONK, and Fflonk are supported (iden3 2023b). For this use case, snarkjs' PLONK implementation is utilized. For the verification and intermediate result hashes, a powers of tau file is utilized to perform the multi-party computation. In Groth16, it is utilized during the trusted set-up. In PLONK, this is the source of public randomness. Figure 5.3 shows the main steps performed for a PLONK-based zk-DApp. PLONK does not require a trusted ceremony for each circuit, i.e., the powers of tau ceremony is universal. After compiling the circuit and calculating the witness, the snarkjs PLONK can be set up. A final protocol transcript key

(zkey) is generated and verified. The zero-knowledge key includes proving and verification key (iden3 2023b). From the zkey, the verification key is exported in JSON format. With this verification key file, the final zkey file and the witness file, the proof file, and the file with the public inputs and outputs are generated. These two files and the verification key are used to verify the proof. The verifier is transformed into a smart contract (PlonkVerifier.sol) that performs the verification. In order to utilize the PlonkVerifier smart contract exported in the previous step, an interface must be created (IPlonkVerifier.sol), which initiates the function verifyProof and outputs a true or false. The ZkDocument.sol smart contract, which stores user inputs, e.g., the commitments and trusted attesters, uses this interface to verify the proof and update the states of the smart contracts. With every user input, a witness and proof are calculated. The commitment is posted, and the smart contracts are updated. Once verified, the user interface (UI) shows an update, e.g., a green highlighting. For deployment, broadcasting, and development, hardhat testnet is used. Metamask is used

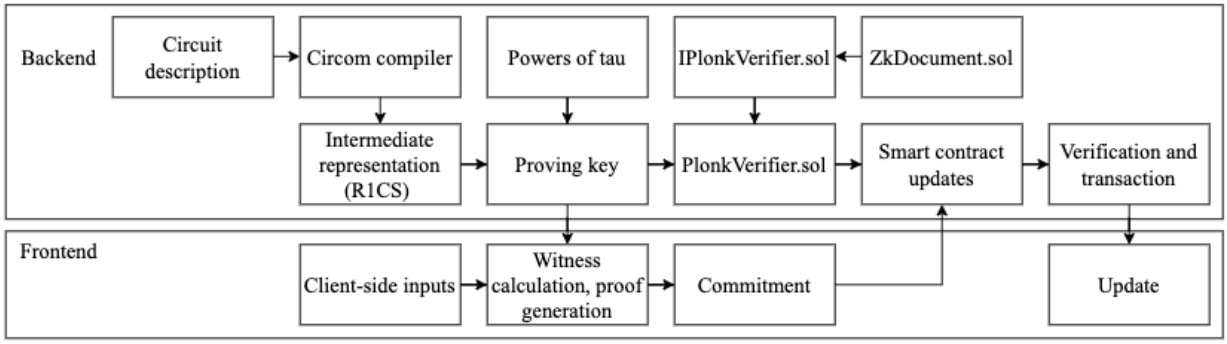


Figure 5.3: PLONK-based zk-DApp Interaction Process Flow

as a crypto wallet browser plugin.

5.3.2 Artifact

Different roles and the steps to be performed in the zk-DApp are described as follows.

1. The process flow and general functionalities are depicted.
2. Essential implementation logic and features are highlighted with references to the implementation code.

For the zk-DApp to start, the command *yarn start* needs to be executed in the zkdocs-backend directory. It will start the hardhat test network and show a set of test accounts

with private keys to use. These test accounts are also used for the accounts of the verifier, submitter, and attesters. The following command in Figure 5.4 is also run in the same directory and deploys the schema for the MRO data entry. It triggers the circom and snarkjs ceremonies and shows the number of constraints, inputs, outputs, wires, and labels in the arithmetic circuit. Also, it shows that the corresponding R1CS, symbols, and witness-generating files are written successfully. Once the setup is finished, the public keys of the deployer (submitter), PLONK verifier (PlonkVerifier.sol smart contract address), and zkDoc (ZkDocument.sol smart contract address) are shown. The zkDoc public key needs to be parsed every time another set of inputs needs to be made or verification needs to be performed. Ultimately, the schema is created, and the corresponding directories are listed. The UI can be started with the same command *yarn start* in the zkdocs-ui directory.

```
elinabeletski@Elinas-MacBook-Pro-2 zkdocs-backend % yarn hardhat build-deploy-schema --schema ./test/test_schemas/mro_bracket.json --network localhost && cd .. && ./cp_script.sh mro_bracket ./zkdocs-backend/test/test_schemas/mro_brack
.json --verbose
yarn run v1.22.19
$ /Users/elinabeletski/Documents/GitHub/zkdocs/node_modules/.bin/hardhat build-deploy-schema --schema ./test/test_schemas/mro_bracket.json --network localhost
(node:17821) ExperimentalWarning: stream/web is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
Circom build:
 template instances: 73
non-linear constraints: 1108
linear constraints: 0
public inputs: 6
public outputs: 0
private inputs: 12
private outputs: 0
wires: 1121
labels: 3815
Written successfully: /Users/elinabeletski/Documents/GitHub/zkdocs/zkdocs-backend/circuit_cache/circuit.r1cs
Written successfully: /Users/elinabeletski/Documents/GitHub/zkdocs/zkdocs-backend/circuit_cache/circuit.sym
Written successfully: /Users/elinabeletski/Documents/GitHub/zkdocs/zkdocs-backend/circuit_cache/circuit_js/circuit.wasm
Everything went okay, circom safe

snarkjs gen zkey:
 [INFO]  snarkJS: Reading r1cs
[INFO]  snarkJS: Plonk constraints: 11288
[INFO]  snarkJS: Setup Finished

Nothing to compile
No need to generate any newer typings.

Deploying contracts
- Deployer public key: 0xf39Fd6e51aad88F6F4ce6aB8827279cfFFb92266
- Plonk Verifier address: 0x5FbDB2315678afecb367f032d93F642f64180aa3
- zkDoc address: 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
✨  Done in 7.27s.
Creating directory at ./zkdocs-ui/public/test_schemas/mro_bracket
mkdir: ./zkdocs-ui/public/test_schemas/mro_bracket: File exists
Copying ./zkdocs-backend/circuit_cache/circuit_js/circuit.wasm -> ./zkdocs-ui/public/test_schemas/mro_bracket
Copying ./zkdocs-backend/circuit_cache/circuit_final.zkey -> ./zkdocs-ui/public/test_schemas/mro_bracket
Copying schema ./zkdocs-backend/test/test_schemas/mro_bracket.json -> ./zkdocs-ui/public/test_schemas/mro_bracket
elinabeletski@Elinas-MacBook-Pro-2 zkdocs %
```

Figure 5.4: Initiating the Deployment of PLONK in circom, snarkjs in the zk-DApp Backend

Sequence of events

The zk-DApp has three roles: verifier, submitter, and attester. In this scenario, the verifier also acts as administrator, operating the zk-DApp. Hence, the administrator operates the schema with constraints, fields, and the list of trusted institutions that can attest in the zk-DApp. The submitter owns MRO data and seeks validation. The attester is assigned to view specific information shared with them and can validate it, i.e., attest to the data provided. The verifier can obtain the proof and verify it. Other stakeholders in the system can learn that there is verified information but do not learn anything besides its correctness. The following sequence of events refers to the landing page of the zk-DApp (Figure 5.5).

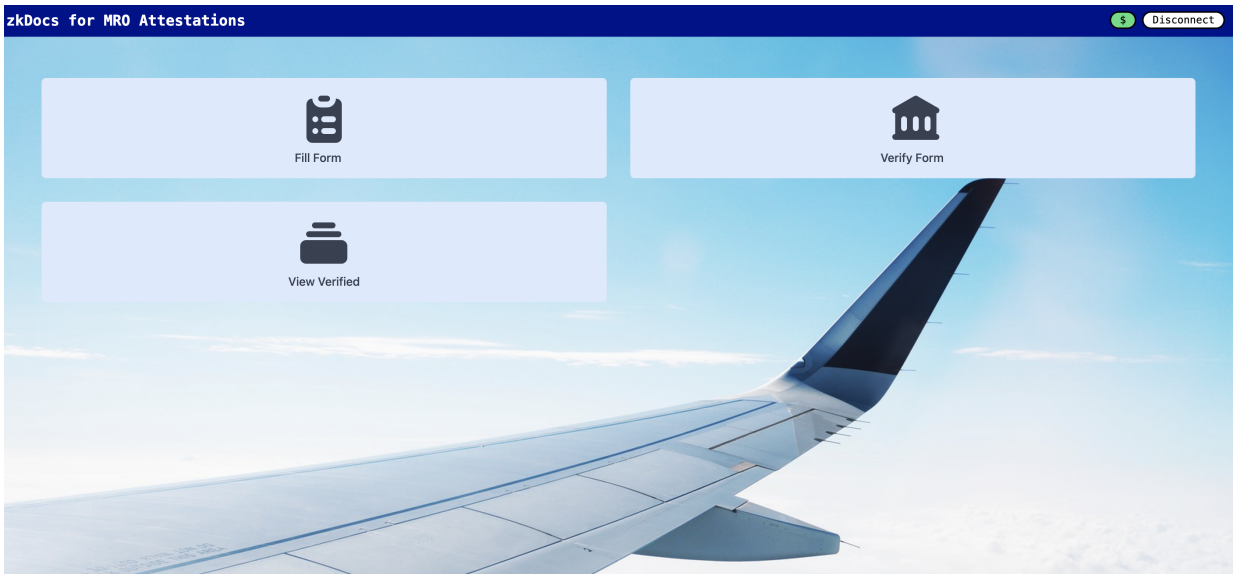


Figure 5.5: zk-DApp Landing Page

1. *Fill Form*: The submitter connects their wallet and can see an input mask for MRO data entries (Figure 5.6). In this prototype, the submitter can enter the part identifier, flight hours, time since the last service, and MRO cycles. Once the values are added, the submitter decides which authority needs to attest to the values, e.g., FAA, EASA. If the constraints are met, the submitter generates the proof, which results in the following: First, the proof is generated and displayed. Second, communication links are generated, which can be parsed in another browser tab. These links are specific to the respective attesters, so the data assigned to them can be viewed. In practice, this has to be realized via separate communication channels to alert attesters once a new set of MRO data inputs is performed. The submitter commits to these entries and the proof on-chain. Now, the values can be viewed by the respective attester and validated.

2. *Attest to MRO data*: The respective attester follows the attestation link, connects their wallet, and only sees the decrypted values of the data assigned to them for validation. The attester marks the data as valid and commits the result. Verification can be initiated once all attesters have successfully attested to the data.

3. *Verify Form*: Another user, e.g., the administrator, the submitter, or any dedicated user with the proof, can perform the verification. The verifier connects their wallet and checks the attestations. The only information available is the green mark, that

data has been successfully attested to, which fields have been attested to, and who attested to these fields and values. The values are shown in an encrypted format. With the proof parsed, the verifier can verify and submit. If the attestations are correct, a bit will be flipped in the smart contract (Andreessen Horowitz a16z 2023), and the verification will be successfully performed.

4. *View Verified*: Any participant can view the latest verified information. However, only the green mark shows the field names, which trusted authority performed the attestation, and whether the data is correct. The verifier and other participants learn nothing besides that the information provided is correct and has been truthfully attested by a trusted institution.

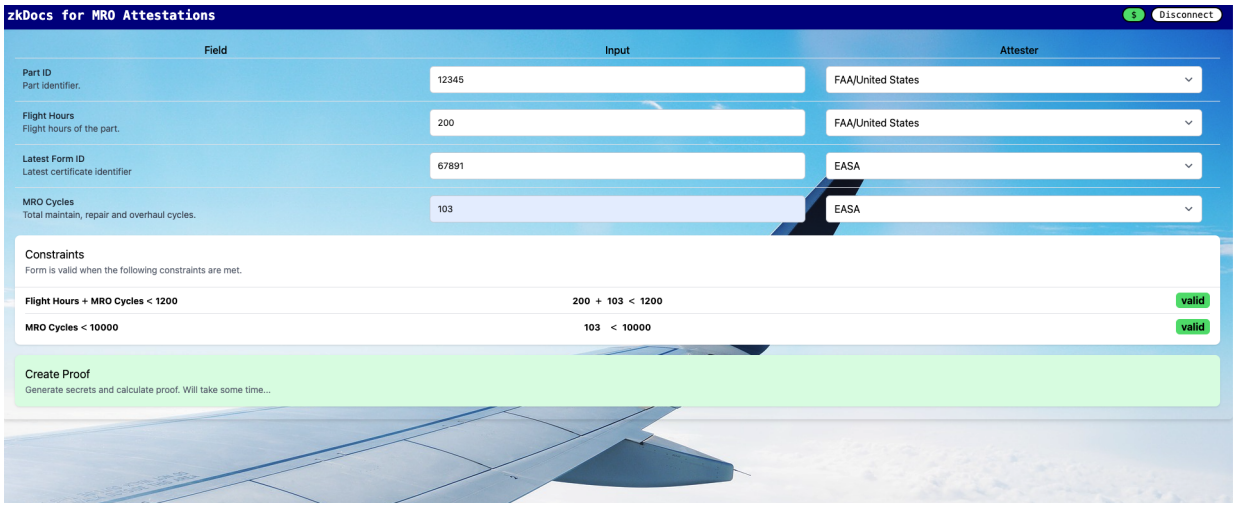


Figure 5.6: zk-DApp Fill Form (Initial)

Highlighted features

The following features illustrate the underlying functionality applied in the zk-DApp.

The circuit and witness calculation input is processed through a schema. The JSON template is a rule book for the input field and values, constraints, and trusted attesting addresses. ZkDocSchema.ts uses the JSON template to define the schema. It sets valid operators and constraint types and validates if the template matches those. Some further value validations are build-in as well, e.g., checks for non-negative numbers.

ZkDocGenerator.ts acts as a wrapper for circom and snarkjs. It contains all necessary commands to initiate the circuit, represent intermediate formats, e.g., R1CS, calculate

the witness and the zero-knowledge key file, and export the verifier as smart contract. It generates the constraint string and looks up the field indices of the fields to attest in the schema. The circuit's functionality is specified in circuit.circom. It defines the constraints checks for each field if the commitment matches the hashed commitment on-chain. During attestation, the assigned attester performs the decryption of the values locally.

The ZkDocument.sol smart contract handles the user input values. It stores the selected attester addresses, the submitter's address, field commitments, and field indices to be attested. A nonce is generated for each field and value, and the commitment scheme is initiated. The commitment scheme for each field is hash(value, nonce). The hashing algorithm used in the circuit is POSEIDON. Due to its reduced prover and verifier complexity, POSEIDON is currently favored in research and developing zero-knowledge decentralized applications. Unlike other popular hashing functions, e.g., SHA-256, POSEIDON can operate on smaller circuits and is adapted to finite fields (Grassi et al. 2019). It can add and remove trusted attesters from the list via addValidInstitution, addValidInstitutions, and removeValidInstitution. It ensures that only the attester specified by the submitter via mask entry can attest to the fields and values committed. During this check, it is also secured that the attestation's field index matches the one specified during submission. Once the check is successful, the bit is flipped to confirm the correctness of the attestation. This procedure is simultaneously applied to all field attestations via attestMultiple.

The function postFields posts the commitments and checks that 1. the commitment length matches the number of fields, 2. the correct number of institutions to perform attestations are in the commitment, and 3. the commitment is unique, and has not been posted previously. Through validateSubmitter, the public signal array for PLONK verification is filled with the commitments. It checks that the specified institutions have attested to all fields. As described in the introduction of circom and snarkjs in Chapter 6.2, the verifier is transformed to the Solidity smart contract PlonkVerifier.sol, containing the values and function logic needed for PLONK verification. ZkDocument.sol uses the interface to PlonkVerifier.sol, the IPlonkVerifier.sol to initiate the function verifyProof, which verifies the proof. By using the interface, it is made sure that a specific standardized function from the generated PlonkVerifier.sol is used (verifyProof). This implementation ensures that the state of PlonkVerifier.sol undergoes an actual change. In order to prevent client-side attacks, the schema hash, using the Keccak-256 hashing algorithm (part of SHA-3 Solidity family), is stored in ZkDocument.sol as well.

The UI finds the correct schema via the scheme hash stored, facilitated through SchemaU-

tils.ts. It obtains the corresponding schema from the schema hash and can utilize the files generated through the PLONK procedure in circom and snarkjs. Additional Figures are provided in Appendix 6.2.

5.4 Evaluation

zk-DApp MRO Attestations

On the developer's machine with a 2,2 GHz 6-Core Intel Core i7 processor and macOS 13.2.1 (22D68) operating system, the circom build, snarkjs PLONK setup, and contracts deployment is executed in 6.90 seconds. Although it has weaknesses compared to Groth16, the implementation with PLONK is motivated by its capability to be built on top of any polynomial commitment scheme. This feature makes it applicable in zk-rollups, i.e., the compatibility with the EVM (Ethereum Virtual Machine) is responsible for the massive scaling of the Ethereum blockchain. Additionally, there is the outlook for the future availability of the snarkjs TURBOPLONK, whereby the number of gates is decreased by introducing custom gates appearing multiple times within a circuit. Comparisons with implementations based on the MiMC hash show that PLONK outperforms Groth16 2.5 times (Walton-Pocock 2019). While the performance depends on the underlying curves and their field sizes, resulting in different pairing friendliness, recent comparisons with Poseidon-128 show less prover time with PLONK. PLONK operates with a larger proof size, i.e., 9 elements in $\mathbb{G}$ and additional 7 field elements in $\mathbb{F}$ (Gabizon et al. 2019). In comparison, Groth16 only needs 2 elements in $\mathbb{G}_1$ and 2 elements in $\mathbb{G}_2$. Any smart contract modification, e.g., circuit changes, need a new trusted set-up. For the Aztec protocol, the private transaction protocol for the Ethereum mainnet, this procedure took more than six weeks (Walton-Pocock 2019). All circuit-specific zk-SNARK implementations need a trusted set-up, which can be made less centralized via MPC. An MPC is characterized by many different parties initializing the generation of proving and the verifying key for the trusted set-up (Thaler 2023).

The result of the evaluation cycle is described as follows, taking feedback from the recent project workshop into consideration: The implementation of the software artifact zk-DApp represents a seized opportunity to meet the second project requirement of constructing a trade-off between privacy, confidentiality, and transparency. This requirement is successfully met due to the nature of zero-knowledge proofs realizing transparency and confidentiality in the trustless environment of MRO spare parts validation.

On the one hand, the use case of MRO key performance indicator (KPI) validation needs to be revisited and could be replaced to meet other project targets, e.g., validation of non-KPI relevant MRO data. The data in the prototype is captured per part identifier only, i.e., the form identifier is obsolete for this prototype. The time since the last service on the part was performed is also relevant and should be added as a third constraint (Figure 5.6). The data points are to be perceived as stand-alone, i.e., validation calculation checks and threshold values yet have to be investigated. Except for the data points, further content analysis and feedback in the MRO industry will be performed in future research to match the use case in this prototype in more granularity.

On the other hand, the demonstration of the technical functionality is successful, and the previously developed DApp for MRO certificate validation and the MRO Smart Hub (in development through Opremic GmbH) can be extended with the zero-knowledge proof functionality of this software artifact. The current code shared with the thesis shows the implementation of the most recent feedback of the evaluation cycle. A screenshot of the input screen (in analogy to Figure 5.6) is shown below (Figure 5.7). Assumptions about the data

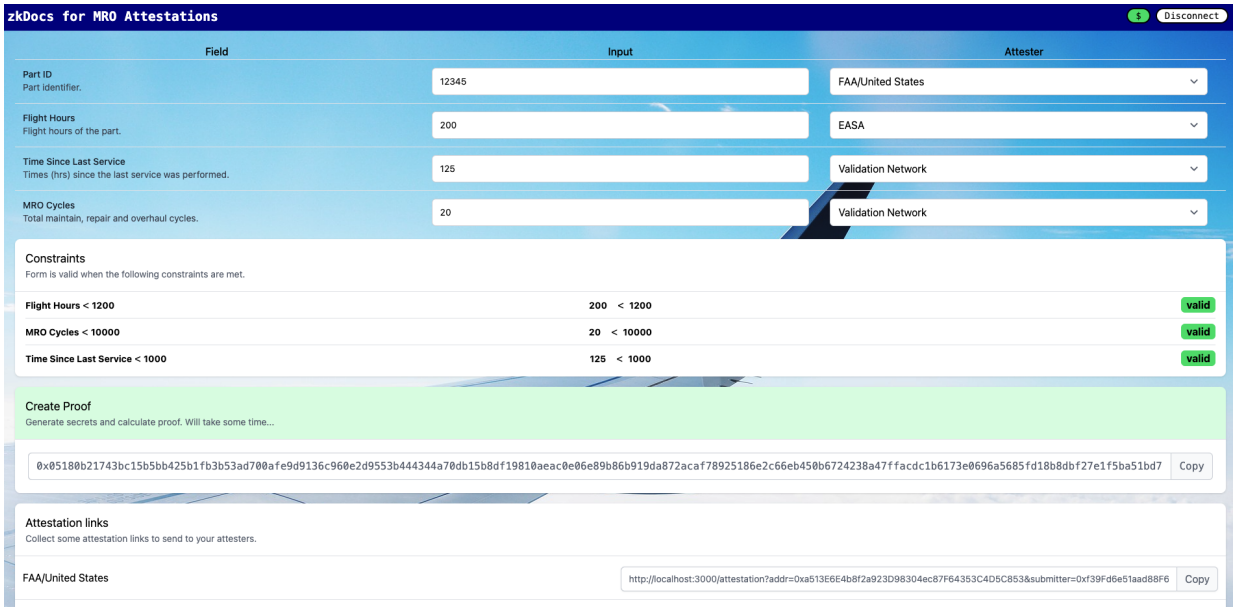


Figure 5.7: zk-DApp Fill Form (Follow-Up after Evaluation)

structure led to the development of the first zk-DApp prototype and are revisited during the evaluation cycle to review the application's data layer in more detail. Current paper-based documentation must be replaced by the automatic digitization of spare part certificates and

brought into a practical storage structure with part identifiers and corresponding certificate identifiers. While the automatic digitization of spare part documents is being realized in the project via optical character recognition (OCR), the authenticity validation check of the data in a trustless environment is still to be approached. As an evaluation aspect for further research implementation, an architectural zero-knowledge data structure is outlined as an opportunity to meet the third project requirement. In parallel, this architecture is the basis to increasingly capture MRO KPI data through combination with the zk-DApp in further research.

Zero-Knowledge Data Structure

The lack of a structured data foundation for proving the authenticity of aviation spare part documentation is an obstacle to enabling an industry-wide adoption of a blockchain-based industry standard for document verification approaches. In combination with the document issuing organization, there is a need to prove the authenticity of documents, e.g., via a transparent availability of the mechanic's signature and the document date stamp. However, this need for transparency conflicts with data privacy and the large amount of paper-based data captured in the past, which cannot be traced back effectively. Exploring a Merkle-tree-based, zero-knowledge proof data structure satisfies the third requirement to ensure document authenticity and data integrity. Further maturation of this architecture, especially the degree of integration with the zk-DApp prototype, must be addressed further.

Incoming MRO data is structured via a Merkle tree, which is initialized by an aggregator role, the operator. The operator receives data sent by the client, e.g., a zk-DApp. The operator maintains the Merkle tree locally. Once MRO data is submitted, newly created leaves are added to the Merkle tree. The structure is organized as follows: spare parts and subordinated certificates have a unique identifier, e.g., a part identifier has many certificates, each with a unique part id and certificate id combination. Each certificate is issued by an organization that employs mechanics. Mechanics service spare parts, generate MRO data and sign the certificate. The leaves contain the mechanics signature, the MRO data, e.g., the same values collected via the zk-DApp prototype, and a time stamp—the Merkle root results from a specific state of the spare part documentation history. The Merkle tree represents the local running account for MRO data capturing and is entirely maintained off-chain.

The architecture (Figure 5.8) ensures local and global data integrity. The operator is

responsible for proving local data integrity. Through Merkle proofs and blockchain receipts, users can track whether their submitted data was correctly included in the structure. The operator keeps its local Merkle tree with sensitive MRO data off-chain but maintains the Merkle root via blockchain. Users must trust the operator with their data privacy during transfer, but not with the integrity of maintaining correct account keeping (Sedlmeir, Völter, & Strüker 2022).

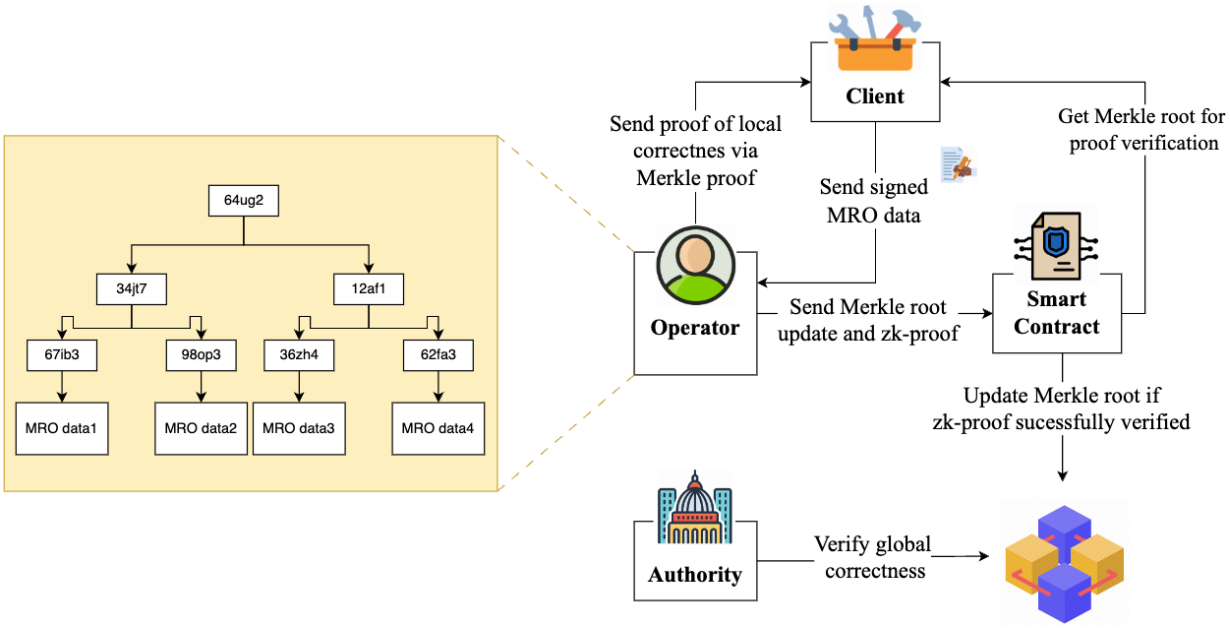


Figure 5.8: Zero-Knowledge Data Structure Architecture

Data integrity is not achieved through trust due to this architecture's introduction of zero-knowledge proofs. In order to decentralize the accounting for correct MRO data submission and to prevent fraud on the side of the operator, the use of Merkle proofs alone is not yet sufficient. It needs to be ensured that the generation of MRO data matches globally in the Merkle tree structure. In order to prove the authenticity of any newly batched update, the operator sends the suggestion for the new Merkle root to a smart contract as zero-knowledge proof during periodical Merkle tree updates. As it is known with zk-SNARKs, all parties must agree on a proving and a verifying key via MPC, i.e., the trusted set-up is initialized. Public inputs are previously correct Merkle root updates. The witness is the signed input submitted by the users and the entire Merkle tree. With this at hand, the operator can generate a zero-knowledge proof to be sent to the smart contract. The smart contract can then verify this proof through the verifying key and the previous Merkle root. Once the

proof is considered valid through most nodes participating, the Merkle root can be updated in the smart contract, creating global integrity and transparency for third-party audits.

This proposal for a zero-knowledge data structure provides the scope to securely manage the documentation of spare parts and certificate data and prove their authenticity by incorporating a zk-rollup-based architecture.

6 Conclusion

This research has implications for scholars and practitioners.

Efthymiou et al. (2022) provides an outlook for blockchain developers to create more familiarity with the technology, examine and implement widely shared use cases for MRO events, and facilitate standardization and data interoperability. This master thesis provides scholars and practitioners with a topical overview and practical implementations for the aviation industry.

At first, a systematic literature review on zero-knowledge proofs was performed. ZKPs consist of proof systems, which are extended with cryptographic tools and transformations to create argument systems. Adding ZK properties will yield zero-knowledge, succinct, non-interactive arguments of knowledge (zk-SNARKs). Different zk-SNARKs and variation protocols are examined and compared to create a current overview of the most practically used algorithms. The literature review finds that ZKPs are widely applied throughout various fields, grouping them into problem-solving domains, while ZKPs have not been discussed with regard to their applicability in the aviation industry. It concludes that current implementation practice evaluates PLONK as a promising zk-SNARK for decentralized applications. Higher-hanging fruits in ZKP research are obtained through developing quantum-resistant algorithms and broader adoptions of zk-Rollups and recursive zk-SNARKs.

The research question is further approached by establishing requirements from previous work within the aircraft maintenance, overhaul, and repair industry. The systematic literature review partially satisfies the first requirement of extending ZKP expertise within the RAPADO project. The zk-SNARKs proof and verification of a polynomial example satisfies the first requirement by providing further practical knowledge and application.

The second requirement is maintaining transparency while keeping specific competitive MRO data confidential. Although blockchain enables high transparency, further use of ZKPs achieves the compatibility of both. The zk-DApp for MRO data attestation and verification is implemented as a software artifact to demonstrate the functionality of PLONK in practice, suggesting a trusted process to verify aviation spare part documentation securely.

Further implications must be investigated to facilitate authenticity checks during back-to-birth traceability of MRO documentation. A zero-knowledge data structure is proposed as a third artifact to meet the requirement of fraud-preventive document authenticity checks. This architecture is based on Merkle trees and ZKPs and proposes to create secure and appropriate data formats for MRO documentation.

6.1 Discussion of Results

This research is bound to several limitations.

The topic of zero-knowledge proofs in blockchain is complex, bringing along resistance to change inside and outside the aviation industry. It is easier to identify appropriate use cases when the technology is well understood.

Specific prototypical software artifacts, e.g., decentralized applications, are implemented on test networks. It does not require any specific assumptions about the type of network the application will be running on. Still, how different network types affect the developed prototype's functionality must be investigated.

The use case for the zk-DApp needs further maturation to test whether the values and constraints are practical. The demonstration of technical functionalities was at focus. The communication channels between attesters and submitters are shared via browser links in the prototype for demonstration purposes. This approach needs to be made more practical and secure. The mechanism to notify attesters of newly entered values for attestations must be integrated into a broader communication platform with a more secure exchange of notifications for newly created values for attestation.

The zero-knowledge data structure architecture is designed and equipped with roles and process flows but still needs practical implementation and evaluation through security modeling, analysis, and tests among industry participants. The architecture needs to be prototypically incorporated into the zk-DApp and tested.

6.2 Outlook

Future work concluded from this thesis is discussed and sorted from lowest to highest-hanging fruits.

The zk-DApp uses the PLONK implementation in circom and snarkjs. Once the TURBOPLONK implementation is available, the application needs to be updated to use the efficiency increase compared to PLONK. All software artifacts developed within the project RAPADO must be integrated, i.e., the DApp for documentation storage and traceability and the zk-DApp for trustless documentation attestation and verification.

The zero-knowledge data structure architecture is a promising base layer for preliminary developed blockchain-based applications in the industry. It must be applied for future prototyping to create a consistent data structure for digitizing aviation spare part data. Furthermore, zero-knowledge proofs must be the focus of technology acceptance modeling in the aviation and MRO industry. Even though it is a technical topic, economic and social analyses concerning change management in the industry need to be performed.

Recursive zk-SNARKs become more relevant for this research, considering that millions of documentation events must be captured and verified, i.e., many proofs must be created and verified at once. New research outcomes in this field must be observed, analyzed, and applied to existing implementations once possible.

Bibliography

Ahmad, H., Wang, L., Hong, H., Li, J., Dawood, H., Ahmed, M., & Yang, Y. (2018). Primitives towards verifiable computation: a survey. *Frontiers of Computer Science*, *12*, 451–478.
https://doi.org/10.1007/s11704-016-6148-4.

Andreessen Horowitz a16z. (2023). *zkdocs.*
https://github.com/a16z/zkdocs, Last accessed on 2023-03-25.

Arora, S., Lund, C., Motwani, R., Sudan, M., & Szegedy, M. (1998). Proof Verification and the Hardness of Approximation Problems. *J. ACM*, *45*(3), 501–555.
https://doi.org/10.1145/278298.278306.

Babai, L. (1985). Trading group theory for randomness. In *Symposium on the theory of computing.*
https://doi.org/10.1145/22145.22192.

Babai, L., Fortnow, L., & Lund, C. (1991). Non-deterministic exponential time has two-prover interactive protocols. *Comput Complexity*(1), 3–40.
https://doi.org/10.1007/BF01200056.

Bansod, S., & Ragha, L. (2022). Challenges in making blockchain privacy compliant for the digital world: some measures. *Sădhană*, *47*(168).
https://doi.org/10.1007/s12046-022-01931-1.

Bellare, M., & Rogaway, P. (1993). Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st acm conference on computer and communications security* (p. 62–73). New York, NY, USA: Association for Computing Machinery.
https://doi.org/10.1145/168588.168596.

Benamara, O. (2022). Algorithms for elliptic curves. *Journal of Discrete Mathematical Sciences and Cryptography*, *25*(2), 455–462.
https://doi.org/10.1080/09720529.2020.1714891.

Ben-Or, M., Goldwasser, S., Kilian, J., & Wigderson, A. (1988). Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. In *Proceedings of the twentieth annual acm symposium on theory of computing* (p. 113–131). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/62212.62223.

Ben-Sasson, E., Bentov, I., Horesh, Y., & Riabzev, M. (2018). Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, *2018*, 46. https://eprint.iacr.org/2018/046.pdf.

Ben-Sasson, E., Chiesa, A., & Spooner, N. (2016). Interactive Oracle Proofs. In *Proceedings, part ii, of the 14th international conference on theory of cryptography - volume 9986* (p. 31–60). Berlin, Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-662-53644-5_2.

Ben-Sasson, E., Goldberg, L., Kopparty, S., & Saraf, S. (2019). DEEP-FRI: Sampling outside the box improves soundness.. https://doi.org/10.48550/arXiv.1903.12243.

Birrell, E., & Vadhan, S. (2009). Composition of Zero-Knowledge Proofs with Efficient Provers. Cryptology ePrint Archive, Paper 2009/604. https://doi.org/10.1007/978-3-642-11799-2_34.

Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., & Paneth, O. (2022, jul). Succinct Non-Interactive Arguments via Linear Interactive Proofs. *J. Cryptol.*, *35*(3). https://doi.org/10.1007/s00145-022-09424-4.

Blum, M., Feldman, P., & Micali, S. (1991). NONINTERACTIVE ZERO-KNOWLEDGE*. In (pp. 1084–1118). https://doi.org/10.1137/0220068.

Brassard, G., Chaum, D., & Crépeau, C. (1988). Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, *37*(2), 156–189. https://doi.org/10.1016/0022-0000(88)90005-0.

Buterin, V. (2019). *Understanding PLONK.* https://www.vitalik.ca/general/2019/09/22/plonk.html, Last accessed on 2023-03-08.

Buterin, V. (2021). *Incomplete Guide to Rollups.* https://vitalik.ca/general/2021/01/05/rollup.html, Last accessed on 2023-05-12.

Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., & Maxwell, G. (2018, 05). Bulletproofs: Short Proofs for Confidential Transactions and More. In (p. 315-334). http://dx.doi.org/10.1109/SP.2018.00020.

Bünz, B., Fisch, B., & Szepieniec, A. (2019). Transparent SNARKs from DARK Compilers. *Cryptology ePrint Archive, Paper 2019/1229*. http://dx.doi.org/10.1007/978-3-030-45721-1_24.

Canetti, R., Goldreich, O., & Halevi, S. (2004). The Random Oracle Methodology, Revisited. *J. ACM*, *51*(4), 557–594. https://doi.org/10.1145/1008731.1008734.

Chatzi, A. (2019). The Diagnosis of Communication and Trust in Aviation Maintenance (DiCTAM) Model. *Aerospace*, *6*(11), 1-23. http://dx.doi.org/10.3390/aerospace6110120.

Chen, T., Lu, H., Kunpittaya, T., & Luo, A. (2022). A Review of zk-SNARKs.. https://doi.org/10.48550/arXiv.2202.06877.

Chung, H., Han, K., Ju, C., Kim, M., & Seo, J. H. (2022). Bulletproofs+: Shorter Proofs for a Privacy-Enhanced Distributed Ledger. *IEEE Access*, *10*, 42067–42082. https://doi.org/10.1109/ACCESS.2022.3167806.

Deng, C., You, L., Tang, X., Hu, G., & Gao, S. (2022). Cuproof: Range Proof with Constant Size. *Entropy*, *24*(3). https://doi.org/10.3390/e24030334.

Eberhardt, J., & Tai, S. (2018). ZoKrates - Scalable Privacy-Preserving Off-Chain Computations. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 1084–1091. http://dx.doi.org/10.1109/Cybermatics_2018.2018.00199.

Efthymiou, M., McCarthy, K., Markou, C., & O/'Connell, J. F. (2022). An Exploratory Research on Blockchain in Aviation: The Case of Maintenance, Repair and Overhaul (MRO) Organizations. *Sustainability*, *14*(5).

https://doi.org/10.3390/su14052643.

Ethereum Foundation. (2023a). *Scaling Overview.* https://ethereum.org/en/developers/docs/scaling/, Last accessed on 2023-05-12.

Ethereum Foundation. (2023b). *Scaling Overview.* https://ethereum.org/en/developers/docs/scaling/validium/, Last accessed on 2023-05-12.

Fiat, A., & Shamir, A. (1986). How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Annual international cryptology conference.* https://doi.org/10.1007/3-540-47721-7_12.

Fortnow, L., Rompel, J., & Sipser, M. (1994). On the Power of Multi-Prover Interactive Protocols. *tcsa*, *134*, 545–557. https://dx.doi.org/10.1016/0304-3975(94)90251-8.

Gabizon, A., Williamson, Z. J., & Ciobotaru, O. (2019). PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. *Cryptology ePrint Archive, Paper 2019/953*. https://eprint.iacr.org/2019/953.

Gennaro, R., Gentry, C., Parno, B., & Raykova, M. (2012). Quadratic Span Programs and Succinct NIZKs without PCPs. *Cryptology ePrint Archive, Paper 2012/215*. https://doi.org/10.1007/978-3-642-38348-9_37.

Godden, T., De Smet, R., Debruyne, C., Vandervelden, T., Steenhaut, K., & Braeken, A. (2022). Circuitree: A Datalog Reasoner in Zero-Knowledge. *IEEE Access*, *10*, 21384–21396. https://dx.doi.org/10.1109/ACCESS.2022.3153366.

Goldreich, O., Micali, S., & Wigderson, A. (1987). How to Play ANY Mental Game. In (p. 218–229). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/28395.28420.

Goldwasser, S., Kalai, Y. T., & Rothblum, G. N. (2008). Delegating Computation: Interactive Proofs for Muggles. In (p. 113–122). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1374376.1374396.

Goldwasser, S., Micali, S., & Rackoff, C. (1985). The Knowledge Complexity of Interactive Proof-Systems. In *Proceedings of the seventeenth annual acm symposium on theory of computing* (p. 291–304). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/22145.22178.

Goldwasser, S., & Sipser, M. (1986). Private Coins versus Public Coins in Interactive Proof Systems. In *Proceedings of the eighteenth annual acm symposium on theory of computing* (p. 59–68). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/12130.12137.

Gong, Y., Jin, Y., Li, Y., Liu, Z., & Zhu, Z. (2022). Analysis and comparison of the main zero-knowledge proof scheme. In *2022 international conference on big data, information and computer network (bdicn)* (pp. 366–372). https://doi.org/10.1109/BDICN55575.2022.00074.

Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., & Schofnegger, M. (2019). Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. Cryptology ePrint Archive, Paper 2019/458. https://eprint.iacr.org/2019/458.

Groth, J. (2010). Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In *International conference on the theory and application of cryptology and information security.* https://doi.org/10.1007/978-3-642-17373-8_19.

Groth, J. (2016). On the Size of Pairing-Based Non-interactive Arguments. *IACR Cryptol. ePrint Arch.*, *2016*, 260. https://doi.org/10.1007/978-3-662-49896-5_11.

Guo, C., You, L., & Hu, G. (2022). A Novel Biometric Identification Scheme Based on Zero-Knowledge Succinct Noninteractive Argument of Knowledge. *Hindawi*. https://doi.org/10.1155/2022/2791058.

Huang, H., Zhu, P., Xiao, F., Sun, X., & Huang, Q. (2020). A blockchain-based scheme for privacy-preserving and secure sharing of medical data. *Computers & Security*, *99*, 102010. https://doi.org/10.1016/j.cose.2020.102010.

iden3. (2019). *iden3-docs.*
https://github.com/iden3/iden3-docs/blob/master/source/docs/iden3_ethcc_presentation.pdf, Last accessed on 2023-04-03.

iden3. (2023a). *circom-docs.*
https://github.com/iden3/circom, Last accessed on 2023-04-03.

iden3. (2023b). *snarkjs.*
https://github.com/iden3/snarkjs, Last accessed on 2023-04-03.

Insider Inc. (2019). *31 pages vanished from the maintenance log of the Lion Air Boeing 737 Max before it crashed, investigators said.*
https://www.businessinsider.com/lion-air-31-maintenance-pages-missing-fatal-737-max-investigators-2019-10, Last accessed on 2022-02-24.

Kanagamani, V., & Karuppiah, M. (2021). Zero knowledge based data deduplication using in-line Block Matching protocolfor secure cloud storage. *Turkish Journal of Electrical Engineering and Computer Sciences*, *29*(4), 2067–2083.
https://doi.org/10.3906/elk-2010-133.

Kate, A., Zaverucha, G. M., & Goldberg, I. (2010). Constant-Size Commitments to Polynomials and Their Applications. In *International conference on the theory and application of cryptology and information security.*
https://doi.org/10.1007/978-3-642-17373-8_11.

Katz, J., Kolesnikov, V., & Wang, X. (2018). Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In *Proceedings of the 2018 acm sigsac conference on computer and communications security* (p. 525–537). New York, NY, USA: Association for Computing Machinery.
https://doi.org/10.1145/3243734.3243805.

Koblitz, N., & Menezes, A. (2015). The Random Oracle Model: A Twenty-Year Retrospective..
https://eprint.iacr.org/2015/140.

Li, H., & Xue, W. (2021). A Blockchain-Based Sealed-Bid e-Auction Scheme with Smart Contract and Zero-Knowledge Proof. *Hindawi*.
https://doi.org/10.1155/2021/5523394.

Liu, W., Wang, X., Peng, W., & Xing, Q. (2019). Center-Less Single Sign-On With Privacy-Preserving Remote Biometric-Based ID-MAKA Scheme for Mobile Cloud Computing Services. *IEEE Access*, *7*, 137770–137783. http://dx.doi.org/10.1109/ACCESS.2019.2942987.

Liu, X., Xu, Y., Xu, G., & Li, B. (2022). Research on the Millionaires' Problem under the Malicious Model Based on the Elliptic Curve Cryptography. *Hindawi*, *2022*, 1–11. https://doi.org/10.1155/2022/6923610.

Lund, C., Fortnow, L., Karloff, H., & Nisan, N. (1992). Algebraic Methods for Interactive Proof Systems. *J. ACM*, *39*(4), 859–868. https://doi.org/10.1145/146585.146605.

Luong, D. A., & Park, J. H. (2022). Privacy-Preserving Blockchain-Based Healthcare System for IoT Devices Using zk-SNARK. *IEEE Access*, *10*, 55739–55752. https://doi.org/10.1109/ACCESS.2022.3177211.

Major, W., Buchanan, W. J., & Ahmad, J. (2020). An authentication protocol based on chaos and zero knowledge proof. *Nonlinear Dyn*, *99*, 3065–3087. https://doi.org/10.1007/s11071-020-05463-3.

Maller, M., Bowe, S., Kohlweiss, M., & Meiklejohn, S. (2019). Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings. In *Proceedings of the 2019 acm sigsac conference on computer and communications security* (p. 2111–2128). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3319535.3339817.

Matter Labs. (2022). *Zk Rollup Architecture.* https://docs.zksync.io/userdocs/tech/#congested-mainnet, Last accessed on 2023-05-12.

Micali, S. (2000). Computationally Sound Proofs. *SIAM Journal on Computing*, *30*(4), 1253–1298. https://doi.org/10.1137/S0097539795284959.

Morais, E., Koens, T., van Wijk, C., & Koren, A. (2019). A Survey on Zero Knowledge Range Proofs and Applications.. https://doi.org/10.1007/s42452-019-0989-z.

Munivel, E., & Kannammal, A. (2019). New Authentication Scheme to Secure against the Phishing Attack in the Mobile Cloud Computing. *Hindawi*, *2019*, 1–11. https://doi.org/10.1155/2019/5141395.

Nitulescu, A. (2019). A Gentle Introduction to SNARKs.. https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf.

Parno, B., Howell, J., Gentry, C., & Raykova, M. (2016). Pinocchio: Nearly Practical Verifiable Computation. *Commun. ACM*, *59*(2), 103–112. https://doi.org/10.1145/2856449.

Querejeta-Azurmendi, I. n., Arroyo Guardeño, D., Hernández-Ardieta, J. L., & Hernández Encinas, L. (2020). NetVote: A Strict-Coercion Resistance Re-Voting Based Internet Voting Scheme with Linear Filtering. *Mathematics*, *8*(9). https://doi.org/10.3390/math8091618.

Reingold, O., Rothblum, G. N., & Rothblum, R. D. (2016). Constant-Round Interactive Proofs for Delegating Computation. In *Proceedings of the forty-eighth annual acm symposium on theory of computing* (p. 49–62). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2897518.2897652.

Sedlmeir, J., Lautenschlager, J., Fridgen, G., & Urbach, N. (2022). The transparency challenge of blockchain in organizations. *Electronic Markets*, *32*. https://dx.doi.org/10.1007/s12525-022-00536-0.

Sedlmeir, J., Völter, F., & Strüker, J. (2022). The next Stage of Green Electricity Labeling: Using Zero-Knowledge Proofs for Blockchain-Based Certificates of Origin and Use. *SIGEnergy Energy Informatics Review*, *1*(1), 20–31. https://doi.org/10.1145/3508467.3508470.

Setty, S. (2020, 08). Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup. In (pp. 704–737). http://dx.doi.org/10.1007/978-3-030-56877-1_25.

Shamir, A. (1992). IP = PSPACE. *J. ACM*, *39*(4), 869–877. https://doi.org/10.1145/146585.146609.

SITA. (2022). *MRO Blockchain Services - A trusted framework and new cost efficiencies Chapter 2 - the MRO perspective.*

https://www.sita.aero/resources/White-papers/mro-blockchain/, Last accessed on 2022-11-24.

Thaler, J. (2023). *Proofs, Arguments, and Zero-Knowledge.* https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf, Last accessed on 2023-06-11.

Vidick, T., & Zhang, T. (2020). Classical zero-knowledge arguments for quantum computations. *Quantum*, *4*, 266. https://doi.org/10.22331/q-2020-05-14-266.

Vom Brocke, J., Winter, R., Hevner, A., & Maedche, A. (2020). Special issue editorial–accumulation and evolution of design knowledge in design science research: a journey through time and space. *Journal of the Association for Information Systems*, *21*(3), 9. http://dx.doi.org/10.17705/1jais.00611.

Šimunić, S., Bernaca, D., & Lenac, K. (2021). Verifiable Computing Applications in Blockchain. *IEEE Access*, *9*, 156729–156745. https://doi.org/10.1109/ACCESS.2021.3129314.

Walton-Pocock, T. (2019). *PLONK Benchmarks I — 2.5x faster than Groth16 on MiMC.* https://medium.com/aztec-protocol/plonk-benchmarks-2-5x-faster-than-groth16-on-mimc-9e1009f96dfe, Last accessed on 2023-05-20.

Wang, D., Zhao, J., & Mu, C. (2021). Research on Blockchain-Based E-Bidding System. *Applied Sciences*, *11*(9). https://doi.org/10.3390/app11094011.

Wang, H., Liu, Z., Ge, C., Sakurai, K., & Su, C. (2022). A Privacy-Preserving Data Feed Scheme for Smart Contracts. *IEICE Trans. Inf. Syst.*, *105-D*, 195–204. https://doi.org/10.1587/transinf.2021BCI0001.

Watrous, J. (2003, 02). Limits on the Power of Quantum Statistical Zero-Knowledge. *Foundations of Computer Science, 1975., 16th Annual Symposium on.* http://dx.doi.org/10.1109/SFCS.2002.1181970.

Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Q.*, *26*.

https://doi.org/10.4236/ce.2019.1013260.

Wickboldt, C., & Kliewer, N. (2018). Blockchain zur dezentralen Dokumentation von Werkstattereignissen in der Luftfahrtindustrie. *HMD Praxis der Wirtschaftsinformatik*, *55*(6), 1297–1310.
https://dx.doi.org/10.1365/s40702-018-00452-y.

Wickboldt, C., & Kliewer, N. (2019). Blockchain for Workshop Event Certificates - a Proof of Concept in the Aviation Industry. In *Ecis.*
https://aisel.aisnet.org/ecis2019_rp/35/.

Wickboldt, C., Meise, C., & Kliewer, N. (2020). Decentralized Maintenance Event Documentation with Hyperledger Fabric. *15th International Conference on Wirtschaftsinformatik, March 08-11, 2020, Potsdam, Germany.*
http://dx.doi.org/10.30844/wi_2020_b2-wickboldt.

Wilde, T., & Hess, T. (2007). Forschungsmethoden der Wirtschaftsinformatik. *Wirtschaftsinformatik*, *49*(4), 280–287.
https://doi.org/10.1007/s11576-007-0064-z.

Xie, H., & Yang, L. (2019). Witness indistinguishability and witness hiding against quantum attacks. *IET Information Security*, *13*.
https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-ifs.2018.5460.

Xue, Y., & Wang, J. (2022). Design of a Blockchain-Based Traceability System with a Privacy-Preserving Scheme of Zero-Knowledge Proof. *Hindawi.*
https://doi.org/10.1155/2022/5842371.

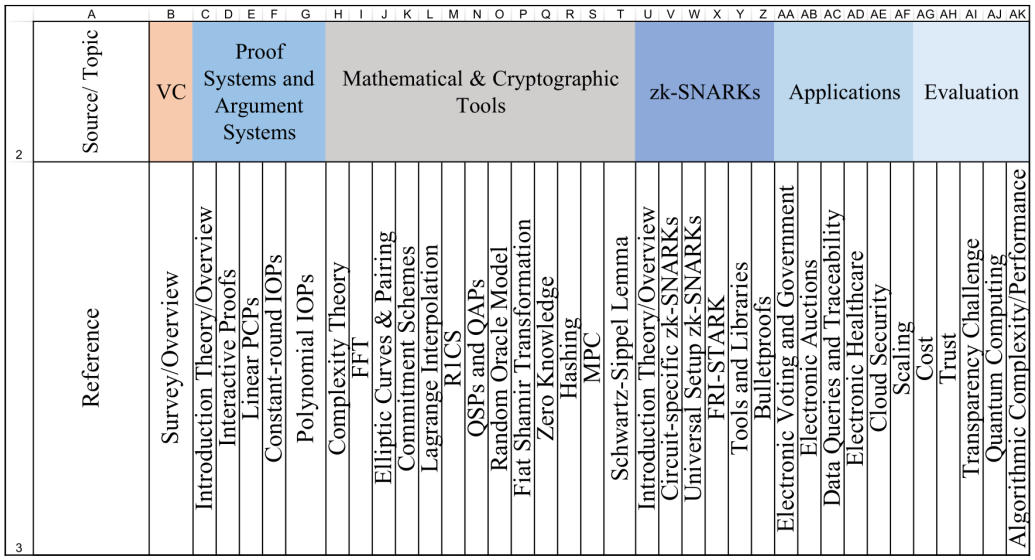
Zedel, J., & Kliewer, N. (2022). Maintaining Privacy in a Blockchain-based Platform: Application to Aviation Industry MRO Documentation. *HMD Praxis der Wirtschaftsinformatik*(59).
https://doi.org/10.1365/s40702-022-00903-7.

Zhang, Y., Wang, S., Zhang, X., Dong, J., Mao, X., Long, F., ... Sun, G. (2021). PipeZK: Accelerating Zero-Knowledge Proof with a Pipelined Architecture. In *2021 acm/ieee 48th annual international symposium on computer architecture (isca)* (pp. 416–428).
https://doi.org/10.1109/ISCA52012.2021.00040.

Zheng, H., You, L., & Hu, G. (2022). A novel insurance claim blockchain scheme based on zero-knowledge proof technology. *Computer Communications*, *195*, 207–216. https://doi.org/10.1016/j.comcom.2022.08.007.

Appendix

Full Concept Matrix



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
4	Ahmad, H., Wang, L., Hong, H., Li, J., Dawood, H., Ahmed, M., & Yang, Y. (2018)	x		x	x				x								x																				
5	Arora, S., Lund, C., Motwani, R., Sudan, M., & Szegedy, M. (1998)							x																													
6	Babai, L. (1985)			x																																	
7	Babai, L., Fortnow, L., & Lund, C. (1991)			x																																	
8	Bansod, S., & Ragha, L. (2022)																										x							x				
9	Bellare, M., & Rogaway, P. (1993)														x																							
10	Ben-Or, M., Goldwasser, S., Kilian, J., & Wigderson, A. (1988			x																																	
11	Ben-sasson, E., Chiesa, A., & Spooner, N. (2016			x																																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
12	Ben-Sasson, E., Chiesa, A., & Spooner, N. (2016)						x																														
13	Benamara, O. (2022)						x																														
14	Birrell, E., & Vadhan, S. (2009)																x																				
15	Blum, M., Feldman, P., & Micali, S. (1991)																				x																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
16	Brassard, G., Chaum, D., & Crépeau, C. (1988).		x																																		
17	Bünz, B., Fisch, B., & Szepieniec, A. (2019).						x																	x														
18	Buterin, V. (2019).																						x															
19	Canetti, R., Goldreich, O., & Halevi, S. (2004).														x																							

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
20	Chen, T., Lu, H., Kunpittaya, T., & Luo, A. (2022)																				x																
21	Eberhardt, J., & Tai, S. (2018).																								x												
22	Fiat, A., & Shamir, A. (1986).															x																					
23	Fortnow, L., Rompel, J., & Sipser, M. (1994)																																				

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
24	Gabizon, A., Williamson, Z. J., & Ciobotaru, O. (2019).																						x														
25	Gennaro, R., Gentry, C., Parno, B., & Raykova, M. (2012).													x																								
26	Smet, R., Debruyne, C., Vandervelden, T., Steenhaut, K., & Braeken, A. (2022)																												x								
27	Goldreich, O., Micali, S., & Wigderson, A. (1987)							x																														

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
28	Goldwasser, S., & Sipser, M. (1986)			x																																
29	Goldwasser, S., Kalai, Y. T., & Rothblum, G. N. (2008).			x																																
30	Goldwasser, S., Micali, S., & Rackoff, C. (1985).			x																																
31	Gong, Y., Jin, Y., Li, Y., Liu, Z., & Zhu, Z. (2022).																																			x

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
32	Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., & Schofnegger, M. (2019)																	x																			
33	Groth, J. (2010)									x		x	x	x			x				x	x															
34	Groth, J. (2016)								x	x		x	x	x				x		x		x																
35	Guo, C., You, L., & Hu, G. (2022)																										x										

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
36	Huang, H., Zhu, P., Xiao, F., Sun, X., & Huang, Q. (2020).																													x							
37	Kanagamani, V., & Karuppiah, M. (2021).																														x						
38	Kate, A., Zaverucha, G. M., & Goldberg, I. (2010).										x																										
39	Katz, J., Kolesnikov, V., & Wang, X. (2018).																																			x	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
40	Koblitz, N., & Menezes, A. (2015).														x																						
41	Li, H., & Xue, W. (2021). A																											x									
42	Liu, W., Wang, X., Peng, W., & Xing, Q. (2019).																														x						
43	Lund, C., Fortnow, L., Karloff, H., & Nisan, N. (1992).			x																																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
44	Luong, D. A., & Park, J. H. (2022)																													x							
45	Major, W., Buchanan, W. J., & Ahmad, J. (2020).																														x						
46	Maller, M., Bowe, S., Kohlweiss, M., & Meiklejohn, S. (2019).																						x														
47	Micali, S. (2000).		x																																		

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
48	Morais, E., Koens, T., van Wijk, C., & Koren, A. (2019)																															x					
49	Munivel, E., & Kannammal, A. (2019).																														x						
50	Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. (2022)			x	x														x			x		x		x											
51	Nitulescu, A. (2019)																				x																	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
52	Parno, B., Howell, J., Gentry, C., & Raykova, M. (2016).																					x															
53	Azurmendi, I. n., Arroyo Guardeño, D., Hernández-Ardieta, J. L., & Hernán-dez Encinas, L.																										x									
54	Reingold, O., Rothblum, G. N., & Rothblum, R. D. (2016).					x	x																													
55	Sedlmeir, J., Lautenschlager, J., Fridgen, G., & Urbach, N. (2022).																																	x	x		

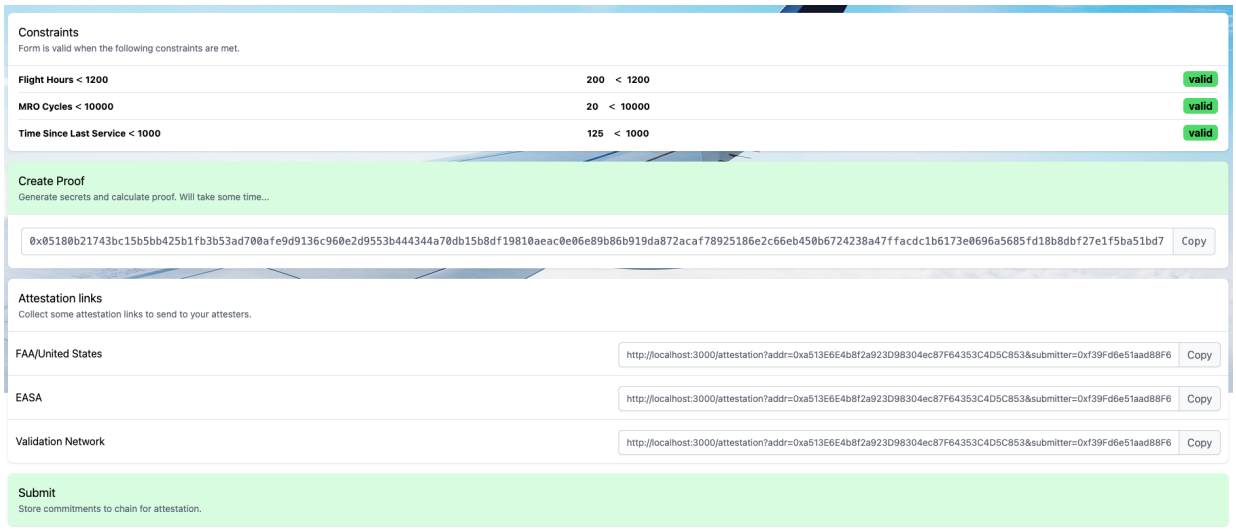
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
56	Sedlmeir, J., Völter, F., & Strüker, J. (2022).																															x			x		
57	Setty, S. (2020, 08)		x	x																	x																
58	Shamir, A. (1992)							x																														
59	Šimunić, S., Bernaca, D., & Lenac, K. (2021).	x																			x												x				

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
60	Thaler, J. (2023).	x																			x																
61	Vidick, T., & Zhang, T. (2020).																																			x	
62	Wang, D., Zhao, J., & Mu, C. (2021).																											x									
63	Wang, H., Liu, Z., Ge, C., Sakurai, K., & Su, C. (2022).																													x							

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
64	Xie, H., & Yang, L. (2019).																																			x	
65	Xue, Y., & Wang, J. (2022).																													x							
66	Zhang, Y., Wang, S., Zhang, X., Dong, J., Mao, X., Long, F., Sun, G. (2021)																																x	x			
67	Zheng, H., You, L., & Hu, G. (2022).																													x							

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
68	Ben-Sasson et al. (2018)						x	x	x		x					x		x						x													x
69	Chung et al. (2020)																									x											
70	Deng et al. (2020)																									x											
71	Bünz e al. (2019)																									x											

zk-DApp for MRO Attestations (Additional Screenshots)



I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Berlin, 12 June 2023

Signature

(signature)