

# **Secure Localization in Wireless Sensor Networks**

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades

“Doctor rerum naturalium”

der Georg-August-Universität Göttingen

im Promotionsprogramm Computer Science (PCS)

der Georg-August University School of Science (GAUSS)

vorgelegt von

Arne Bochem

aus Bad Mergentheim

Göttingen, Februar 2022

### Betreuungsausschuss

Prof. Dr. Dieter Hogrefe,  
Telematics Group, Institut für Informatik  
Georg-August-Universität Göttingen

Prof. Dr. Omar Alfandi,  
College of Technological Innovations  
Zayed University, Abu Dhabi

### Mitglieder der Prüfungskommission

Referent: Prof. Dr. Dieter Hogrefe,  
Telematics Group, Institut für Informatik,  
Georg-August-Universität Göttingen

Korreferent: Prof. Dr. Xiaoming Fu,  
Computer Networks Group, Institut für Informatik,  
Georg-August-Universität Göttingen

### Weitere Mitglieder der Prüfungskommission

Prof. Dr. Omar Alfandi,  
College of Technological Innovations  
Zayed University, Abu Dhabi

Prof. Dr.-Ing. Marcus Baum,  
Data Fusion Group, Institut für Informatik  
Georg-August-Universität Göttingen

Prof. Dr. Carsten Damm,  
Theoretical Computer Science Group, Institut für Informatik  
Georg-August-Universität Göttingen

Prof. Dr. Florin Manea,  
Fundamentals of Computer Science Group, Institut für Informatik  
Georg-August-Universität Göttingen

### Tag der mündlichen Prüfung

17. März 2022

## Acknowledgement

First, I would like to express my gratitude to Prof. Dr. Dieter Hogrefe for his supervision and support throughout my studies, to Prof. Dr. Omar Alfandi for his guidance and help, as well as to Prof. Dr. Xiaoming Fu who was willing to join in as second examiner even at a late stage.

Furthermore, I would like to thank Prof. Dr.-Ing. Marcus Baum, Prof. Dr. Carsten Damm and Prof. Dr. Florin Manea for joining as further members of the examination committee.

My thanks and gratitude go out to all my colleagues and friends whom I made at the institute for the many interesting, fun and fruitful discussions we had.

Finally, I wish to thank my mother and father for their unwavering supportiveness.

## Abstract

*With the growing popularity of the Internet of Things, Wireless Sensor Networks also only grow more and more common in various different forms. However, sensor data is often only useful in connection with information about where it comes from. For this reason, localization schemes that allow sensor nodes to localize their positions are a very active field of research. As schemes are refined, localization results grow increasingly more accurate, but it also becomes more and more important to make localization approaches more robust against malfunctioning or malicious nodes in the network, as well as network scale attacks. This thesis presents two approaches, Unchained and Rechained, to monetarily disincentivize the creation of Sybil identities in decentralized networks, mitigating a common class of network level attacks against localization schemes. Furthermore, Robustness Enhanced Sensor Assisted Monte Carlo Localization (RESA-MCL) is introduced, evaluated and compared against previous comparable schemes. Evaluation is performed in simulations without attacks and under three different attack models that are introduced for the application field of Wireless Sensor Networks. RESA-MCL outperforms other approaches both without and with attacks and performs well in both low and high anchor density scenarios (e.g. a localization error of 0.5 is reached at an anchor density of 0.33), reaching a localization error up to 48% lower than that of a recent comparable approach at a similar anchor density. It is shown to be much more robust than other approaches under attacks while computational complexity is barely increased.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Localization in Wireless Sensor Networks . . . . .	1
1.2	Motivation . . . . .	2
1.2.1	Applicability . . . . .	2
1.2.2	Hardware and power requirements . . . . .	3
1.2.3	Robustness regarding network integrity . . . . .	3
1.2.4	Robustness regarding malfunctioning and malicious anchor nodes . . . . .	4
1.2.5	Goal . . . . .	4
1.3	Research questions . . . . .	4
1.4	Thesis structure . . . . .	5
1.5	Copyright and contribution . . . . .	5
<b>2</b>	<b>Unchained Identities</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Supplementary Literature and Related Works . . . . .	8
2.2.1	Blockchain Technology . . . . .	9
2.2.2	Related Works . . . . .	10
2.3	Unchained Identities in Mobile Ad-hoc Networks (MANETs) . . . . .	10
2.3.1	Creating a New Identity . . . . .	10
2.3.2	Identity Validation . . . . .	11
2.4	Parameter Choice . . . . .	13
2.4.1	Network Parameters . . . . .	13
2.4.2	Pricing an Identity . . . . .	15
2.5	Handling Difficulty Adjustments . . . . .	15
2.5.1	Maximum Seen Difficulty . . . . .	16
2.5.2	Bundled Updates . . . . .	16
2.5.3	Majority Vote . . . . .	17
2.6	Evaluation . . . . .	17
2.6.1	Bitcoin Difficulty Analysis . . . . .	18

2.6.2	Bitcoin Price Analysis . . . . .	19
2.6.3	Ethereum Difficulty Analysis . . . . .	19
2.7	Conclusion and Future Work . . . . .	20
<b>3</b>	<b>Rechained</b>	<b>23</b>
3.1	Introduction . . . . .	23
3.2	Supplementary Literature and Related Work . . . . .	25
3.2.1	Self-Sovereign Identities and Decentralized Identifiers . . . . .	25
3.2.2	Blockchain Technology . . . . .	26
3.2.3	Related Work . . . . .	27
3.3	Protocol Specification . . . . .	28
3.3.1	Creating Identities . . . . .	29
3.3.2	Verifying Identities . . . . .	31
3.3.3	Revoking Identities . . . . .	31
3.3.4	Distributed Identities . . . . .	32
3.4	Parameter Choices and Updates . . . . .	32
3.4.1	Network Parameters . . . . .	32
3.4.2	Price Considerations . . . . .	34
3.4.3	Updating Parameters . . . . .	35
3.5	Protocol Formalization . . . . .	37
3.5.1	Colored Petri Nets . . . . .	37
3.5.2	Modeling Strategy . . . . .	38
3.5.3	AOM Model . . . . .	39
3.5.4	Mapping AOM Models to CPN Models . . . . .	40
3.5.5	Protocol Semantics . . . . .	43
3.6	Evaluation and Discussion . . . . .	44
3.6.1	Bitcoin Price and Difficulty Analysis . . . . .	45
3.6.2	Ethereum Price and Difficulty Analysis . . . . .	47
3.6.3	Transaction Fee Analysis . . . . .	49
3.6.4	CPN State-Space Analysis . . . . .	50
3.6.5	Proof-of-Concept Implementation . . . . .	51
3.6.6	Discussion . . . . .	51
3.7	Conclusion and Future Work . . . . .	52
3.8	Appendix: Rechained Protocol Formalization . . . . .	53
<b>4</b>	<b>RESA-MCL</b>	<b>55</b>
4.1	Introduction . . . . .	55
4.2	Related Works . . . . .	58
4.2.1	Range-based approaches . . . . .	58
4.2.2	Range-free approaches . . . . .	59

4.2.3	AI-based approaches . . . . .	60
4.2.4	Security-aware approaches . . . . .	61
4.3	Localization Scheme . . . . .	62
4.3.1	Notation . . . . .	63
4.3.2	MCL . . . . .	63
4.3.3	SA-MCL . . . . .	66
4.3.4	RESA-MCL . . . . .	66
4.4	Evaluation . . . . .	71
4.4.1	Experimental setup . . . . .	71
4.4.2	Baseline . . . . .	72
4.4.3	Optimal particle subsetting . . . . .	73
4.4.4	Performance under attacks . . . . .	73
4.4.5	Ablation experiments . . . . .	75
4.4.6	Comparison under anchor density . . . . .	76
4.5	Conclusion and future works . . . . .	76
<b>5</b>	<b>Conclusions and future works</b>	<b>79</b>
	<b>Bibliography</b>	<b>91</b>





# List of Figures

2.1	Blockchain structure, adapted from [1]. . . . .	9
2.2	Creating a new identity. . . . .	11
2.3	Overview of the validation process. . . . .	12
2.4	Average daily price in USD and block difficulty level between Dec. 2016 and Nov. 2017 (Source: [2–4]). . . . .	18
3.1	General blockchain structure – based on [1]. . . . .	26
3.2	Creating a new identity proof . . . . .	29
3.3	Overview of the validation process . . . . .	31
3.4	Selection of Agent-Oriented Modeling (AOM) notation elements . . . . .	39
3.5	Rechained top-level AOM goal model . . . . .	40
3.6	Mapping a behavior interface model to a CPN model . . . . .	42
3.7	Rechained CPN model . . . . .	43
3.8	Average daily price of bitcoin in USD and block difficulty level . . . . .	45
3.9	Average daily price of ether in USD and block difficulty level . . . . .	48
4.1	Example illustrating the particle subsetting process (S.4.3.4) with two anchors, one unknown node $U$ and $k \in \{1, 2, \dots, 8\}, t = 5, j \in \{1, 2\}$ . . . . .	68
4.2	Comparison at different communication ranges without attack. . . . .	72
4.3	Evaluation of particle subset sizes without attacks and $s_\phi = 16$ . . . . .	72
4.4	Evaluation of particle subset sizes under fixed position attack with $s_\phi = 4$ . . . . .	72
4.5	Biased position attack. . . . .	72
4.6	Random position attack. . . . .	73
4.7	Fixed position attack. . . . .	73
4.8	Fixed position attack with 50 anchor nodes. . . . .	74
4.9	Ablation experiments. . . . .	74



# List of Tables

2.1	Affected starting dates after which the Bitcoin price drops below a certain percentage of the given day's price. . . . .	19
3.1	Parameters . . . . .	33
3.2	Behavioral interfaces of activities for Rechaind . . . . .	41
3.3	Notation mapping CPN to AOM . . . . .	42
3.4	Exemplary acronyms, names and description of token colors of the Rechaind CPN model – extension of [5] . . . . .	44
3.5	Bitcoin price decline analysis . . . . .	47
3.6	Ethereum price decline analysis . . . . .	49
3.7	State-space analysis results of the Rechaind CPN model . . . . .	50
4.1	Symbols with references to sections and listings. . . . .	63
4.2	Error comparison between RESA-MCL and MCL-DE without attacks. . . . .	75



# List of Listings

4.1	MCL algorithm. . . . .	64
4.2	Particle filter condition function. . . . .	65
4.3	SA-MCL algorithm. . . . .	66
4.4	Particle dead reckoning. . . . .	67
4.5	Anchor position plausibility check function. . . . .	69
4.6	Anchor distrust point update function. . . . .	69
4.7	RESA-MCL algorithm. . . . .	70



# List of Abbreviations

<b>AI</b>	Artificial Intelligence . . . . .	58
<b>ANFIS</b>	Adaptive Neural Fuzzy Inference System . . . . .	60
<b>AoA</b>	Angle of Arrival . . . . .	3
<b>AOM</b>	Agent-Oriented Modeling . . . . .	ix
<b>APIT</b>	Approximate Point in Triangle . . . . .	61
<b>CCDV-Hop</b>	Centralized Connectivity based DV-Hop . . . . .	59
<b>CNN</b>	Convolutional Neural Network . . . . .	60
<b>CPN</b>	Colored Petri Net . . . . .	25
<b>CPN-ML</b>	Colored Petri Net Markup Language . . . . .	38
<b>CSI</b>	Channel State Information . . . . .	60
<b>DCDV-Hop</b>	Distributed Connectivity based DV-Hop . . . . .	59
<b>DE</b>	Differential Evolution . . . . .	59
<b>DID</b>	Decentralized Identifier . . . . .	6
<b>DV-Hop</b>	Distance Vector-Hop . . . . .	59
<b>GPS</b>	Global Positioning System . . . . .	56
<b>HMAC</b>	Hash Message Authentication Code . . . . .	11
<b>IMU</b>	Inertial Measurement Unit . . . . .	59
<b>IoT</b>	Internet of Things . . . . .	1
<b>LUDM</b>	Localization using Uncertain Data Mapping . . . . .	58
<b>MANET</b>	Mobile Ad-hoc Network . . . . .	v
<b>MCL</b>	Monte Carlo Localization . . . . .	56
<b>MNDC</b>	Malicious Node Detection algorithm based on Clustering and consistency evaluation . . . . .	61
<b>NLoS</b>	Non Line of Sight . . . . .	56
<b>P2P</b>	Peer to Peer . . . . .	8
<b>PoW</b>	Proof of Work . . . . .	4
<b>RESA-MCL</b>	Robustness Enhanced Sensor Assisted Monte Carlo Localization . . . . .	5
<b>RSS</b>	Received Signal Strength . . . . .	2
<b>SA-MCL</b>	Sensor Assisted Monte Carlo Localization . . . . .	57

<b>SCC</b>	Strongly Connected Component . . . . .	50
<b>SF-APIT</b>	Sybil-free APIT . . . . .	3
<b>SSI</b>	Self-Sovereign Identity . . . . .	25
<b>TDoA</b>	Time Difference of Arrival . . . . .	2
<b>ToA</b>	Time of Arrival . . . . .	2
<b>ToF</b>	Time of Flight . . . . .	58
<b>VANET</b>	Vehicular Ad-hoc Network . . . . .	24
<b>WSN</b>	Wireless Sensor Network . . . . .	1







# Chapter 1

## Introduction

With advancing technology, Wireless Sensor Networks (WSNs) become more and more prevalent, having a multitude of applications, ranging from environmental monitoring [6], wildlife monitoring [7] over to industrial [8] applications. The growing prevalence is also linked to the growth of Internet of Things (IoT) deployment, as most smart devices will invariably also contain sensors of various sorts.

### 1.1 Localization in Wireless Sensor Networks

To make proper use of sensor data measured from WSNs, it is usually beneficial to know where exactly the data was gathered. For example, if sensors are monitoring the correct operation of machinery in the machine park of an industrial installation, when a failure is detected, this information obviously only becomes useful together with the information of which machine is failing, otherwise, it would be necessary to search the whole installation.

In the field of wildlife monitoring, tracking the positions of animals is often the very point of attaching sensors to them to track their migration or movement through a certain area. For example, Sommer et al. [7] have attached sensors to large bats to trace their movements.

Since location information is of such high importance, many different approaches have been developed to localize sensor nodes in various scenarios, such as static or mobile networks. Commonly, in these approaches, a difference is made between anchor nodes, which know their own locations and broadcast this information, and unknown nodes, which determine their own locations from the information provided by anchor nodes.

## 1.2 Motivation

Given the importance of location information to actually make sense of sensed data, it is also highly important to ensure that this location data is correct, even if some nodes in the network are malfunctioning or actively behaving maliciously. Most localization approaches focus on just optimizing localization accuracy with no regard to failing or misbehaving nodes within the network.

Overall, the following points have to be considered when designing a robust localization approach for WSNs and the IoT:

1. Applicability of the approach to a given scenario
2. Hardware and power requirements
3. Robustness regarding network integrity
4. Robustness regarding malfunctioning or malicious anchor nodes

In the following, the challenges in each area will be described.

### 1.2.1 Applicability

Certain localization approaches are only applicable in certain scenarios, while other approaches are more general. While more specific approaches can exploit the specific properties of the scenario they were designed for to increase localization accuracy, this also restricts their use to their specific domains and may open them up to certain attacks.

For example, an indoor localization algorithm that relies on anchor nodes positioned inside a building at certain, known positions, will take some effort to adapt to new environments. Additionally, moving one or multiple of the anchor nodes may be an easy way for an attacker with physical access to the building to cause incorrect localization results.

Another aspect to be considered is the necessary localization accuracy in a given scenario. In the indoor case, localization accuracy on the room level can already be sufficient for many purposes, such as assisting users in navigating a building. On the other hand, an input device used for virtual reality gaming or simulation might require localization results that are accurate to the millimeter in order to correctly track the user's hands.

In many WSN scenarios, coarse grained localization accuracy is sufficient, allowing for the use of range-free approaches, which are based only on connectivity and do not require any type of range measurements, such as those based on Received Signal Strength (RSS) [9], Time of Arrival (ToA) [10], Time Difference of Arrival (TDoA) [11] and so on. Such range-free approaches are often very efficient and have low requirements for deployment.

Another aspect to consider is whether node and anchor node locations are static or mobile. In fully mobile networks, many assumptions that could be otherwise made, such as the existence of static anchor nodes with known locations, may no longer apply.

Finally, it is important to determine whether a given scenario allows the use of centralized infrastructure or not. Often it is beneficial if nodes can localize themselves without relying on a central server, as in that case, only location information needs to be transmitted or stored, rather than the usually bigger amounts of data necessary to perform the localization process, such as neighboring node lists or similar network topology information.

### 1.2.2 Hardware and power requirements

WSN and IoT devices usually run on battery power. Depending on the use case, even the use of high capacity batteries may not be possible due to size or weight constraints, such as in the case of tracking bats by attaching sensor nodes to them [7].

Different types of localization approaches have different hardware and energy requirements. For example, GPS usually has relatively high power consumption [12]. Approaches relying on Angle of Arrival (AoA) [13] might require sectorized antennas or arrays of multiple antennas to determine directionality.

### 1.2.3 Robustness regarding network integrity

For the sake of this work, robustness regarding network integrity shall be defined as localization being robust with respect to attempts at undermining the integrity of the network. One common type of attack in this category is the Sybil attack [14]. In this attack, malicious nodes pretend to have multiple identities, which allows them to strongly influence various processes in the network. For example, a malicious anchor node that pretends to be multiple anchor nodes may have a stronger influence than regular anchor nodes. If an unknown node receives information from two regular anchor nodes and twenty Sybil identities of a malicious anchor node, its location estimate will then most likely be mainly influenced by the information provided by the malicious node.

Robustness against this type of attack can be achieved in multiple ways. One type of approach aims to detect Sybil nodes once they have already appeared in the network and attempts to mitigate the ongoing attack. For example, Sybil-free APIT (SF-APIT) [15] follows this approach.

Another approach is to prevent Sybil nodes from joining the network by ensuring that nodes cannot create new identities at will. In a network that is administered by a single entity, nodes can be equipped with certificates in a standard PKI approach (e.g. as per [16]). These certificates will not be forgeable and will prevent nodes from pretending to be other or entirely new nodes, because they do not have access to valid certificates for the identities of those nodes.

If decentralization is required, or there is no single entity, economic incentives can be used to

restrict the creation of identities. Proof of Work (PoW) is an approach where participants perform a computation with a given complexity and easy to verify solution. If identity creation is tied to the ability of performing such a computation, identity creation will incur a certain cost (time and power). However, regular users are quite limited in the amount of work they can reasonably be expected to perform when creating legitimate identities, which makes common PoW based schemes infeasible as found by Prünster et al. [17].

Once the cryptographic integrity of node identities can be ensured in some manner, the overall communications in the network can be secured using power-efficient, symmetric encryption and message authentication mechanisms [18], protecting transmitted data from manipulation and preventing unauthorized nodes from injecting invalid information into the network.

#### **1.2.4 Robustness regarding malfunctioning and malicious anchor nodes**

Localization performance can also be affected by anchor nodes if they provide incorrect location information to unknown nodes. This can either be the case due to a malfunctioning anchor node that, for example, is no longer able to correctly determine its own location or due to actively malicious behavior by the anchor node, trying to reduce unknown nodes' ability to localize themselves.

#### **1.2.5 Goal**

The goal of this work is to find an architecture for localization in WSNs and the IoT that keeps power requirements and hardware costs low, is easy to deploy and addresses both types of robustness concerns without incurring high computational effort or requiring centralization. The first requirements (power and hardware requirements, ease of deployment) indicate that a range-free localization approach will be most suitable, leaving mostly the questions of robustness and decentralization as open problems to explore.

### **1.3 Research questions**

Following the goals set in Section 1.2.5, the following research questions are defined:

- RQ1.** How can decentralized Sybil resistant identities be implemented for WSNs and IoT?
- RQ2.** How can localization accuracy be improved in range-free localization for mobile networks?
- RQ3.** How can the effects of malfunctioning and malicious nodes be mitigated during localization?

## 1.4 Thesis structure

The thesis is structured as follows. **RQ1** will be addressed in Chapter 2 and Chapter 3, through the Unchained and ReChained protocols for identity creation. In Unchained, rather than making users perform the work for PoW, it is instead outsourced to existing PoW Blockchain networks and replaced by a monetary transaction, while also not relying on any central authority. At the same time, offline verifiability of identity proofs is facilitated. ReChained focuses on formal verification of the protocol, introduces a revocation mechanism for identities and addresses eclipse attacks.

**RQ2** and **RQ3** are addressed in Chapter 4, which introduces Robustness Enhanced Sensor Assisted Monte Carlo Localization (RESA-MCL), a decentralized, range-free localization scheme that also mitigates the effects of incorrect location data being transmitted by malfunctioning or malicious anchor nodes.

Finally, Chapter 5 summarizes the contributions made throughout this thesis.

## 1.5 Copyright and contribution

Chapter 2 cites the paper titled "Unchained Identities: Putting a Price on Sybil Nodes in Mobile Ad hoc Networks" by Arne Bochém, Benjamin Leiding and Dieter Hogrefe of the Institute of Computer Science, Telematics Group, University of Goettingen, Germany (e-mail: {bochem,benjamin.leiding,hogrefe}@informatik.uni-goettingen.de) [19] verbatim, except for formatting and handling of acronyms. The copyright assignment to EAI signed by the authors states the following, permitting use in this thesis: "The right to reuse any portion of the Work, without fee, in future works of the author(s) or employer, including books, lectures and presentations in all media, provided that a citation of the EAI-published work, notice of the Copyright, and EAI DOI are included." The author of this thesis is the primary author of this work and has contributed to it significantly, initiating the research, designing the scheme and evaluation methodology and evaluation as well as writing most of the paper except for mainly parts of Section 2.1 and Section 2.2. The proof of concept implementation was made by Simon Schuler as part of his work as a student research assistant and was supervised by both Arne Bochém and Benjamin Leiding.

Chapter 3 cites the paper titled "ReChained: Sybil-Resistant Distributed Identities for the Internet of Things and Mobile Ad hoc Networks" by Arne Bochém and Benjamin Leiding of the Institute of Computer Science, Telematics Group, University of Goettingen, Germany (e-mail: {bochem,benjamin.leiding}@informatik.uni-goettingen.de) [20] verbatim, except for formatting and handling of acronyms and Section 3.8. It was published in the open-access MDPI Sensors journal under the Creative Commons CC BY 4.0 license, permitting use in this thesis, as long as attribution, such as this, is provided. The author of this thesis is one of the two primary authors sharing equal contribution to this work and has contributed to it significantly, designing and refining the ReChained scheme and revocation mechanism, analyzing the issue of eclipse attacks, and large

parts of the evaluation, while Benjamin Leiding has mainly contributed to the formal modeling and verification of the scheme, to the integration of Decentralized Identifiers (DIDs) and to Section 3.2.

Chapter 4 cites the paper titled "Robustness Enhanced Sensor Assisted Monte Carlo Localization for Wireless Sensor Networks and the Internet of Things" by Arne Bochem and Hang Zhang of the Institute of Computer Science, Telematics Group, University of Goettingen, Germany (e-mail: {bochem,hang.zhang}@informatik.uni-goettingen.de) [21] verbatim, except for formatting and handling of acronyms. This paper has been published at the IEEE Access open-access journal under the Creative Commons CC BY 4.0 license, allowing reproduction as long as attribution, such as this, is provided. The author of this thesis is the primary author of this work and has contributed to it significantly, initiating the research, designing the localization approach, performing the experiments and evaluation as well as writing most of the paper except for parts of Section 4.1 and Section 4.2.



## Chapter 2

# Unchained Identities: Putting a Price on Sybil Nodes in Mobile Ad hoc Networks

### Abstract

*As mobile MANETs and similar decentralized, self-organizing networks grow in number and popularity, they become worthwhile targets for attackers. Sybil attacks are a widespread issue for such networks and can be leveraged to increase the impact of other attacks, allowing attackers to threaten the integrity of the whole network. Authentication or identity management systems that prevent users from setting up arbitrary numbers of nodes are often missing in MANETs. As a result, attackers are able to introduce nodes with a multitude of identities into the network, thereby controlling a substantial fraction of the system and undermining its functionality and security. Additionally, MANETs are often partitioned and lack Internet access. As a result, implementing conventional measures based on central authorities is difficult. This paper fills the gap by introducing a decentralized blockchain-based identity system called Unchained. Unchained binds identities of nodes to addresses on a blockchain and economically disincentivizes the production of spurious identities by raising the costs of placing large numbers of Sybil identities in a network. Care is taken to ensure that circumventing Unchained results in costs similar or higher than following the protocol. We describe an offline verification scheme, detail the functionalities of the concept, discuss upper- and lower-bounds of security guarantees and evaluate Unchained based on case-studies.*

### 2.1 Introduction

Stimulated by the persistent growth and expansion of the IoT [22, 23], as well as progressing digitalization of our daily life, e.g., [24] and [25], wireless ad hoc networks such as mobile ad hoc networks (MANETs) or vehicular ad hoc networks (VANETs) become more common and popular. MANETs and their sub-types are often heavily partitioned, with transient connections occurring between nodes due to their mobility, resulting in a constantly changing network topology.

Furthermore, communication in MANETs is usually organized in a decentralized manner without a connection to any central authority or the Internet [26, 27]. As a result, these networks are worthwhile and easy targets for attackers. This raises the issue of providing proper security and privacy protection mechanisms in order defend them against attacks. Without such, the distributed nature of MANETs and their lack of a central authentication authority leaves them easy targets for Sybil attacks. This type of attack is a common issue in large-scale Peer to Peer (P2P) systems, where hostile or faulty computing elements threaten the security of the whole network. Single faulty entities may be able to present multiple identities, thereby controlling a substantial fraction of the system, consequently undermining its functionality and security [14].

Several techniques focus on preventing Sybil nodes from joining a network at all [28, 29]. Other approaches attempt to detect them when they are already part of the network [30, 31]. One of the key enablers of Sybil attacks is the absence of a mechanism that prevents attackers from setting up arbitrary numbers of (virtual) nodes. In MANETs, there usually is no central authority that controls or administers the network. Since detecting Sybil nodes after joining a network is a cumbersome and inaccurate task, we propose the *Unchained* protocol which introduces economic disincentives of introducing Sybil nodes to a network by leveraging blockchain technology and combining it with an offline verification approach.

*Unchained* uses blockchain technology to bind ad hoc network node identities to blockchain-based wallet addresses, i.e. public/private key pairs, and requires a certain deposit to be made on the blockchain in order to join the network. Circumventing the protocol and introducing a Sybil node means investing even more financial assets than it would cost to create an *Unchained* identity the regular way. Due to its offline verification approach, *Unchained* operates in environments without internet access and without direct access to the underlying blockchain, thereby “unchaining” its security mechanism. This allows its use in MANETs with no or merely intermittent Internet connectivity.

The remainder of this paper is structured as follows: Section 2.2 introduces related works and supplementary literature. Section 2.3 focuses on the operational details of the *Unchained* approach. Afterwards, Section 2.4 details security properties of the protocol and explains how to customize the protocol for various use cases, while Section 2.5 elaborates on different options to handle difficulty changes in the underlying cryptocurrency. Section 2.6 provides a discussion and evaluation based on case studies. Finally, Section 2.7 concludes this work and provides an outlook on future work.

## 2.2 Supplementary Literature and Related Works

This section provides background information and describes related works regarding previous approaches to solve the issue of Sybil attacks. Section 2.2.1 provides general information on the concepts of blockchain technology, terms and frameworks. Section 2.2.2 focuses on related works.

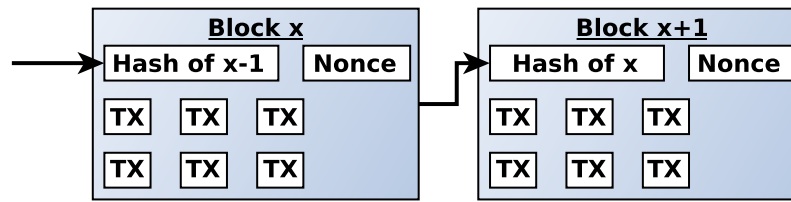


Figure 2.1: Blockchain structure, adapted from [1].

### 2.2.1 Blockchain Technology

A blockchain consists of a (theoretically) unlimited number of blocks which are chained together in a chronological order. Each block consists of transactions that successfully passed a validation procedure. As illustrated in Figure 2.1, the collected valid transactions result in a new block that is added to the existing blockchain. The blockchain concept, also called distributed ledger system, is most notably known for providing the foundation of the P2P cryptocurrency and payment system Bitcoin (฿) [1].

A key enabler of blockchains is the so called mining process, allowing to achieve a global consensus on which transactions to include in the next block in a decentralized way. Currently, the most common blockchain consensus algorithm is based on PoW, which is used by Bitcoin, Ethereum [32], and others. When collecting transactions to form a new block, participants have to solve a computationally hard puzzle that is referred to as PoW. A PoW is a piece of data that is difficult to produce but easy to verify and satisfies certain requirements. Bitcoin's PoW is based on searching for a nonce (value) that when hashed together with a block header, begins with a number of zero bits. "The average work required is exponential in the number of zero bits required and can be verified by executing a single hash" [1]. The varying number of zero bits is used to adjust the difficulty of finding a valid block. Hardware speed ups and growing user participation in building blocks result in more computing power being available for the mining process. In order to publish new blocks in, on average, a given time intervals, the difficulty of the PoW is adjusted depending on the available computing power. In the case of Bitcoin, the target time per block is ten minutes. In the case that new blocks are generated too fast, the difficulty increases, if new blocks are generated too slowly, it is decreases.

As soon as a block with a valid nonce is found, the block is published and attached to the chain. All participants verify the submitted PoW for correctness, the included transactions for validity and accept it as the new latest block. Since each block depends on its predecessor, changing the content of a block requires an infeasible recalculation of all successor blocks. The first user to find a new block also receives a block reward. In the case of Bitcoin, this reward is, as of December 2017, 12.5 ฿ plus additional transaction fees. These block rewards have both the purpose of disseminating the currency among users, as well as incentivizing miners to spend energy on securing the blockchain. If multiple blocks are found at the same chain height, mining may proceed on either block and the

longer chain is considered valid.

### 2.2.2 Related Works

Several other projects focus on Sybil attack prevention and Sybil attack detection in different network environments, therefore we only highlight some further publications. SybilGuard [33] is one of the well-known protocols that aims to limit the corruptive influence of Sybil attacks in peer-to-peer networks. The SybilLimit protocol is an advanced version of SybilGuard and aims to defend online social networks from Sybil nodes [34]. SybilGuard as well as SybilLimit rely on human-established trust relationships, hence they cannot be applied to mobile ad hoc networks.

[30] and [31] focus on Sybil attack detection in MANETs, whereas [35] targets the detection and localization of Sybil nodes in VANETs. In contrast to these approaches, Unchained focuses on preventing sybil nodes from joining a network instead of detecting them when they are already part of the network.

Furthermore, [28] and [36] try to prevent and detect Sybil attacks in sensor networks, whereas Unchained focuses on mobile ad hoc networks in general.

Blockchains matured and grew in popularity, resulting in various blockchain architectures, e.g., Ethereum [32], Qtum [37], or IOTA [38], as well blockchain-based applications and use cases, e.g., as a platform for IoT applications [39,40], applications in the automotive sector [41], in the finance sector [42,43] or as a part of security and authentication protocols [40,44,45].

## 2.3 Unchained Identities in MANETs

When setting up a new network, e.g., a MANET, each node is equipped with an identity that uniquely identifies the specific device within the network. When deployed, communicating devices have to validate each others identities for security and privacy reasons before exchanging information. The following section describes the general process of creating new identities when flashing the firmware to a device as well as validating identities. Both of these processes are based on blockchain technology and do not require any trusted third parties apart from a decentralized cryptocurrency's P2P network. For illustration purposes, we use the Bitcoin network in the following sections. However, *Unchained* can be implemented on all PoW based Blockchains.

### 2.3.1 Creating a New Identity

The process of creating a new unchained identity is illustrated in Figure 2.2 and assumes that a key pair, i.e. public and private key, presenting a Bitcoin wallet address already exists and that it holds a certain amount of Bitcoin. The amount contained within this address needs to be sufficient to make the deposit necessary in the first step of the identity creation process. First, the coins in the Bitcoin wallet are transferred to a pre-defined deposit address (step 2) and the resulting transaction

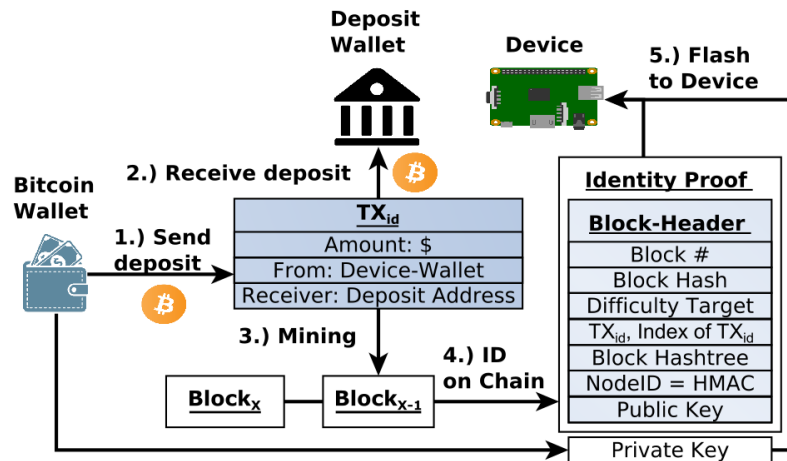


Figure 2.2: Creating a new identity.

is mined into a block by the Bitcoin network as part of  $Block_x$  (step 3). In the subsequent step 4, an *Identity proof* is created based on the information from the mined block. The proof contains the block header (block number, block hash, difficulty target of the block), the deposit transaction, hashes for the merkle tree allowing to prove that the transaction is part of the block, the index number of the deposit address in the block as well as the public key.

Furthermore, a unique *NodeID* is calculated based on the Hash Message Authentication Code (HMAC) as illustrated in Equation 2.1 - 2.3. First, the block's PoW hash is used as a key for the HMAC calculation in combination with the index number of the deposit transaction in the block. The purpose of this approach is to prevent attackers from attempting to create node IDs matching arbitrary attacker defined criteria. Since the ID depends on the deposit transaction, but also on the block's PoW hash, guessing the node ID is equivalent to predicting the correct hash of the next block of the Bitcoin blockchain and therefore not feasible.

$$k_{HMAC} := \text{Block}_{PoWHash} \quad (2.1)$$

$$TX_{index} := \text{index of deposit TX in Block} \quad (2.2)$$

$$\text{nodeID} := \text{HMAC}(k_{HMAC}, TX_{index}) \quad (2.3)$$

Finally, the constructed *identity proof* and the node's private key are flashed onto the node, which is afterwards deployed in the network.

### 2.3.2 Identity Validation

Communication between nodes of a network is an essential functionality of ad hoc network. Before transmitting application data, nodes verify each others' identity in a bidirectional manner in order to secure and protect sensible network data.

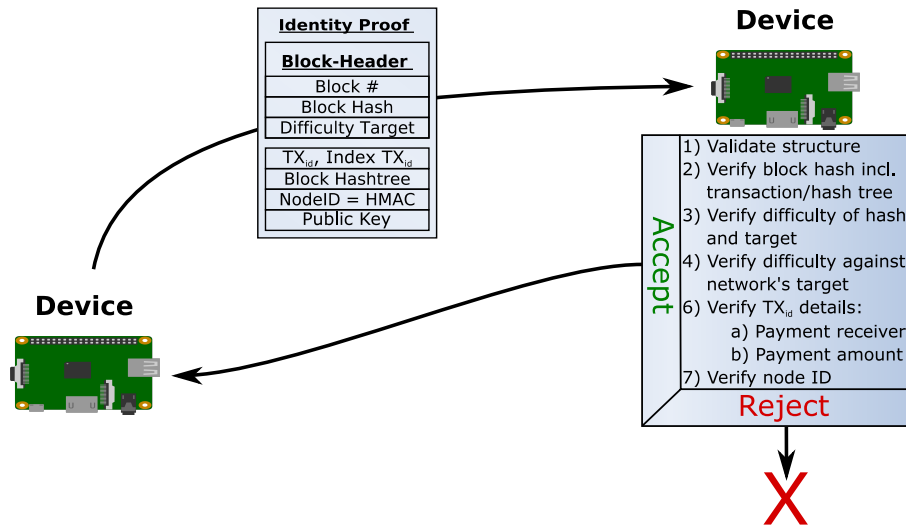


Figure 2.3: Overview of the validation process.

The validation of node identities is performed upon first contact of each two nodes, such as a two-way handshake or the broadcasting of identity information to let nodes learn about their neighbors. In the following, we describe how participating nodes verify an identity upon receipt of its *identity proof*. A graphical overview of this procedure is given in Figure 2.3. In the case of a two-way handshake, the procedure is simply repeated on each side after receiving the *identity proof*. Another potential scenario is a number of nodes broadcasting their *identity proofs* and verifying received *identity proofs*, allowing them to connect to surrounding peers when necessary.

First, the structure of the *identity proof* is validated. A valid *identity proof* contains a block header, a deposit transaction, the hashes of a merkle tree proving that the transaction is part of the specific block, a node ID and the public key of the node, which was also used to sign the deposit transaction. At this point, it is also verified that the block's height is above  $\text{networkParameter}_{\text{height}}$ . Afterwards, the block corresponding to the block header is verified based on the hashes from the merkle tree that is used to verify the transaction's presence in the block. Next, the node checks that the difficulty of the block hash matches the difficulty target in the block header and that the difficulty target is at least  $\text{networkParameter}_{\text{minDifficulty}}$  for the given block height.

Once these properties are confirmed, the deposit transaction is verified. It contains a payment greater or equal to  $\text{networkParameter}_{\text{amount}}$ , which is sent to the mandatory receiver address  $\text{networkParameter}_{\text{receiver}}$ . The transaction has to be correctly signed with exactly one key pair that is also used for any future cryptographically secure (encryption, signatures, key exchange) communications with other nodes and the public key that is included in the *identity proof*. Finally, it is verified that the node ID matches the formula given in Equation 2.3.

If all steps described above are successful, the validation process finishes and the participating

nodes may start communicating. Otherwise the received *identity proof* is discarded and no further communication is initiated.

## 2.4 Parameter Choice

In Section 2.4.1 we discuss various network parameters that allow adjusting the behavior of *Unchained* to suit different use cases. In addition, security properties and considerations are detailed. Section 2.4.2 focuses on pricing an identity.

### 2.4.1 Network Parameters

First, we describe the network parameters that allow the customization of *Unchained* to suit different use cases.

#### Starting Block Height

The parameter  $\text{networkParameter}_{\text{height}}$  defines the minimum block height that is accepted for *identity proofs*. The block height is defined by the number of blocks preceding a block on the blockchain. The genesis block's block height is zero [46]. The block height corresponds to the block height at the start of the network's lifetime. By rejecting *identity proofs* at lower block heights, there is no need to consider allowing blocks with significantly lower difficulties.

#### Deposit Address

$\text{networkParameter}_{\text{receiver}}$  is the deposit address to which a transaction, used when creating an *identity proof*, sends a certain amount of Bitcoin. The main property of this address is that funds sent there should not be recoverable by an attacker that is trying to create a large number of identities. Hence, using transaction fees instead of a deposit is not a viable option since an attacker may mine a valid block on their own and directly recover all funds. Currently, *Unchained* provides three different options that define what happens to the deposit.

The first option is proof-of-burn [47], where an invalid receiving address with no (known) existing private key is used. As a result, the sent deposit cannot be recovered. This method is secure, but not elegant since it destroys a certain amount of Bitcoin and the Bitcoin supply is strictly limited by the underlying Bitcoin protocol.

The second option is that the software of the network secured with *Unchained* is developed by a certain entity, or the network is maintained or controlled by a certain entity. The entity may choose to use an address under its control as the receiving address. This way, the developers or maintainers of the network could raise funds for further development by receiving Bitcoin through the creation of identities used by users of the network. As the developers have an interest in keeping the network secure, this approach is a viable choice that prevents attackers from recovering funds.

A third approach is to use the donation address of a charity. Unless the charity itself has an interest in attacking the network or is otherwise compromised, an attacker is unlikely to be able to recover the funds. If desired by a network operator, they may choose to allow multiple deposit addresses to be used. Hence, users may choose between different charities when making a donation to create an *Unchained* identity.

### Deposit Amount

The `networkParameteramount` parameter determines the minimum deposit size required to set up a new identity. The amount is chosen in such a way that it is affordable for those who would like to participate in the network, while still being high enough to disincentivize the creation of large numbers of Sybil nodes. For larger networks, the deposit size may be lower since the network may tolerate higher numbers of spurious identities before an attacker gains a tangible benefit from their use. Section 2.4.2 details further considerations, limits and implications that depend on the deposit size.

A potential alternative is to use a small value for `networkParameteramount` and introducing a bigger `networkParameterlockedAmount` value. The first amount gets sent to `networkParameterreceiver`, while the second amount is sent back to the identity's owner, but locked up using the `CheckLockTimeVerify` output [48] of a transaction or another type of smart contract. The locktime is equal to the lifetime of the identity. This way users may recover their funds after leaving the network, while ensuring that the creation of high numbers of concurrent identities still lock up significant amounts of capital.

### Minimum Difficulty

This parameter defines the minimum amount of work that is required to generate a new block that may be used to build an *identity proof*. While Section 2.5 details more sophisticated approaches to control the allowable difficulty of blocks for *identity proofs*, the most basic way is to set a simple minimum difficulty parameter `networkParameterminDifficulty` that matches the underlying blockchain's difficulty at the time of setting up the network using *Unchained*. Alternatively, a value slightly below this value may be chosen to allow for drops in network difficulty.

If the parameter is hardcoded, it should be selected sufficiently low. If Bitcoin's target difficulty drops below the hardcoded value, it becomes impossible to create further identities. To avoid this issue, implementations should set `networkParameterminDifficulty` dynamically as described in Section 2.5.

### Updates

Changes in the valuation of Bitcoin, difficulty or simply the general operating environment of the secured network may change over time. Therefore, it may become necessary to update the



parameters described above to ensure that the network is operating as desired.

Assuming an entity that is maintaining and developing the network, it is possible to periodically distributed signed bundles of updated network parameters, including the block height at which this bundle should take effect. After the bundle is published, users who generate fresh identities should attach the update to their *identity proof* before flashing it onto their node. Similar considerations are also described in Section 2.5 with a special focus on difficulty updates.

### 2.4.2 Pricing an Identity

An attacker trying to create a large number of identities aims to minimize costs. One option is to mine a block conforming to the  $\text{networkParameter}_{\text{minDifficulty}}$  parameter, filling it solely with deposit transactions, but never publishing it to the Bitcoin network. Since *Unchained* does not verify the full blockchain, these transactions do not even require valid inputs. However, mining a block with valid difficulty and not publishing it incurs a high opportunity cost, as well as energy cost. Hence, instead of paying for the identities, the attacker pays for the hashing power used to create the block. While the energy costs may vary, depending on location, the opportunity cost is easy to quantify and equal to the block reward plus additional transaction fees.

Given the current block size (1 MB), minimum transaction size (224 B) and block reward (12.5 ₿ plus fees) of Bitcoin, this also leads to an upper limit on the price for one identity, as given in Equation 2.4, with the current maximum amount given in Equation 2.5 [47, 49, 50].

$$\text{amount}_{\text{max}} = \text{block reward} \cdot \frac{\text{min TX size}}{\text{max block size}} \quad (2.4)$$

$$= 12.5 \text{ ₿} \cdot \frac{224 \text{ B}}{1 \text{ MB}} = 2.8 \text{ m₿} \quad (2.5)$$

Given the current price of Bitcoin as of 2017-10-29 at approximately \$10399 [2, 3], the resulting maximum price per identity is roughly equivalent to \$29. Going above this limit makes it cheaper for an attacker to generate a fake block than simply paying for the identities, as long as fees and energy costs are disregarded. Most networks will likely set a lower value than 2.8 m₿ for  $\text{networkParameter}_{\text{amount}}$ , in order to make identities more affordable for users and anticipate volatility with regards to Bitcoin valuation.

## 2.5 Handling Difficulty Adjustments

Our system has to adapt to changes in the target difficulty of the underlying cryptocurrency. In the case of Bitcoin, the difficulty is adjusted every 2016 blocks. This is equivalent to roughly two weeks. These adjustments are made to keep the time interval between each block at, in the case of

Bitcoin, on average 10 minutes. To handle these adjustments, we propose multiple approaches with different trade-offs.

Each node keeps a list of accepted target difficulties for each 2016 block interval. If the accepted target difficulty of an interval is adjusted upwards due to new information, identities confirmed using blocks with lower difficulty are retroactively invalidated. When the accepted target difficulty is lowered, it may be prudent to retroactively accept discarded peers into the network. However, since invalid identities are unlikely to be stored, the second case is unlikely to be implemented. The different approaches of handling difficulty changes concern the way this list of accepted target difficulties is updated.

### 2.5.1 Maximum Seen Difficulty

The first approach is both simple to implement as well as fully decentralized. The list of accepted target difficulties is initialized to zero or a known history at the point the node is initialized. Whenever an *identity proof* is received by a node, it looks up the target difficulty for that block in the list of difficulties. If both difficulty values match, the identity is accepted. In case the difficulty of the received *identity proof* is lower, the identity is discarded. Alternatively, when the difficulty of the received identity is higher, it is accepted and the target difficulty in the list is updated. If the list of accepted target difficulties was initialized with a known history however, these known-good values should not be overwritten even if an identity with a higher difficulty is encountered.

This solution allows the eventual detection and invalidation of forged identities that were validated using blocks of insufficiently high difficulty, as long as a connection to an honest node from the same two week period is made at some point. No infrastructure in addition to the previously described system is necessary.

As a caveat, this method is vulnerable to a denial of service attack. Assuming an attacker is able to mine a block targeting a difficulty that is higher than the difficulty of the underlying cryptocurrency and uses the block in an *identity proof*, the targeted nodes will update their lists of target difficulties accordingly, invalidating all regular identities that were generated during the timeframe corresponding to the malicious block. However, mining a block targeting a higher difficulty is even more expensive than mining a regular block. This issue can be mitigated by combining this approach with one of the two following methods. At the same time, this method can be used as a fallback solution for both of them.

### 2.5.2 Bundled Updates

Assuming network is run by a single operator, the operator may publish signed messages containing the target difficulty for each 2016 block range. The message is appended to each *identity proof*, setting the target difficulty in the list to the provided value. A drawback of this solution is that, in case the operator ceases to exist, no further difficulty updates can be broadcasted, leaving new

nodes unable to join the network. However, when combining this approach with the mechanism from Section 2.5.1, only nodes worried about denial of service attacks need to attach update messages to their *identity proofs*. Nevertheless, joining the network without one of these messages also remains as an alternative. Nonetheless, when the operator ceases operations, the mitigation for the denial of service attack vector also ceases to be functional.

### 2.5.3 Majority Vote

Rather than relying on a single operator as in Section 2.5.2, nodes may choose to accept signed difficulty updates from multiple providers. One or more of these messages may then be attached to an *identity proof*. The values of each update provider are stored in the list of target difficulties. In the event that for any interval mismatching difficulty update messages are detected, the majority value is considered the true difficulty target. Whenever there is no majority, the highest value is treated as the true difficulty target. In the case of a majority, nodes might mistrust future update messages provided by providers belonging to the minority.

This approach has multiple benefits over the previous approach. There is no single point of failure that prevents the network from growing. Additionally, supposing an attacker is able to trick a difficulty update provider to forge an update, the result is not necessarily a successful attack, as the attacker is still missing a majority that accepts the update. Hence, the attacker has to compromise at least 50% of the update providers to perform a denial of service attack.

Moreover, this approach is compatible with the solution from Section 2.5.1, allowing nodes to join the network even without access to any signed difficulty update messages.

## 2.6 Evaluation

The following section focuses on evaluating the *Unchained* protocol and the provided security guarantees. Since *Unchained's* security guarantees mainly depend on the difficulty level as well as the token price of the underlying PoW blockchain, we analyze how changing difficulty levels and token prices would have affected the Sybil attack prevention mechanism of a fictional MANET deployed in December 2016. Section 2.6.1 and Section 2.6.2 perform analysis based on the assumption that the Bitcoin blockchain is used, and Section 2.6.3 uses the same scenario based on the Ethereum blockchain. We choose these two chains for several reasons: First, they are the most popular and most utilized PoW blockchains that currently exist. Second, the different changes in difficulty and price cover important corner cases such as increasing and decreasing difficulties over time with sudden drops and raises. We assume a scenario of a MANET that was initially deployed in December 2016 with nodes continually joining and leaving the network and operated until the time of writing this work in November 2017.

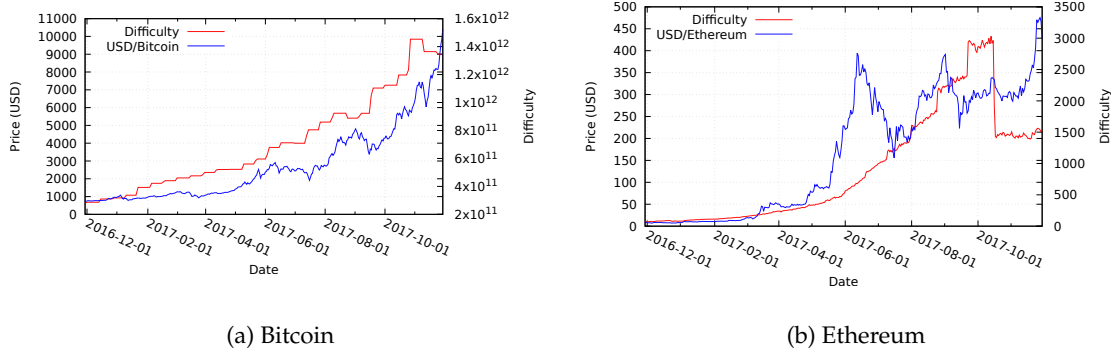


Figure 2.4: Average daily price in USD and block difficulty level between Dec. 2016 and Nov. 2017 (Source: [2–4]).

### 2.6.1 Bitcoin Difficulty Analysis

The difficulty level of the Bitcoin blockchain is adjusted every 2016 blocks, which is equivalent to 14 days given a blocktime of ten minutes per block. As illustrated in Figure 2.4a, the difficulty level is steadily rising with two minor exception in August and November 2017. At the same time, as shown in Figure 2.4a, the Bitcoin price itself also increased almost steadily by a factor of ten within the last twelve months.

As already discussed in Section 2.4 the lower bound of security guarantees provided by *Unchained* is always the lowest level of difficulty and the lowest price per block that occurred during the existence of the network. Given the initial deployment of our hypothetical MANET in December 2016, all nodes joining at later stages have higher security guarantees than the initial nodes due to an increased block difficulty and price. As discussed in Section 2.4.2, the price is only a theoretical measurement for security guarantees, since it is up to the network operator to decide the minimum price of a node’s identity. However, increasing token prices and therefore also increasing block prices, may also result in more expensive identities thereby raising the bar for a Sybil attacks.

Assuming a MANET setup at the beginning of November 2017, days before the decreasing Bitcoin price in (see Figure 2.4a), reflects the exact opposite where it becomes less expensive to introduce new identities to the system for a short period of time. However, as discussed previously, it is up to the network operator to decided whether to pick the maximum possible identity-per-block-price or a lower price. For practical reasons it is likely that most operators pick a lower price and therefore minor price declines do not affect the security guarantees of our example MANET a lot. Moreover, it is also up to the network operator to define a minimum identity-price that is higher than the identity-per-block-price since it is still unlikely that a malicious entity has the computational power to mine a block with a matching difficulty level, given the vast hashing power of Bitcoin’s mining pools.

### 2.6.2 Bitcoin Price Analysis

Using the historic price data gathered for Figure 2.4a, going back until July 17th 2010, it is possible to calculate for each date, on which a network using *Unchained* could have been started, the highest drop in price and thus security level experienced by the network. While future developments cannot reliably predicted, this provides an intuition on the historic worst case performance of *Unchained*. In Table 2.1, the proportion of starting dates that would have lead to a drop on any subsequent day of at least a given percentage is given. Only for 0.1 % of possible starting dates the security level would have at any later point dropped below 10 % of the given date.

Drop to	Affected started dates
< 10 %	0.1 %
< 20 %	2.8 %
< 30 %	8.5 %
< 40 %	13.5 %
< 50 %	18.0 %
< 60 %	22.8 %
< 70 %	27.2 %
< 80 %	36.2 %
< 90 %	49.5 %
< 100 %	77.5 %

Table 2.1: Affected starting dates after which the Bitcoin price drops below a certain percentage of the given day's price.

Historically, high drops in security level only occur very rarely. Smaller drops occur more frequently, with almost 50 % of possible starting days experiencing drops of at least 10 % at some point in the future. While most networks will be able to tolerate smaller drops in security level, raising Bitcoin prices can also be an issue, as they can make identities too expensive for regular users. Considering this, for networks intended to exist over long time frames, provisions for an update mechanism for `networkParameteramount` should be made. In case mass adoption occurs, the volatility level of cryptocurrencies and fiat currency is expected to converge. Hence, *Unchained*'s level of security will stabilize as well.

### 2.6.3 Ethereum Difficulty Analysis

*Unchained* is blockchain agnostic as long as the underlying chain architecture uses a PoW consensus algorithm. Therefore, we also analyze how the changing difficulty levels and token prices of the Ethereum blockchain and how this would have affected the Sybil attack prevention mechanism of our fictional MANET deployed in December 2016.

Similar to the Bitcoin price, the Ethereum price also increased heavily, starting around \$8 in December 2016 to more than \$450 at the end of November 2017, even though the Ethereum price suffered some decreases in July 2017. Ethereum's block difficulty, illustrated in Figure 2.4b, also

increased over the last twelve month. However, a sudden drop occurred on October 16 due to a difficulty adjusting hard-fork of the Ethereum network [51]. Nevertheless, even the reduced difficulty level is far higher than the initial level in December 2016.

As a result, the security guarantee evaluation results are similar to the Bitcoin evaluation of Section 2.6.1. MANET nodes setup in December 2016 with the initial difficulty are cheaper and easier to create in terms of identity price and block difficulty. All nodes created afterwards provided higher security guarantees. When focusing on the timeframe briefly before and after the difficulty adjustment, identities created before the adjustment are less difficult than identities created afterwards. The same applies for the price of identities both before and after the price drop of Ether in July 2016 as illustrated in Figure 2.4b.

In both the case of the Bitcoin as well as the Ethereum blockchain, price and difficulty increased heavily within the last 12 month. As a result, the lower bound of provided security is defined by the earliest nodes that joined the test MANET when created, since their *identity proofs* depends on the lowest block difficulty and identity price. All following node identities provide security guarantees above this lower bound. Furthermore, given the case that the difficulty levels will likely not increase indefinitely and remain somehow static (with minor fluctuations) at some point in the future, *Unchained's* lower and upper bounds will also converge and be less volatile.

## 2.7 Conclusion and Future Work

Detecting Sybil node attacks is major issue of large-scale P2P networks where malicious nodes threaten the security of the overall system. After joining a network, detecting such nodes is a cumbersome and inaccurate task. In this work we introduce a protocol for a decentralized blockchain-based identity system with offline verification that raises the difficulty of introducing high numbers of Sybil nodes to a network by providing economic disincentives.

*Unchained* uses blockchain technology to bind ad hoc network node identities to blockchain-based wallet addresses, i.e., public/private key pairs. In order to join the network, a proof-of-identity is created for each device. The proof is derived from a deposit transaction made from the wallet address to a deposit address and flashed to the node afterwards. Nodes validate each others' identities using the uniquely generated *identity proofs*.

Circumventing the protocol and introducing a Sybil node is equivalent to investing more financial assets than it would cost to create a malicious block on the blockchain. In addition, *identity proofs* are designed in such a way that no Internet access or direct connection to the underlying blockchain is required after the initial setup of the ad hoc network, thereby raising the bar to introduce Sybil nodes to even highly partitioned networks.

We detail the network parameters and update mechanism of *Unchained* and discuss upper- and lower-bounds of security guarantees. Finally, an evaluation based on a hypothetical MANET

deployed leveraging the Bitcoin and Ethereum blockchain is used to analyze the protocol's security properties depending on the block difficulty and token prices between December 2016 and November 2017.

For future work, we plan to generalize the protocol and not only focus on MANETs or other ad hoc networks and instead integrate *Unchained* into IoT environments. Furthermore, we intend to explore the feasibility of adapting existing Sybil attack prevention or detection algorithms to consider the node's *identity proof* block difficulty and block price as attributes for their trust scoring systems.

We also aim to implement and deploy the *Unchained* protocol on the Bitcoin as well as Ethereum blockchain and evaluate real-world use-cases.





## Chapter 3

# Rechained: Sybil-Resistant Distributed Identities for the Internet of Things and Mobile Ad hoc Networks

### Abstract

*Today, more and more Internet of Things devices are deployed, and the field of applications for decentralized, self-organizing networks keeps growing. The growth also makes these systems more attractive to attackers. Sybil attacks are a common issue, especially in decentralized networks and networks deployed in scenarios with irregular or unreliable Internet connectivity. The lack of a central authority that can be contacted at any time allows attackers to introduce arbitrary amounts of nodes into the network and to manipulate its behavior according to the attacker's goals, by posing as a majority participant. Depending on the structure of the network, employing Sybil node detection schemes may be difficult, and low powered Internet of Things devices are usually unable to perform impactful amounts of work for proof-of-work based schemes. In this paper, we present Rechained, a scheme that monetarily disincentivizes the creation of Sybil identities for networks that can operate with intermittent or no Internet connectivity. We introduce a new revocation mechanism for identities, tie them into the concepts of self-sovereign identities and decentralized identifiers. Case-studies are used to discuss upper- and lower-bounds for the costs of Sybil identities and, therefore, the provided security level. Furthermore, we formalize the protocol using Colored Petri Nets to analyze its correctness and suitability. Proof-of-concept implementations are used to evaluate the performance of our scheme on low powered hardware as it might be found in Internet of Things applications.*

### 3.1 Introduction

The persistent growth and expansion of the IoT [22,23], the progressing digitization of our daily life [24,25] and the emergence of complex machine-to-machine, or machine-to-human transaction

and interaction scenarios [5] results in a growing popularity of wireless ad hoc networks such as MANETs or Vehicular Ad-hoc Networks (VANETs). While participants of the Internet of Things should be always connected to the Internet by default, MANETs and their sub-types are often heavily partitioned, with transient connections occurring between nodes due to their mobility, resulting in a constantly changing network topology. Moreover, communication in MANETs is usually organized in a decentralized manner without a connection to any central authority or the Internet [26,27]. However, even IoT scenarios have to account for Internet disconnects and short period of ad hoc organization due to missing coverage or temporary disconnects, e.g., [52,53].

The growing popularity raises the issue of providing proper security mechanisms. Without such, the distributed nature of ad hoc networks and their lack of a central authentication authority leaves them easy targets for Sybil attacks. In a Sybil attack, malicious nodes participate in a network not only with their own identity, but also present multiple other identities under which they act. For example, in voting or majority based systems, if left unchecked, this type of attack can allow an attacker to use a minority of nodes with many identities to overvote outvote the legitimate participants. Such attacks are very common in P2P networks and can threaten the overall security and integrity. Malicious or faulty agents that either by intent or accident act under multiple identities can end up subverting the system by assuming control of a substantial fraction of it [14].

Many previous works put their focus on the prevention of Sybil attacks by barring misbehaving nodes from entering the network [28,29]. Another common approach is to detect misbehaving nodes that act under multiple identities [30,54]. However, the main reason that such attacks are possible at all is the fact that there is no mechanism in place, which prevents the creation of (virtual) identities or nodes in the network. This is usually the case for ad hoc networks or other networks without access to a central authority that manages and restricts access to the network. Such a central authority would often require access to an internet connection which may not be available in the network, or it may be available only intermittently. We previously proposed the *Unchained* [19] as a different approach to this issue. *Unchained* economically disincentivizes the creation of new identities which could be used to deploy Sybil attacks. The approach is based on requiring a PoW, but avoiding the necessity for the user creating the identity to perform this work by themselves. Instead, it is in effect offloaded to the mining network of a public blockchain and a direct payment transaction on that network is used to generate an identity creation proof which may be verified offline.

Both the original *Unchained* protocol and *Rechained*, as we propose here, create identities from such transactions on a blockchain. Since these transactions are signed using the public/private key pair of the sender, the identity also becomes tied to this key pair. The actual transaction used to create an identity has to follow certain requirements, such as a minimum amount of currency being transferred to one or multiple specific receiver addresses. We propose a way of determining certain amount boundaries, which ensure that attempting to circumvent the protocol would require the expenditure of more funds than are required by following it. *Unchained* focuses on offline

verification of proofs, in effect “unchaining” its security mechanism and allowing its use in isolated networks with no internet connectivity.

This work builds on top of the initial Unchained publication [19] as well as a further extension called *UnchainedX* [5]. In the following, we extend the *Unchained* protocol and its extension to incorporate the concept of Self-Sovereign Identity (SSI) for network participants and add the missing functionality of revoking *Rechained* identities. Moreover, we formalize the protocol using Colored Petri Nets (CPNs) [55,56] in order to detect and eliminate possible design flaws, missing specification details as well as thus far undetected security issues [57]. In *Rechained* we also consider how scenarios with or with intermittent internet connectivity allow nodes to verify current blockchain parameters, which is also the reason for the updated name. Finally, we address the issue of eclipse attacks targeting the *Rechained* protocol.

The remainder of this paper is structured as follows: Section 3.2 introduces supplementary literature and related work. Section 3.3 focuses on the operational details and outlines the security properties of the *Rechained* protocol. Next, Section 3.4 elaborates on different options to handle difficulty changes in the underlying cryptocurrency. In Section 3.5, we utilize CPNs to create a formal model of our protocol. Afterwards, Section 3.6 presents an evaluation of *Rechained* based on case studies and the previously created CPN models. Finally, Section 3.7 concludes this work and provides an outlook on future work.

## 3.2 Supplementary Literature and Related Work

This section provides background information, supplementary literature, and also introduces related work regarding previous approaches to solve the issue of Sybil attacks. Section 3.2.1 briefly summarizes the concept of SSI and DID, while Section 3.2.2 provides general information on the concept of blockchain technology. Section 3.2.3 focuses on related work.

### 3.2.1 Self-Sovereign Identities and Decentralized Identifiers

DIDs are a specific instantiation of the SSI concept. It has been proposed and is also currently under development by the W3C [58]. DIDs provide a digital identity representation that is controlled by the owning entity while at the same time being “independent of any centralized registry, identity provider, or certificate authority” [58].

A DID (*did:rechained:123456789abcdefghi*) consists of three parts. First, the so-called URL scheme identifier (*did*), second the DID method identifier (e.g., *rechained*) and last the DID method-specific identifier (*123456789abcdefghi*). The scheme part simply explains that we are handling a DID. The DID method identifier defines “how a specific DID scheme can be implemented on a specific distributed ledger or network, including the precise methods by which DIDs are resolved and deactivated and DID documents are written and updated” [58] – in our case the *Rechained* protocol.

The last part of the example details the unique entity identifier.

A DID corresponds to an entity and resolves to a DID document, which is represented by JSON-LD documents and describes how to use the DID. The DID document consists of a reference that links it to the corresponding DID, public keys that can be used for verification purposes, authentication methods to authenticate a DID, or the owning entity and service endpoints [58]. Moreover, DID documents may contain an *authentication* property, a mechanism “by which a DID subject can cryptographically prove that they are associated with a DID” [58]. The *authentication* property provides a list of various verification methods, e.g., public keys. Proving control over a DID document is exerted by resolving the DID to a DID document according to its DID method specification. Proving control over the public key specified in a DID document is achieved via a signature-based challenge-response mechanism using the private key corresponding to the public key.

### 3.2.2 Blockchain Technology

Figure 3.1 illustrates the general structure of a blockchain as used by, e.g., the Bitcoin [1], or Ethereum platform [32]. As the name suggests, a blockchain consists of a sequentially ordered number of blocks that records transaction events (denoted as *TX*), e.g., transfer of a cryptocurrency from entity A to entity B. Each block contains the hash of the previous ancestor block, thereby chaining all blocks together. Changing a transaction in a block results in a hash mismatch of the succeeding block. As a result, tampering with one block requires the recalculation of all succeeding blocks.

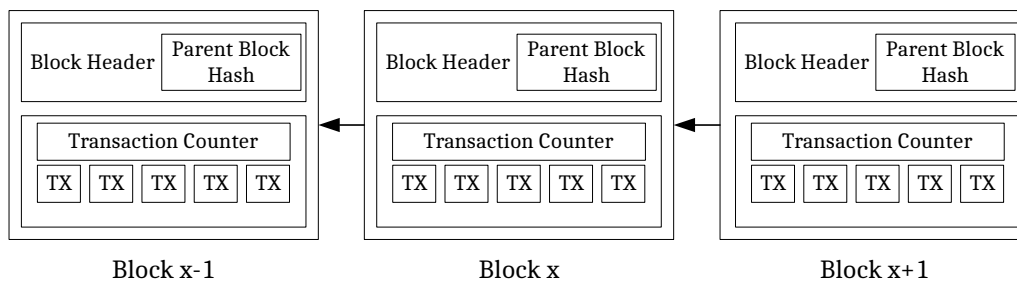


Figure 3.1: General blockchain structure – based on [1]

Different methods for achieving a global consensus on which transactions are included in a blockchain exist. The most common approach is called PoW and is in use by the most used blockchains Bitcoin and Ethereum [32]. PoW in the way it is used in Bitcoin was first introduced as Hashcash [59]. In this approach, the so called mining process works by having many different parties attempt to solve a computational problem of variable difficulty. Specifically, a search problem is used, where the hash of a nonce value concatenated with the next block’s block header is hashed and a hash with a value falling below the predefined difficulty target number is searched.

This means that the first miner to find a hash with beginning with a certain number of zero bits is allowed to publish the next block in the blockchain.

The important properties of this process are that the actual difficulty of the search problem can be adjusted over a wide range, while the result always remains easy to verify in constant time (a single hash operation). Increasing the difficulty target or number of leading zeros, exponentially increases the average amount of work necessary to solve the problem [1]. Usually, each blockchain has a set target block time. This is the average time it should take for a new block to be found. Using this as a target, the difficulty is adjusted to adapt the mining process to the amount of available computing power. If blocks are found too fast, the difficulty is automatically increased to compensate. If finding blocks takes too long, it is decreased. In the case of Bitcoin, these adjustments take place every 2016 blocks, which, with a target block time of 10 minutes, corresponds to around two weeks.

To incentivize the participation in the mining process, miners who find and publish a valid block may include a special transaction awarding themselves the so called block reward, which also serves to initially distribute the currency. For Bitcoin, as of June 2020, this block reward amounts to 6.25 ₿ plus any transaction fees paid by users of the currency to have their transactions included in the blocks. Once a new block is published, all participants in the blockchain's network will verify the validity of the block's hash, its correspondance to the difficulty target as well the validity of the included transactions. Each block contains the hash of the previous block, chaining them together into a blockchain and ensuring that past blocks cannot be tampered with. As the longest chain, meaning the chain with the highest accumulated PoW, is considered the valid chain, any attacker that would attempt to tamper with a past block would have to redo proofs-of-work for all subsequent blocks while being faster than the rest of the network so as not to fall behind.

### 3.2.3 Related Work

Sybil attack prevention and Sybil attack detection in different network environments is a topic of on-going research with various approaches. Thus, we only highlight a selection of related work. SybilGuard [33] is a well-known protocol that limits the harmful influence of Sybil attacks in P2P networks. The SybilLimit protocol is an advanced version of SybilGuard and aims to defend online social networks from Sybil nodes [34]. Both rely on human-established trust relationships; hence they cannot be applied to mobile ad hoc networks or the IoT. Moreover, the correctness of SybilGuard depends on the fast-mixing property of the underlying social network graph, which constrains its applicability in IoT scenarios.

Other works focus on specific types of ad hoc networks. While [54] focuses on signal strength-based Sybil attack detection in MANETs, [60] targets the detection of Sybil nodes in VANETs. The latter deploys monitoring nodes that collect information like distance, angle, or signal-strength with neighboring nodes and use fuzzy logic on the collected data to detect suspicious nodes. While the presented solutions are applicable in some MANET or IoT networks, they have quite specific

requirements, e.g., monitoring nodes, which limits their applicability and adds complexity to the network in terms of node configuration and computational tasks performed on the nodes. [61] and [62] try to prevent and detect Sybil attacks in sensor networks. The approaches are either limited to a particular application scenario (wildfires) or add the cost of additional complexity by deploying monitoring nodes. In contrast to the approaches presented above, *Rechained* focuses on preventing Sybil nodes from joining a network unless they sacrifice a pre-defined amount of resources instead of detecting them once they are already part of the network. *Rechained* does not require any monitoring nodes that collect data and perform extra computational tasks.

[63] rely on a PoW-dependent Sybil node prevention strategy that requires nodes to calculate proofs-of-work to join a network, a solution that is rarely applicable in energy- and computationally-constrained IoT scenarios. The authors of [17] briefly outline some limitations of PoW-based Sybil node attack prevention and question the applicability of PoW for node identity generation. They argue that by constraining the PoW difficulty so that users are willing to wait for it during identity creation, it will be limited to low costs that are not effective at disincentivizing Sybil attacks. *Rechained* however, offloads the actual PoW process onto an existing Blockchain, such as the Bitcoin blockchain, making this argument inapplicable. Instead, the PoW process is replaced by a direct payment transaction on the blockchain.

Distributed ledgers and blockchains matured and spread in popularity – most noticeably by providing the foundation of the cryptocurrency Bitcoin [1]. Inspired by the Bitcoin system, several further DLT platforms emerged, e.g., Ethereum<sup>1</sup>, Hyperledger<sup>2</sup>, or Tezos<sup>3</sup>. Moreover, a variety of applications involving blockchain technology have been proposed, e.g., for IoT applications and platforms [64,65], in the automotive sector [41], in the agriculture industry [66], or for asset tokenization [67].

### 3.3 Protocol Specification

Our primary consideration is the creation of identities that can be verified without, or with only irregularly available Internet connections. MANETs may be deployed in situations or environments without infrastructure, which facilitates Internet access for nodes. In IoT or vehicular applications, nodes may move through areas without connectivity or even remain there for a longer duration (e.g., underground parking without WiFi). In scenarios without, or intermittent Internet connectivity, identities are generated before the deployment of nodes and preloaded onto them.

Specific applications further require resistance against eclipse attacks. In an eclipse attack, an attacker creates Sybil nodes with identifiers selected to form a neighborhood around a target node

---

<sup>1</sup><https://ethereum.org/>

<sup>2</sup><https://www.hyperledger.org/>

<sup>3</sup><https://tezos.com/>

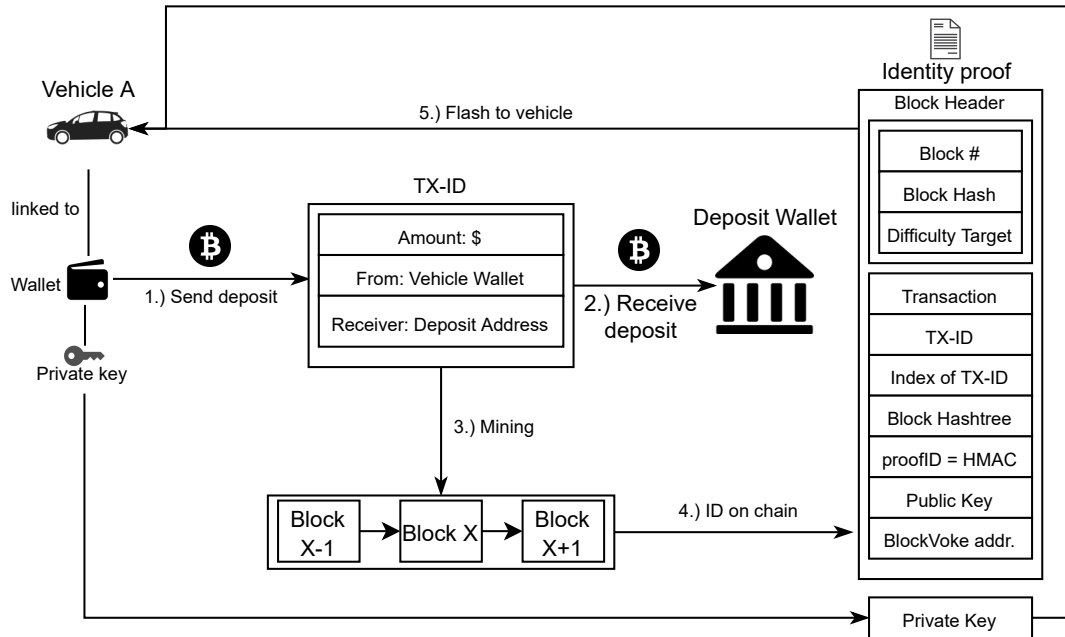


Figure 3.2: Creating a new identity proof – based on [19] and [5]

according to some distance metric. To prevent this, the identifier of an identity should be selected randomly in such a way that the owner of the identity has no direct influence on the selection process. If generating new identities is sufficiently expensive, and there exists a sufficient number of nodes, it then becomes infeasible to perform such an attack. We design our scheme with this in mind.

In the following, we present the design of our protocol, i.e., how to create and verify identities, as well as how our scheme ties into the concept of DIDs.

### 3.3.1 Creating Identities

An overview of the binding process for an existing DID-based identity, e.g., *did:rechain:123456789abcdefghi*, is given in Figure 3.2. It assumes a pre-defined identity outside the *Rechain* context and a corresponding key pair which represents a wallet address. Alternatively, a non-DID based identity is instantiated without the DID linking by creating an identity using a public and private key pair that will be associated with the created identity and sends a transaction with a predefined amount to one or multiple predefined addresses. Once this transaction is mined, the transaction and certain meta data will be used to construct an identity proof, which will allow network participants to confirm that the identity of the given public key was created legitimately. The created identity proof can finally be deployed on a device to let it join the *Rechain* secured network.

We assume that a Bitcoin (or other cryptocurrency) address with public key `pub` and private key `priv` has been prepared for the process and has been funded with the necessary amount of tokens, e.g., Bitcoin. To explain the process, we first define a number of parameters.

`receiver` shall be the deposit address for our scheme. During the identity creation process, the deposit is sent to this address. Possible attackers must not be able to recover the deposit from this address. In Section 3.4, we will detail various further properties of this address.

`amount` shall be the price of creating an identity. This is the minimum amount of cryptocurrency sent to `receiver` to create a new identity. More information about the choice of this parameter can be found in Section 3.4.

To make it possible to revoke this identity at a later point in time, we follow the basic approach of BlockVoke [68], which will be explained in more detail in Section 3.3.3. For this purpose, the user generates another separate Bitcoin (or other) address and adds it to the identity proof as well.

To create a new identity, an amount of at least `amount` is sent to `receiver` from the address corresponding to `pub` (steps 1 and 2). Additionally, an `OP_RETURN` output is included in the transaction. `OP_RETURN` outputs are a way to include an arbitrary 40 byte payload in a transaction. In this case, it contains the address to be used for revocation purposes. The transaction is mined into block  $x$  of the blockchain (step 3). Subsequently, the block is used to create an *identity proof* (step 4) containing the block header (block number, block hash, difficulty target of the block, etc.), the deposit transaction, the Merkle tree proof hashes necessary to prove that the transaction is part of the block, the transaction index `txIndex` of the deposit transaction in the block, the public key `pub` and a unique proof ID `proofID` of at least 128 bit. The proof ID is determined as given in Equation 3.1 - 3.2.

$$\text{key}_{\text{HMAC}} := x_{\text{BlockHash}} \quad (3.1)$$

$$\text{proofID} := \text{HMAC}(\text{key}_{\text{HMAC}}, \text{txIndex}) \quad (3.2)$$

As the user creating the identity has no influence on the block hash or index of the transaction within the block, this way of calculating the proof ID makes it resistant to eclipse attacks. Even assuming that a user collaborates with a miner, influencing the resulting proof ID is costly, since it requires the miner to discard valid solutions for the PoW search puzzle in order to find a block hash and transaction index combination that satisfies any given requirements by the user.

The *identity proof* and private key `priv` are flashed or otherwise transferred to the device which uses the generated identity (step 5). The device can then be deployed or otherwise join the network.



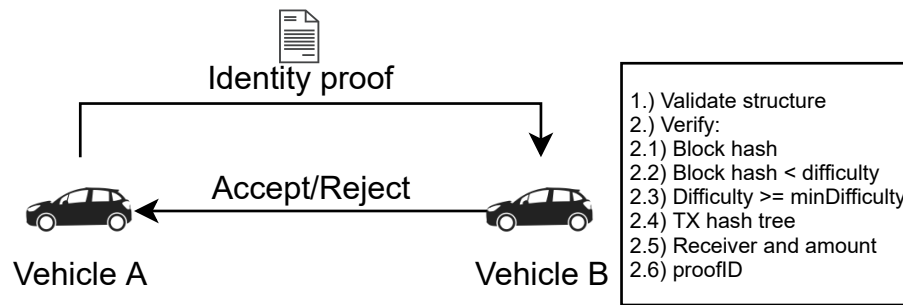


Figure 3.3: Overview of the validation process – based on [19] and [5]

### 3.3.2 Verifying Identities

Before nodes start communicating, they verify their peers' identities to prevent Sybil nodes from entering the network for free. An overview of this process is given in Figure 3.3. To start communicating, two nodes exchange *identity proofs* as part of a two-way handshake. Alternatively, a node may have received an *identity proof* for another node through some other method, such as learning about it from neighboring peers.

First, the structure of the *identity proof* is verified to ensure that it contains the necessary block header, transaction data and index, Merkle tree proof, the public key, and the proof ID. Moreover, it is checked whether the public key matches the private key used during the two-way handshake.

For the next verification step, two configurable parameters `minHeight` and `minDifficulty` are used, which will be further explained in Section 3.4. If the block number from the block header is higher than `minHeight`, the block hash corresponds to a difficulty of at least `minDifficulty` as well as the block's own difficulty target and matches the contents of the block header, this part of the verification process succeeds.

Next, the Merkle tree proof is used to verify that the given transaction is indeed part of the given block. It is also checked that the transaction sends an amount of at least `amount` to `receiver` and that its signature is correct and can be verified with the public key `pub`.

Finally, the proof ID is verified according to Equation 3.1 - 3.2.

If all verification steps succeed, the *identity proof* is accepted, and communications may take place. Otherwise, it is discarded, and no further communication takes place between the nodes.

### 3.3.3 Revoking Identities

Depending on the intended lifetime of the network, it may become necessary to revoke identities. The most basic mechanism is to give each identity a limited lifetime. However, the drawback is that identities need to be recreated periodically, incurring unnecessary costs for users who have no reason to revoke their identities.

Therefore, we propose the following mechanism to revoke identities, basically following the approach of [68]. As specified in Section 3.3.1, each identity proof contains an additional Bitcoin address. To revoke this identity proof, enough funds need to be sent to the revocation address to cover transaction fees for an outgoing transaction. Thereafter, a Bitcoin transaction will be made from the revocation address containing only an `OP_RETURN` output containing the six bytes "RECHND" and the 32 byte transaction ID of the transaction originally used to create the identity proof. This transaction will then act as a proof of revocation that any node in the network can verify, if it knows the corresponding identity proof.

When creating their identity proof, users may scan the blockchain for revocations by scanning for transactions containing only an `OP_RETURN` output containing the six bytes "RECHND" and then verifying that the transaction specified by the transaction ID following these bytes could actually generate a valid identity proof following the usual verification rules. If this is the case, the proof ID may be added to a list of revoked identities and flashed on the node together with the identity proof. A user may further include the most recent revocation transactions on the node and disseminate them through the network to help in preserving its integrity.

### 3.3.4 Distributed Identities

Previous Section 3.2.1 described the concept of DIDs as a specific instantiation of SSI. *Rechained* may be integrated with DID-based SSIs in two ways. First, a *Rechained* identity proof is linked as part of the *Authentication* field of the DID document corresponding to the machine identity. The *Authentication* field contains verification methods authorized by the DID subject for authentication purposes. This way, the *Rechained* identity proof is used for authentication purposes.

Alternatively, we propose a standalone instantiation of *Rechained* by implementing a specific *Rechained* DID scheme, e.g., *did:rechained:proofID*. The generated proofID is used as the DID method-specific identifier and "did:rechained" indicates the implementation of a DID scheme implementation specifically for *Rechained*.

## 3.4 Parameter Choices and Updates

*Rechained* depends on a number of parameters that can be set up in different ways to make it suitable for different types and sizes of networks. Depending on the estimated value of an attack, the cost of identities can be adjusted to either make it easier for users to join the network or to make it costlier for attackers gain enough identities to perform attacks. An overview of network parameters is given in Table 3.1.

### 3.4.1 Network Parameters

In the following, we explain the configurable network parameters of *Rechained*.

Table 3.1: Overview of *Rechained* parameters

Parameter	Description
<b>minHeight</b>	Identity proofs have to refer to blocks of at least this height on the chain.
<b>receiver</b>	One or multiple addresses that receive the payment for transaction creation.
<b>amount</b>	The minimum amount of crypto currency that has to be sent to create an identity.
<b>amountLocked</b>	Optional: A timelocked output sent back to the user, with the lock time determining the lifetime of the identity.
<b>minDifficulty</b>	The minimum mining difficulty required for an identity proof to be valid.

### Starting Block Height

The parameter `minHeight` is used to set a certain block height as the minimum height acceptable to ensure that no identity proofs created prior to the creation of the network can be used. This also means that no significantly lower difficulty values need to be considered as difficulty tends to go up over time, as is shown in Section 3.6.

### Deposit Address

The parameter `receiver` is the address that funds have to be sent to to create a new identity. The main property of this address is that an attacker trying to create a large number of identities is unable to recover the funds from it in any way. A receiver address has to be used rather than using the funds as mining fees, since an attacker may be able to collude with a miner or successfully mine a block themselves, allowing them to recover mining fees. In the following, we present three options for a `receiver` address.

The most secure way is proof-of-burn [47]. To burn funds, they are sent to an address with no knowable existing private key. However, as this method destroys the funds and the supply of Bitcoin is limited, it is considered to not be an elegant solution.

If the network secured with *Rechained* is developed or maintained by a certain entity, this entity may provide a `receiver` address under their control, i.e., the network operators sell identities. The funds thus gained are used for further development and maintenance or simply be considered as profit. Unless there is a conceivable reason for the network operators to attack their own network, this should also prevent possible attackers from recovering funds after the creation of identities.

Finally, `receiver` could be the donation address of a charity. In fact, it would be possible to slightly extend the scheme in such a way, that `receiver` contains multiple donation addresses and funds must either be distributed equally among them or sent to one of them according to the user's choice. Unless the chosen charities all have an interest in attacking the network or are compromised in some way, an attacker is highly unlikely to be able to (fully) recover the funds.

### Deposit Amount

The parameter `amount` sets the minimum amount that has to be sent to the `receiver` address. The value is chosen in such a way, that it is high enough to disincentivize the creation of spurious or malicious identities, while still being affordable for regular users who wish to participate in a given network. The choice of `amount` may also be influenced by the expected size of the network, as smaller networks may be more vulnerable against attacks and may therefore require a higher `amount` setting to defend against them. At the same time, the expected value that can be gained from attacking a network with Sybil identities should also be considered when setting the `amount` parameter.

Alternatively, *Rechained* can be implemented in such a way that a smaller `amount` is sent to `receiver`, while a second bigger `amountLocked` is sent back to the user creating an identity. However, the output `amountLocked` is time locked for the expected life time of the identity using a `CheckLockTimeVerify` [48] output. As a result, users recover most of their funds once they stop participating in the network, while still requiring attackers to acquire a significant amount of funds up front.

Further considerations on the choice of `amount` are given in Section 3.4.2.

### Minimum Difficulty

The parameter `minDifficulty` represents a set of different parameters, depending on the current block height. It is initially set at the time the network is created to the current difficulty, or slightly lower to compensate for fluctuating difficulty values. Unless overwritten by some sort of update mechanism as discussed in Section 3.4.3, this remains the minimum accepted difficulty for identity proofs. If an update mechanism is implemented, each period of time between difficulty adjustments (2016 blocks in Bitcoin, or roughly two weeks) will receive its own `minDifficulty` value.

## 3.4.2 Price Considerations

We attempt to disincentivize the creation of Sybil identities by introducing cost to the creation of identities. Any attacker would attempt to minimize their cost of attack. In networks without Internet connectivity and rare network parameter updates, an attacker may attempt to do so by making use of the actual Bitcoin network's target difficulty and the presumably lower target difficulty still used by *Rechained*. Such an attack is performed by cooperating with a miner to create a block with a target difficulty `minDifficulty`, filled completely with identity creating transactions. As this block targets a lower difficulty than the target difficulty of the Bitcoin network, it is cheaper to mine, but it cannot be published to earn block rewards. This introduces a significant opportunity cost for any miner cooperating with an attacker, unless the difference in target difficulties is very large.

An attacker has to compensate the miner for at least the opportunity cost value and most likely

also has to pay an additional fee. The opportunity cost of creating a block and not publishing it is easy to quantify as it is equal to the block reward (6.25 ₿) and any additional fees paid by transactions. Given the block size limit of 1 MB and a minimum transaction size of 224 B, this leads us to an upper limit for identity prices as calculated in Equation 3.3-3.4. Above these identity prices, the opportunity cost of creating a block optimized for identity creation may fall below the amount spent to create these identities properly.

$$\text{amount}_{\max} = \text{block reward} \times \frac{\text{min TX size}}{\text{max block size}} \quad (3.3)$$

$$= 6.25 \text{ BTC} \times \frac{224\text{B}}{1\text{MB}} = 0.0014 \text{ BTC} \quad (3.4)$$

As of 2021-04-22, the price of Bitcoin is at approximately 52009 USD [2,3], leading to a maximum identity price of approximately 72.81 USD.

In networks with at least intermittent Internet connectivity, an attack as described above is unlikely to occur as nodes can easily check if blocks are actually part of the blockchain once they get online. To do so, the nodes do neither have to download the full blockchain, nor have light-client capabilities. Instead, they act like super-light clients that just query a trustworthy source of their choice for the relevant block hashes. The calculations regarding the amount may still be a valuable guideline for pricing identities.

#### **Blockchain-based 51% attacks**

At this point, the possible impact of blockchain-based 51% attacks, where one party controls more PoW hashing power than the rest of the network, should also be considered. Generally, this kind of attack can be considered a catastrophic event for the underlying network and undermine its security and public trust. In the case of *Rechained*, performing a 51% attack would be a highly costly path of attack, compared to the aforementioned methods. However, if such an attack is already taking place, an attacker might be able to use such an ongoing attack to create identities more cheaply. It is expected that the price of a cryptocurrency undergoing a 51% attack would decline sharply, in turn reducing the costs of creating identities. It might also become possible for an attacker to bribe the party performing the 51% attack to let the attacker double-spend the funds used for identity creation.

#### **3.4.3 Updating Parameters**

For scenarios where network participants have Internet connectivity at least sometimes, nodes may check the difficulty at certain block heights and set their corresponding `minDifficulty` accordingly. The `amount` parameter is updated either through signed updates published by the network operator or one of the mechanisms discussed in the following, except that there is no need to distribute these updates in a P2P-manner.

In scenarios without Internet connectivity, the following mechanisms are proposed to keep the network parameters up-to-date and minimize the impact of difficulty and price changes. They may also be used in scenarios with intermittent Internet connectivity to bridge offline periods.

### Maximum Seen Difficulty

This is a fully decentralized approach that is also easy to implement. Once a node first receives an identity proof created within a given, so far unknown, difficulty adjustment period of two weeks, it sets the target difficulty seen in that identity proof's block as the `minDifficulty` for this period if it is above the network's initial `minDifficulty` value. If an identity proof is received for a period that has been initialized in this way, and the new proof's block contains a target difficulty that is higher, any previous identity proofs for this period with a lower target difficulty are invalidated and the `minDifficulty` for the period is updated to the new value.

This method allows the eventual detection of identities created using forged lower difficulty blocks once a connection to an honest node whose identity was issued in the same two weeks period is made. It does not require any additional infrastructure to operate.

However, the approach is vulnerable to a denial-of-service attack. If an attacker creates a block for a two weeks period that has a higher difficulty than the target difficulty of the actual Bitcoin network, all honestly created proofs will be discarded. As creating such a block is even more difficult than creating a regular Bitcoin block, and it also cannot be transmitted to the Bitcoin network due to a mismatch in the target difficulty block header field, this attack incurs significant costs for the attacker.

The method can be used as a fallback method for the following approaches.

### Bundled Updates

If the network is run by a single operator, this network operator may publish signed network parameter messages for every difficulty adjustment period. When creating their identity proofs, users also download the corresponding update message. When joining the network, the user's node distributes the update message together with the identity proof. Nodes receiving the update message verify the signature and set their `minDifficulty` and `amount` for the given period accordingly. This can be combined with the previous approach to allow nodes to join without having to retrieve an update bundle. A `minDifficulty` set through an update message would never be overridden by a higher difficulty, thus preventing denial-of-service attacks. At the same time, combining both methods still allows users to keep joining the network through the maximum seen difficulty approach if the network operator ceases publishing update messages.

### Majority Vote

A more decentralized alternative to the previous approach is for nodes to accept signed difficulty update messages from multiple providers. Any number of these messages may be provided together with an identity proof. The difficulty values are stored in a list for each difficulty adjustment period. If different values are provided by different providers, the majority vote is treated as the correct difficulty value. If there is no majority, the highest value is treated as the correct difficulty value.

The approach removes the single-point-of-failure that is present in the previous proposals above. If an attacker wants to influence the target difficulty for a given period, the attacker has to compromise a majority of update providers to get their difficulty target chosen. Like bundled updates, this approach can be used with the maximum seen difficult approach as a fallback option for nodes unable to provided bundled update messages or in case these messages cease being available at some point.

## 3.5 Protocol Formalization

Designing and specifying a new security protocol such as *Rechained* is a difficult task. Designing the protocol in such a way that prevents design flaws, security issues as well as incomplete specifications that pose risks to the protocols stakeholders and the user is yet another challenge [69–72]. Even in a best-case scenario, issues of a security protocol can pose dangers to the individual users who rely on it, while in other cases, design flaws and errors bring about serious real world consequences: The broken encryption of a wireless network [73] is an example for the first case, whereas a broken security protocol that grants an attacker access to sensible parts of nuclear power plants [74] illustrates a more serious threat.

Formal methods, such as Petri nets [75],  $\pi$ -calculus [76] and communicating sequential processes [77], address the posed challenge and are utilized for the design, development and analysis of new as well as existing protocols, thereby eliminating, or minimizing the security issues of the targeted protocols [78,79].

### 3.5.1 Colored Petri Nets

In the following sections, we formalize the *Rechained* protocol using CPNs [55,56] in order to detect and eliminate eventual design flaws, missing specification details as well as thus far undetected security issues [57]. CPN is a graphically oriented language used to design, specify, simulate as well as to verify systems. Moreover, it allows describing the states of a modeled system and the events that cause the system to change states.

CPN models are represented using a directed bipartite graph that consists of places, transitions, arcs and tokens. Places are denoted as circles and transitions as rectangles. Arcs connect places

with transitions, or transitions with places and have inscriptions given as Colored Petri Net Markup Language (CPN-ML) expressions [55,56,80–82]. CPN-ML is an expression programming language for inscriptions which are used to further specify data types and operations of the modeled system. CPN tokens and their colors represent the different data types of the modeled system. The resulting CPN model “of a system describes the states of the system and events (transitions) that can cause the system to change state. By making simulations of the CPN model, it is possible to investigate different scenarios and explore behaviors of the system” [55].

Besides its general suitability for system formalization, CPNs are especially well-suited for application in the context of blockchain-based systems. CPN models are discrete state machines and change states via transitions. Analogous to this, blockchains are discrete state machines as well, where the most recent block represents the current state of the system. With each new block, the system’s state transitions to a successor state. While in CPNs data structures are represented in the form of colored tokens, many blockchain platforms also use a data structure concept of tokens for the same reason. Moreover, blockchain transactions can be easily mapped to CPN token data structures. Furthermore, CPNs use CPN-ML expressions to specify and implement data types and operations of the modeled system, which correspond to the functionalities of smart contracts in the context of blockchain technology. Finally, the hierarchical structure of CPN models can be used to formalize blockchain-based dApps (decentralized applications) components of interleaved smart contracts. Thus, CPNs are well-suited as a formalism of choice for blockchain systems.

In the following, CPN-Tools<sup>4</sup> is used to design, evaluate and verify the CPN models. The result is a formal specification of the protocol that is used to guide further implementation efforts and design decisions.

### 3.5.2 Modeling Strategy

An appropriate modeling strategy is required to map the existing descriptions of the *Rechained* protocol as described in Section 3.3 and Section 3.4 to the corresponding elements of a CPN model. We map the informal descriptions and requirements of *Rechained* to a formal model using CPN, resulting in a sound formal model. To do so, we first outline the modeling strategy used to create the CPN models before presenting the resulting CPN models in the subsequent sections.

*Rechained* organizes and defines the exchange of information between different entities that are modeled as agents. In software engineering, various agent-oriented approaches exist, such as: Tropos [83], Gaia [84], Prometheus [85], MASB [86, 87] and MaSE [88]. In [89], Mahunnah et al. introduce a mapping heuristics from agent models to CPN models based on Sterling’s and Taveter’s [90] sociotechnical requirements-engineering methodology of AOM.

In system development and software engineering, good requirements follow certain characteristics. According to [91,92], requirements address one issue only and are completely specified without

---

<sup>4</sup><http://cpntools.org/>



missing information. Moreover, they have to be consistent and must not contradict themselves, or pose contradictions in correlation with other requirements. Finally, a requirement must also be atomic and without conjunctions [93].

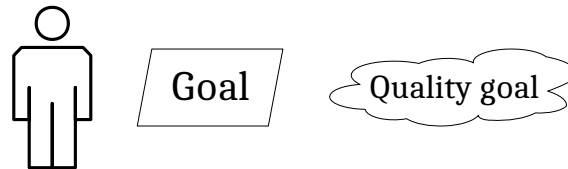


Figure 3.4: Selection of AOM notation elements

The AOM methodology allows technical- and non-technical stakeholders to model complex systems by capturing and understanding their functional- and non-functional requirements. An AOM goal model relies on three main elements to capture the system requirements and goals as illustrated in Figure 3.4. Involved stakeholders are represented as sticky men, usually used solely for human entities, but this work also comprises IoT devices, agents, and infrastructure components. Parallelograms depict functional requirements and are referred to as goals. Non-functional requirements are depicted as clouds and refer to as quality goals of the modeled system. The AOM goal model follows a tree-like hierarchy with the root value proposition of the modeled system at the top. Subsequently, this main goal is decomposed into sub-goals, where each sub-goal represents an aspect for achieving its parent goal [94]. The goals are further decomposed into multi-layered sub-goals until the lowest atomic level is reached. Additionally, roles and quality goals may be assigned to goals and are inherited to lower-level goals.

### 3.5.3 AOM Model

Figure 3.5 presents the AOM goal model of the *Rechained* protocol. The main objective of *Rechained* is to disincentivize and price the cost of Sybil node attacks. The main goal is further decomposed into multi-layered sub-goals until the lowest atomic sub-goal is reached. In the context of *Rechained*, the main goal is further divided into the following sub-goals: *Create deposit transaction*, *Mine transaction*, *Create identity proof*, *Validate identity proof* and *Revoke identity proof*. The five quality goals *secure*, *correct*, *tamperproof*, *entity agnostic* and *automated* are attached to the overall main goal of the goal model, meaning they are relevant and inherited to all sub-goals. The quality goal *reliable* pertains to the three sub-goals of *Mine transaction*, *Validate identity proof* and *Revoke identity proof*. In addition, we list three different roles: The *user* – either a human, or machine -, the *mining* entity that performs the PoW calculations of the underlying blockchain and the *validator* who validates an identity proof once received.

Next, the AOM behavior model refines the AOM goal model for specific agents and activities. A behavior model in AOM has two parts: An agent behavior model is coupled with a behavior interface model [90]. The former describes the rule-based behavior of an agent, while the latter

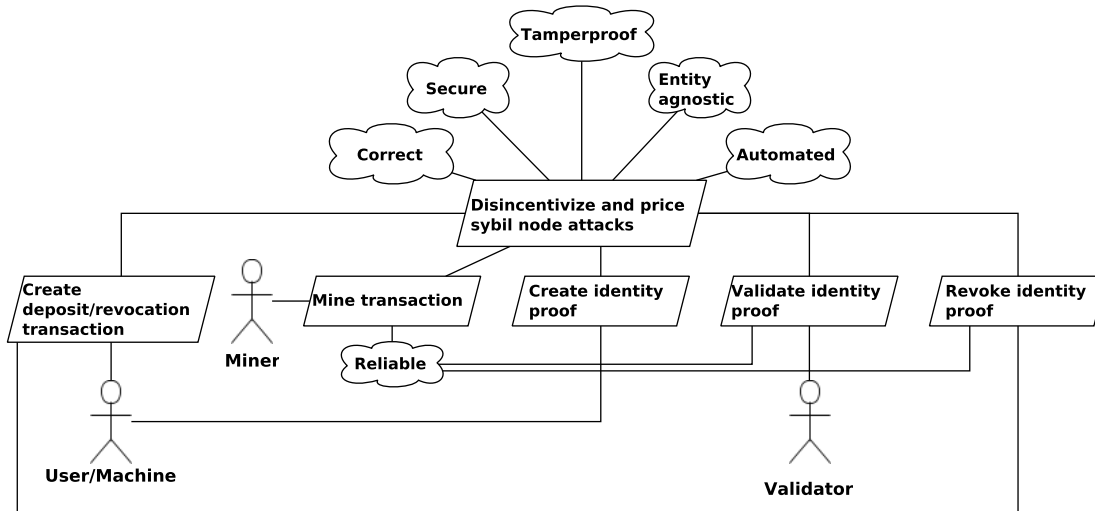


Figure 3.5: Rechaind top-level AOM goal model – extension of [5]

focuses on identifying activities with associated triggers, preconditions and post-conditions [89].

Table 3.2 presents the behavior interface model of the goals depicted in the goal model of Figure 3.5. Each activity is listed with its corresponding trigger, optional pre-conditions and its post-conditions. The execution of an activity is either triggered by an event, or by a pre-condition after the occurrence of an event [89].

The *Create Deposit Transaction*-activity is triggered after providing the required network input parameters as well as a machine identity (e.g., a network node) and a wallet. Afterwards, as part of the *Mining*-activity, the resulting deposit transaction is mined into a new block. During the *Create Identity Proof*-activity, the block, the machine identity and the provided wallet are used to create an identity proof for the node. Finally, the created identity proof is validated as part of the *Validate Identity Proof*-activity which takes an identity proof, the network parameter, the machine identity and the initial wallet to determine whether the identity proof is valid.

### 3.5.4 Mapping AOM Models to CPN Models

Mapping the created AOM goal and the AOM behavior interfaces to CPN is the final step necessary to derive a CPN model of *Rechaind*. Table 3.3 and Figure 3.6 illustrate the mapping heuristic of AOM goal models to CPN models as well as the mapping of behavior interface models to CPN models.

In the CPN model, the *Rechaind* protocol execution is modeled using places and transitions connected by directed arcs. The goals of the AOM goal model are mapped to rectangular CPN transitions. Double-boarded transitions are used to indicate sub-goals and hierarchically structured

Table 3.2: Behavioral interfaces of activities for *Rechained* – extension of [5]

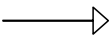

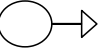
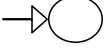
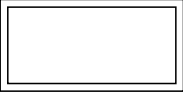
Activity	Trigger	Pre-Condition	Post-Condition
Create Deposit Transaction	User wants to create a deposit transaction	Network parameters, machine identity and machine wallet	Network parameter, machine identity, machine wallet, deposit transaction
Mining	Received deposit/revocation transaction	Deposit/revocation transaction, previous block hash, deposit/revocation wallet and blockchain difficulty target	Block, previous block hash, blockchain difficulty target, deposit/revocation wallet
Create Identity Proof	Deposit transaction mined into block and user wants to create new identity proof	Block with deposit transaction, machine identity and machine wallet	Identity proof, machine identity, machine wallet
Validate Identity Proof	Incoming identity proof	Identity proof, network parameter, machine identity and machine wallet	Boolean statement whether the provided identity proof is valid, or not
Create Revocation Transaction	User wants to create a revocation transaction	Identity proof, network parameter, machine identity and machine wallet	Network parameter, machine identity, machine wallet, revocation transaction and identity proof
Revoke Identity Proof	Revocation transaction mined into block	Block with revocation transaction, identity proof, identity revocation list, machine identity and machine wallet	Identity proof, machine identity, machine wallet and identity revocation list

CPN modules. As illustrated in Figure 3.6, the CPN modules and sub-modules of the overall CPN model map to the relation between goals and sub-goals of the AOM goal model. The triggers and pre-conditions of the AOM behavior interfaces are displayed as places with outgoing arcs, while post-conditions are represented by places with incoming arcs [89].

The complete and formalized *Rechained* CPN model as derived from the AOM goal model and the AOM behavior interfaces and implemented using CPN-Tools is shown in Figure 3.7.

The CPN model consists of six transitions derived from the five sub-goals of the top-level AOM

Table 3.3: Notation mapping CPN to AOM – based on [89])

Notation	Name
	Connecting Arc
	Sub-goal or Activity
	Trigger or Precondition
	Postcondition
	Goal

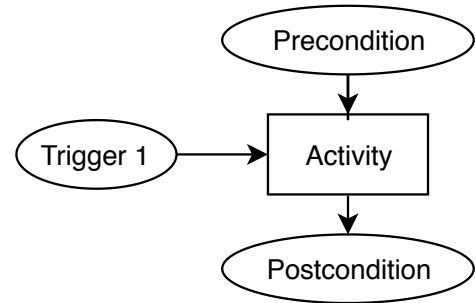


Figure 3.6: Mapping a behavior interface model to a CPN model – based on [95] and [96]

goal model. The protocol flow starts on the left-hand side of Figure 3.7 with the *Create deposit transaction*-transition. An infrastructure provider, user, or machine that wishes to create a new identity triggers the transition by providing the required network parameters as described in previous sections as well as information relating to the entity’s identity (in the CPN model, a machine is assumed), i.e., a machine identity consisting of a DID and a public/private key in addition to a wallet that corresponds to the used key pair. In case the wallet balance is sufficient to make a deposit, a matching deposit transaction with the target deposit address is created via the *Create deposit transaction*-transition. After that, the transaction is mined into a new block of the underlying blockchain platform (*Mining*-transition). The resulting block contains a BlockID, the hash of the previous block, the blockchain’s difficulty target, and a list of included transactions. Once the block is mined, an identity proof is created (*Create identity proof*-transition) according to the specifications described in previous sections. Next, the resulting identity proof is checked for validity while passing the *Validate identity proof*-transition resulting in a Boolean representation of the validation process’s success or failure. Subsequently, a valid identity proof may be revoked by the owning entity. To do so, the *Create revocation transaction*-transition creates a revocation transaction similar to the deposit transaction, which is mined into the blockchain. Afterward, the *Revoke identity proof*-transition allows to revoke an identity proof, register the revoked proofID with the *IdentityRevocationList*. Once revoked, the identity proof validation fails, and the CPN model terminates.

The CPN token color sets, names, and abbreviations used in the model are introduced in the following sections, while the complete and executable CPN model is available in Section 3.8.

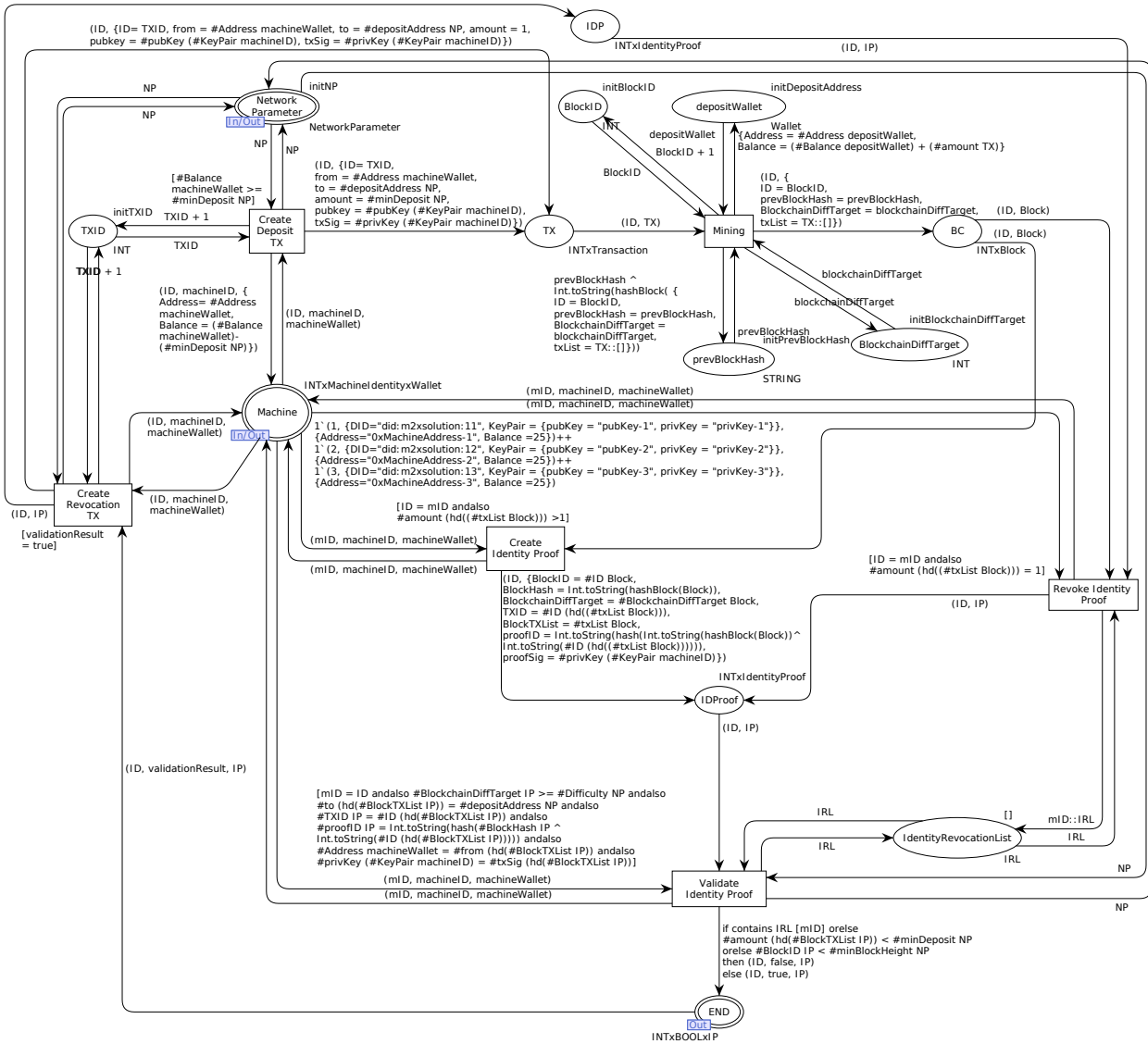


Figure 3.7: Rechaind CPN model – extension of [5]

### 3.5.5 Protocol Semantics

Next, we introduce the CPN token color sets, names and acronyms of the *Rechaind* CPN model. CPN token colors represent the data structures of data objects that are used to illustrate the data flow throughout the CPN model. Exemplary acronyms, names and the description of the token colors of the CPN model are presented in Table 3.4. The first column specifies the name, followed by a short description in the second column. The last column lists information concerning the data types. A complete list of all acronyms, names and abbreviations as well as the description of the

token colors of the *Rechained* CPN model is available in Section 3.8.

Table 3.4: Exemplary acronyms, names and description of token colors of the *Rechained* CPN model – extension of [5]

Token Color	Description	Type
KeyPair	Key pair	(pubKey, privKey)
Wallet	Blockchain wallet	(Address, Balance)
NetworkParameter, NP	Rechained network parameter	(Difficulty, minBlockHeight, minDeposit, depositAddress)
Difficulty	Minimum PoW difficulty for an identity proof as defined by the network operator	Integer
minBlockHeight	Minimum block height as defined by the network operator	Integer
minDeposit	Minimum deposit to be made for an identity proof as defined by the network operator	Integer
depositAddress	Deposit address as defined by the network operator	String
Transaction, TX	Structure of a deposit transaction	(ID, from, to, amount, pubKey, txSig)
Block	Blockchain block	(ID, prevBlockHash, BlockchainDiffTarget, txList)
IdentityProof, IP	Identity proof	(BlockID, BlockHash, BlockchainDiffTarget, TXID, BlockTXList, proofID, proofSig)
proofID	proofID as specified by the protocol	String
MachineIdentity	Machine entity identity	(DID, KeyPair)
depositWallet	Deposit wallet as defined by the network operator	Wallet
machineWallet	Machine's wallet	Wallet
IdentityRevocation-List, IRL	List of IDs of revoked identity proofs	[Integer]
validationResult	Result of the identity proof validation	Boolean

### 3.6 Evaluation and Discussion

The following section focuses on evaluating the *Rechained* protocol. First, Section 3.6.1 and Section 3.6.2 focus on the evaluation of the security guarantees provided by *Rechained*. Those mainly depend on the target difficulty level as well as the token price of the underlying PoW blockchain.

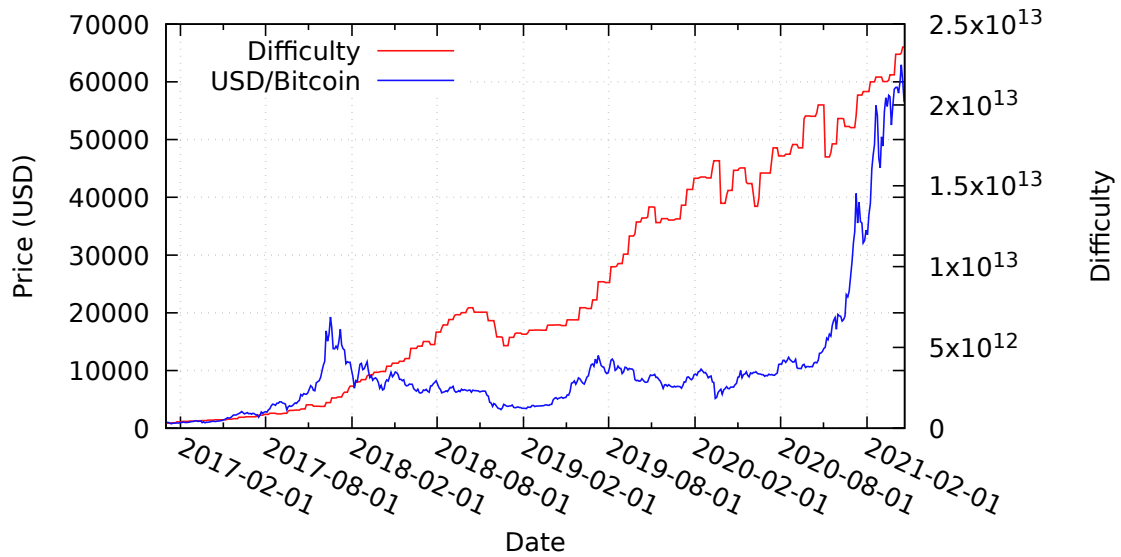


Figure 3.8: Average daily price of Bitcoin in USD and block difficulty level between January 2017 and April 2021 – partially based on [5, 19], Data Source [3]

In the context of this work, we choose the Bitcoin and the Ethereum blockchain as the most popular and utilized PoW chains to deploy a fictional network of *Rechained* nodes. The combination of evaluating both blockchains covers important corner cases of changing difficulties and token prices, such as increasing and decreasing difficulty in combination with sudden price declines and raises. Our evaluation focuses on the time period from January 2017 to April 2021. Even though neither future price developments of Ether and Bitcoin, nor the target difficulty can be reliably predicted, the analysis provides an intuition on the historical worst-case performance of *Rechained*.

Afterwards, Section 3.6.4 presents the results of the state-space analysis of the *Rechained* CPN model. Subsequently, Section 3.6.5 presents an Ethereum- and Bitcoin-based proof-of-concept implementation. Finally, Section 3.6.6 critically discusses the evaluation results.

### 3.6.1 Bitcoin Price and Difficulty Analysis

Figure 3.8 presents the target difficulty level and the token price of the Bitcoin blockchain between January 2017 and April 2021. The target difficulty level is steadily increasing with minor decreases in October 2018, October 2019 and March 2020 – the last one caused by the global COVID-19 pandemic. The price of the Bitcoin token follows a similar pattern but with a high-price peak at the end of December 2017 and higher volatility. The lowest token price (ignoring the low price at the beginning of our evaluation period) – caused by the largest price-drop – occurs in January 2019. Very recently, throughout the beginning of 2021, the price has spiked upwards again significantly.

Section 3.4 discusses the lower bound security guarantees of *Rechained*, which depend on the

minimum target block difficulty and the lowest token price equivalent per block that occurred during the existence of a particular network. Figure 3.8 shows that all nodes joining a hypothetical network later than January 2017 have higher security guarantees than the initial bootstrapping nodes due to an increased block difficulty and token price – even despite substantial declines of the Bitcoin price and the target difficulty levels between the end of 2017 and the end of 2018, as well as a subsequent decline in early 2020; the lowest price and difficulty levels still remains above the values at the time of the network initialization. Nonetheless, since the minimum price per identity is determined by the network operator, the token price and the difficulty level only represent a theoretical measurement for security guarantees. Yet, a higher token price results in an increased block price, which again results in higher attack costs.

Deploying a network at the lowest price of our evaluation period results in subsequently higher security guarantees after price gains. However, deploying a network at the highest point in Bitcoin's price history, at the beginning of November 2017 – days before the decreasing Bitcoin price as illustrated in Figure 3.8 – decreases the security guarantees, and makes it less expensive to introduce new identities into the system for a short period of time. Again, the actual pricing of identity proofs depends on the network operator determining the minimum price of an identity. For practical reasons, it is likely that most operators pick minimum values below the maximum determined in Section 3.4.2, allowing for difficulty declines without affecting the security guarantees of a given network. In case of substantial price and difficulty declines, e.g., between the end of 2017 and the end of 2018, a price and difficulty network parameter update is recommended.

Upwards price spikes like in December 2017 and the beginning of 2021 can present a concern with regards to the usability of a network using *Rechained*. A sharp increase in price will also increase the costs to join the network. If a network is created during a price spike, the operator may elect to configure the required amount to be lower to compensate, but if a network is created and configured before such an increase, it may cause issues. To address this, one of the proposed network parameter update mechanisms from Section 3.4.3 should be used. Overall, higher prices are beneficial to the security of *Rechained* as they allow more freedom and a higher upper bound of the dollar price per identity when choosing a price per identity as per Equation 3.3. At the same time, it remains possible to choose lower prices per identity as well.

Figure 3.8 only provides an overview of the volatility of the Bitcoin token price and the block difficulty level. Table 3.5 utilizes the same data set and analyzes the occurrence of substantial token price declines by calculating, for each possible deployment date of a hypothetical network, the highest drop in price and thus security level experienced by the network. Price drops are calculated based on the daily average Bitcoin prices at the end of each given day. A price lower than the previous day indicates a price decline while the opposite is true for increasing prices. The probability of dropping, at any point in time during the evaluation period, below 10 % of the initial security level is 0 % while there is only a 1 % chance of dropping below 20 % of the price at any point in time.



Table 3.5: Affected starting dates after which the Bitcoin price drops below a certain percentage of the given day's price between January 2017 and April 2021.

Drop to	Affected start dates
< 10 %	0.0 %
< 20 %	0.8 %
< 30 %	4.6 %
< 40 %	10.1 %
< 50 %	22.0 %
< 60 %	36.3 %
< 70 %	43.2 %
< 80 %	52.6 %
< 90 %	60.0 %
< 100 %	81.1 %

Smaller price-drops occur frequently. Almost 76 % of all starting days experience price drops of at least 10 % at some point in our analysis. However, substantial declines are rare. Most networks can tolerate some price volatility without relevant declines with regards to the provided security level. Nonetheless, large price drops or rising token prices are an issue. While price drops result in lower security guarantees, rising prices may be an issue too since they can make identities too expensive for regular users. Thus, an update mechanism for the network parameter `amount` is recommended in the context of networks that operate for a long time. We have proposed different mechanisms to perform such updates in Section 3.4.3. Even though price volatility is a major issue for most cryptocurrencies right now, the volatility level of cryptocurrencies and fiat currency is expected to converge if mass-adoption of cryptocurrencies occurs. As a result, *Rechained's* security guarantees could be expected to vary less, due to lower price volatility.

### 3.6.2 Ethereum Price and Difficulty Analysis

*Rechained* itself is a blockchain-agnostic protocol. It only requires the underlying blockchain platform to utilize a PoW-like consensus algorithm. Therefore, we also analyze the provided security guarantees for hypothetical *Rechained* networks when deployed on the Ethereum platform during the same time period as above. Figure 3.9 illustrates the Ether token price and the Ethereum block difficulty between January 2017 and April 2021. Similarly to the Bitcoin token price, the Ethereum token price also increased massively between January 2017 and the end of December 2017, followed by a significant drop until April 2018. Subsequently, a price recovery in May 2018 is followed by a further decline until December 2018, before slowly starting to increase again until February 2020, followed by a decline caused by the COVID-19 pandemic at the end of the evaluation period and another steep incline at the beginning of 2021. The block difficulty increases steadily until October 16, 2017 before a sudden drop due to a difficulty adjusting hard-fork of the Ethereum network [51]. Another drop is shown at the end of 2019. Despite substantial reoccurring declines of the block difficulty level, even the reduced values were higher than the initial level in

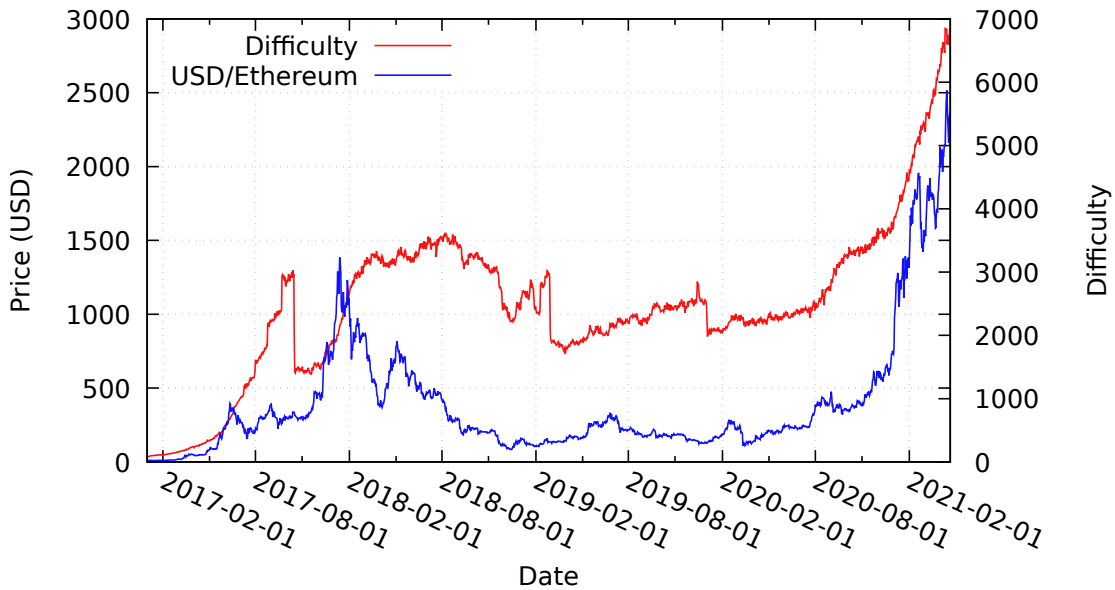


Figure 3.9: Average daily price of Ether in USD and block difficulty level between January 2017 and April 2021 – partially based on [5,19], Data Source [4]

January 2017. During the steep incline in price in 2021, the difficulty also increases accordingly. It appears that in the case of Ethereum, the difficulty is more closely linked to price than in the case of Bitcoin, where it increased steadily even through a period of lower prices.

The overall behaviour is similar to Bitcoin. Hence, the security guarantee evaluation results are similar to the evaluation of the hypothetical Bitcoin-based *Rechained* network assumed in Section 3.6.1. Network nodes deployed in January 2017 with the bootstrapping difficulty level and token price are cheaper and easier to create in terms of identity price and block difficulty. All nodes deployed at later points in time provide higher security guarantees. Identity proofs created briefly before and/or after the Ethereum difficulty adjustment, as well as the difficulty drop in early 2019 and late 2019, are less difficult to create than identities created at later on. The same applies for the price of identities both before and after the price declines of Ether in 2018 as illustrated in Figure 3.9. As Ethereum seems to have sharp drops in difficulty more often than Bitcoin, it seems prudent to choose the initial `minDifficulty` significantly lower than the current difficulty, to account for such drops in the future, unless timely network parameter updates can be guaranteed. With respect to the recent increase in price with the beginning of 2021, the same considerations as with Bitcoin apply as well.

Analogously to Table 3.5, Table 3.6 analyzes the occurrence of substantial token price declines for the Ethereum network by calculating, for each potential deployment date of a hypothetical network, the highest drop in price and thus security level experienced by the network. The probability of dropping, at any point in time during the evaluation period, below 10% of the initial security

Table 3.6: Affected starting dates after which the Ethereum price drops below a certain percentage of the given day's price between January 2017 and April 2021.

Drop to	Affected start dates
< 10 %	2.5 %
< 20 %	14.8 %
< 30 %	25.9 %
< 40 %	31.7 %
< 50 %	40.6 %
< 60 %	45.8 %
< 70 %	54.3 %
< 80 %	60.3 %
< 90 %	71.5 %
< 100 %	89.3 %

level is 3.3 %. Drops below 20 % has would have occurred for 19.6 % of possible starting dates. Almost half of the affected dates experience price drops of 50 %. These values are much higher than the values for the Bitcoin network as presented in Table 3.5. Thus, the Bitcoin-based *Rechained* solution provides more certainty and more steady security guarantees than the Ethereum-based implementation – at least based on the analysis of past events which is, of course, not indicative of future prices and volatility.

### 3.6.3 Transaction Fee Analysis

Storing data on a PoW-base chain such as the Bitcoin blockchain incurs costs that correlate with the transaction size. As suggested in [68], a revocation transaction in the Bitcoin network with a payload of 40 bytes has a size of fewer than 283 bytes. Since *Rechained* does not depend on fast transaction processing, it is unnecessary to aim for a high mining priority by paying high fees. In March 2020, the cost of a transaction averaged around \$0.001751 per byte. As a result, a revocation transaction of 283 bytes on the Bitcoin platform cost \$0.496 [68]. However, the most recent Bitcoin hype increased the transaction fees heavily. In March 2021, the cost for the same revocation transaction increased to \$6.281 with an average price of \$0.022194 per byte [97, 98].

Despite the drastic increase in transaction costs, we do not expect any long-term disadvantages for *Rechained* for several reasons. First, *Rechained* does not explicitly rely on the Bitcoin blockchain. Instead, all PoW-based blockchain platforms can be used as an alternative. Second, we expect the transaction fees to decline in the long run, either due to an end of the hype or as part of the widespread adoption of Bitcoin as a payment processing network. Finally, high transaction fees are addressed in various proposals that aim to reduce the fees, e.g, [99, 100].

### 3.6.4 CPN State-Space Analysis

Next, we evaluate the *Rechained* CPN model by performing a state-space analysis, which we use to derive model properties and explain their implications. A state-space calculates a directed graph of all reachable states and state changes of a given CPN model. The nodes of the graph correspond to the set of reachable markings, and the arcs correspond to occurring binding elements [56]. Properties of the CPN model – and the system presented by the model – are deduced from the resulting graph. The state-space analysis used in this work is generated using the built-in functionalities of CPN-Tools. Subsequently, we calculated the Strongly Connected Component (SCC) of our CPN model based on the state-space analysis's previously generated directed graph. The nodes of the SCC are "obtained by making a disjoint division of the nodes in the state space such that two state-space nodes are in the same SCC if and only if they are mutually reachable, i.e., there exists a path in the state space from the first node to the second node and vice versa" [56]. Properties derived from the SCC may imply one or more cycles in case the SCC contains fewer nodes than the state-space graph of the CPN model.

The state-space analysis results and selected properties derived from the analysis of the *Rechained* CPN model are presented in Table 3.7, while the complete state-space analysis is available in Section 3.8.

Table 3.7: State-space analysis results of the Rechained CPN model

Loops	Home markings	Dead markings	Dead transitions	Live transitions
No	No	Yes	No	No

As shown in Table 3.7, none of the tested modules contain any loops. Thus, no infinite occurrences of execution paths in the state-space graph exist, which guarantees the termination of the model. The state-space analysis also shows the absence of any home markings. A home marking is a marking that can be reached from any other reachable marking, meaning that it is impossible to have an occurrence of a sequence that cannot be extended to reach the home marking. The occurring dead markings are caused by customized input values that prevent a state-space explosion. "A dead marking is a marking in which no binding elements are enabled" [56]. The existence of at least one dead marking guarantees a termination of executable actions at a certain point, thereby preventing infinite runtime. The existence of a dead marking implies that the CPN model does not have a live transition. "A transition is live if from any reachable marking we can always find an occurrence sequence containing the transition" [56]. Finally, the state-space analysis did not yield any occurrences of dead transitions. A transition is considered dead if there is no reachable marking that enables the transition. Therefore, in the context of the *Rechained* CPN model, all transitions of the model can be potentially enabled at a certain point during the protocol execution [56].

### 3.6.5 Proof-of-Concept Implementation

Previous sections evaluated *Rechained* with regards to the provided security guarantees based on the underlying blockchain platform and cryptocurrency. Moreover, a state-space analysis using a formal CPN model was conducted. However, the real-world applicability and feasibility within the context of IoT devices has not been covered yet. Thus, a proof-of-concept implementation of the original Unchained protocol based on the Bitcoin and Ethereum blockchain was created – the repositories are available online<sup>56</sup>. The implementations use public/private key pairs instead of DIDs thereby reducing development overhead and do not yet contain revocation support. However, this does neither affect provided security guarantees, nor the performance of the proof-of-concept. The implementations are evaluated using common IoT hardware, i.e., the Raspberry Pi 3 platform.

The size of an identity proof varies depending on the blockchain platform as well as the size of the block containing the proof transaction. Bitcoin-based proofs require between 10KB-50KB of storage and can be verified in about two seconds using a Python based implementation. Both runtime and proof size could most likely be optimized further for actual use in production. The Ethereum proofs consume around 50KB-150KB of storage, but take around 60 seconds to be verified. The slow verification processing is caused by a deliberate design choice of the Ethereum hash function Ethash [101], which was designed to achieve ASICS resistance. Therefore, deploying *Rechained* on a Bitcoin-like PoW blockchain is more practical even though other PoW blockchain platforms that do not rely on Ethash or similar algorithms with the same property are suitable as well.

Since the identities are deployed to networks by the network operators or device manufactures, we assume that the proofs itself are not generated on the actual devices – hence, we do not conduct any performance benchmarks for creating proofs on the Raspberry Pi 3.

### 3.6.6 Discussion

While the integration of DIDs in the context of *Rechained* allows for an effective mechanism to disincentivize and accurately price Sybil node attacks, the presented solution still misses a comprehensive integration into industry-standard implementations, e.g., for vehicle-specific use cases, the recently presented vehicle identity standard [102]. Integrating the vehicle identity standard, which uses a DID-structure, with *Rechained* identity proofs is desirable.

Evaluation limitations of the *Rechained* CPN model result from the customized input statements of the model as well as the modeling process itself, which requires several simplifications, e.g., neither the Bitcoin nor the Ethereum consensus algorithm and mining process were implemented in the CPN model. Furthermore, we simplified data structures of the *Rechained* protocol and

---

<sup>5</sup><https://github.com/bleidingGOE/unchained-cli-btc> or <https://codeocean.com/capsule/7153326/tree/v1>

<sup>6</sup><https://github.com/bleidingGOE/unchained-cli-eth> or <https://codeocean.com/capsule/1342035/tree/v1>

the blockchain platform, e.g., no Merkle trees, blocks have no nonce, a simplified calculation of *Rechained's* proofID calculation. Further limitations originate from the limited scripting capabilities of CPN-Tools, e.g., the implemented hashing function does not provide real hashing properties. Similar applies to the symbolic implementation of public-key cryptography which only allows for the symbolic signing of hashed data records.

Since *Rechained* is a cryptocurrency-based protocol, it suffers from volatile cryptocurrency prices that complicate its everyday use.

Finally, *Rechained* may allow network operators to determine the cost of a Sybil node attack, but it does neither fully prevent such an attack, nor does it help to actually detect the Sybil nodes. However, depending on the scenario, Blockchain analysis techniques [103] could be employed to help identify identities created by the same entity.

### 3.7 Conclusion and Future Work

Once Sybil nodes have entered a network, detecting them can be an error-prone and cumbersome process. In this work, we propose *Rechained*, which introduces a direct and adjustable per-identity cost and disincentivizes Sybil attacks even before they start. Our protocol is fully decentralized and uses public blockchains to offload the PoW process, while at the same time providing offline verification and identity revocation mechanisms. *Rechained* is formally evaluated by means of CPNs, includes revocations and support for SSI.

Our analysis shows that circumventing the security mechanisms of *Rechained* incurs equivalent or higher costs for an attacker than following the protocol as intended, showing the suitability of the scheme for disincentivizing the creation of spurious or malicious identities.

To create new identities, which can even be used in scenarios where eclipse attacks have to be considered, *Rechained* binds blockchain-based wallet address (i.e. cryptographic public and private key pairs) to identities. During this process, an identity proof is generated, which can be verified offline. The identity proof further contains an unpredictable proof ID. Identity creation has an adjustable price tag and different types of payment (i.e., to a network operator, to charity or proof of burn) are described. When connecting to a network secured by *Rechained*, nodes exchange identity proofs and verify them before proceeding to connect. The generation of separate revocation proofs is also supported.

Cryptocurrencies are somewhat notorious for their highly volatile prices, but our analysis shows that even in the worst case, this volatility impacts the security of *Rechained* only to a very limited degree. Network parameter update mechanisms allow further adjustment in case of price or difficulty changes. For networks with at least intermittent Internet connectivity, we describe different ways of adjusting network parameters to accommodate a changing environment. At the same time, we detail approaches that can be used to keep the network's configuration in working

order even in the complete absence of Internet connectivity, making *Rechained* suitable for many different operating environments.

We also formalize the protocol by means of CPNs to analyze its properties and behavior and provide a basic proof-of-concept implementation that shows the suitability of *Rechained* for low powered nodes, such as those that may be found in Internet of Things applications.

In the future, we plan to investigate the feasibility of implementing our scheme on blockchains based on other consensus mechanisms than PoW as it is often criticized for its high energy consumption. We also plan to evaluate further IoT use-cases for *Rechained* in the real world. Finally, employing blockchain analysis techniques should be investigated as it could be a promising approach for identifying possible attacks that occur despite the economic disincentivization.

### **3.8 Appendix: Rechained Protocol Formalization**

All data is available via IEEE DataPort: <http://dx.doi.org/10.21227/e5tk-3y77>





## Chapter 4

# Robustness Enhanced Sensor Assisted Monte Carlo Localization for Wireless Sensor Networks and the Internet of Things

### Abstract

*With distributed sensor systems commonly found in Wireless Sensor Networks or the Internet of Things, knowing the location sensor data was acquired from is very important, especially in scenarios with mobile sensors. Range-free Monte Carlo Localization based approaches are very energy efficient and do not require additional hardware beyond a radio, which is found on sensor nodes anyways. However, the use of motion sensor data based dead reckoning greatly improves the accuracy of location estimates and increases robustness against faulty or malicious actors within the network. In this work, we propose Robustness Enhanced Sensor Assisted Monte Carlo Localization (RESA-MCL). We show RESA-MCL's effectiveness with respect to both general localization accuracy and robustness against malicious attacks or malfunctioning nodes. To evaluate and compare our scheme against existing approaches, we introduce three attack models based on malicious anchor nodes. The performance of RESA-MCL is evaluated under these attack models and our approach outperforms existing schemes in both very low and higher anchor node density environments, achieving a localization error of 0.5 with an anchor density of 0.33. Overall, RESA-MCL outperforms comparable approaches at lower anchor densities with up to 48% lower localization error and demonstrates strongly increased robustness against attacks with minimal computational overhead.*

### 4.1 Introduction

In today's world, more and more IoT devices with various types of sensors, as well as WSN, are getting deployed to cover a wide range of scenarios, from smart homes [104], decentralized initiatives by volunteers for measuring air quality [105], [106], over industrial uses [8] to wildlife

monitoring [7]. To make sense of the gathered data, it is important to know where it was measured. In the case of, for example, a WSN with fixed nodes, the installation points of each sensor can be noted, but many applications rely on mobile sensors, which makes it necessary for sensor nodes to be able to determine their locations dynamically.

The most common approach to this is the use of the Global Positioning System (GPS). However, the use of GPS has a number of disadvantages. The sensors are relatively costly and consume high amounts of power. They also rely on being able to receive satellite signals, which makes indoor operations impossible and also leads to reduced accuracy in certain outdoor environments. To mitigate the first two points, a solution is to equip only a small subset of nodes with GPS sensors. These nodes then act as so-called seed or anchor nodes, which assist other nodes in localizing themselves. Instead of using mobile anchor nodes equipped with GPS sensors, the use of static anchors with preset locations is also a common approach.

Different types of localization algorithms exist. They can be mainly divided into range-based [107], [108], [109], [13], [10] and range-free approaches [110], [12], [111], [112]. In range-based approaches, unknown nodes (non-anchor nodes trying to localize themselves) must actively determine the distance to anchor nodes or the angles of incoming radio signals. Common measurements used in such approaches include ToA [10], TDoA [11], AoA [13], and RSS [9].

A popular technique is to use RSS along with an appropriate propagation model to estimate the distance between an unknown node and an anchor node. This solution is based on the assumption that the RSS decreases proportionally with increasing distance from the transmitter. However, certain limitations need to be considered. For example, in TDoA-based solutions highly precise clock synchronization between nodes must be guaranteed; it's important to consider the multipath, Non Line of Sight (NLoS) conditions and array calibration in AoA-based solutions; radio noise levels, multipathing and measurement errors can affect the performance of RSS-based solutions [10]. Generally speaking, range-based approaches typically require additional, specialized hardware, clock synchronization and have higher power consumption, which enables the active measurements that have to be performed by unknown nodes. In addition, the inherent limitations of each type of measurement can affect localization accuracy in certain situations.

In order to reduce complexity, hardware dependency and energy costs, research is conducted on range-free solutions which are usually based on connectivity alone. Since no active measurements are required from the unknown nodes, these approaches are easier to implement and also have lower deployment costs. A well-known representative approach in this category is the Monte Carlo Localization (MCL) algorithm, which was adapted for localization in mobile WSNs by Hu and Evans in 2004 [110]. Unlike approaches designed for partially or fully static networks, MCLwork to move arbitrarily over time and uses the movement to improve localization performance. The probability distribution of each node's current position is represented as a set of weighted samples (particles) in MCLare outside the communication range of anchor nodes are eliminated by a

Bayesian filtering process. The estimated node's location is the average of all remaining samples after the filtering process. MCLdware and is suitable for both mobile and static scenarios. However, MCL's performance quickly degrades in situations where connectivity between unknown nodes and anchor nodes only occurs rarely. Therefore, guaranteeing a high anchor node density is very important when applying MCL.

The various approaches presented so far have not considered the security aspects of localization. If malicious nodes that publish erroneous locations are present in the network, these approaches may not work as well as shown in their experiments. Even in the absence of malicious nodes, performance degradation or even collapse of the entire system may occur if a subset of the anchor nodes does not function properly. For example, in approaches where anchor nodes are statically distributed in the environment, if one or multiple of the anchor nodes do not work, normal nodes in the vicinity of these anchor nodes will not be able to obtain the information required for localization such as RSS, TDoA, etc., which will eventually lead to localization failure. Therefore, to make the proposed approaches more suitable for real-world scenarios, anchor node malfunctions need to be considered in the development of the localization algorithms.

In this work, we propose RESA-MCL, which both achieves a higher localization accuracy compared to previous schemes and also improves the localization scheme's robustness against incorrect information being broadcast by malicious anchor nodes. To achieve this, RESA-MCL continuously employs dead reckoning, as described by Hartung et al. [12], instead of only using it when out of anchor range like the original Sensor Assisted Monte Carlo Localization (SA-MCL) scheme. RESA-MCL detects malicious anchor nodes through motion-based plausibility checks and limits the influence that malicious nodes can exert on the location estimate through a novel particle subsetting technique. We thoroughly analyze the performance of RESA-MCL under different attack models with up to 90 % of anchor nodes acting maliciously and motivate parameter choices in data-driven manner.

The contributions of our paper can be summarized as follows: (1) We propose the RESA-MCL scheme, which achieves higher localization accuracy (up to 48 % lower error than comparable recent approach) in fully mobile low anchor density situations than comparable schemes with low computational complexity while also including robustness enhancements to mitigate attacks. (2) Specific attack models for scenarios with malicious or malfunctioning anchor nodes are defined. (3) RESA-MCL and previous approaches are evaluated and compared under three different attack scenarios. (4) Robustness enhancements in RESA-MCL greatly improve its performance in all attack models compared to approaches without robustness enhancements. (5) Thorough experimental evaluation including ablation experiments are used to verify the favorable properties of the scheme, such as low localization error at low anchor densities and resistance against attacks. (6) An optimized version of the original MCL [110] simulator—including implementations of SA-MCL [12] and RESA-MCL—is provided to ease future evaluation and comparison with other approaches.

The structure of this paper is organized as follows: Section 4.2 reviews related works. In Section 4.3, we describe our proposed scheme in detail. Our evaluation methodology and results are shown in Section 4.4. Finally, we present conclusions and future works in Section 4.5.

## 4.2 Related Works

Various localization algorithms for WSNs/IoT have been proposed in recent years. Concerning our work, the consideration of security aspects in localization is the most relevant factor. Therefore, we present overviews of related works classified into four groups: range-based, range-free, Artificial Intelligence (AI)-based and security-aware approaches.

### 4.2.1 Range-based approaches

As introduced in section 4.1, many range-based localization approaches have been proposed. Luo et al. proposed an RSS-based Localization using Uncertain Data Mapping (LUDM) for WSN [107]. Simulation results show that the proposed approach outperforms other solutions in terms of the absolute mean localization error. However, the four anchor nodes are statically fixed at the corners of the experimental area. Moreover, the localization accuracy may decrease greatly when resorting to the RSS attenuation model to improve the generality in unknown localization environments.

In 2019 Wang et al. proposed a Time of Flight (ToF)-based localization algorithm for asynchronous WSN [108]. The simulation results show that the proposed approach outperforms conventional algorithms in terms of localization accuracy. However, again the anchor nodes are statically deployed in the network and localization depends on a centralized server, which must provide sufficient computational capacity for estimating clock skews and performing the localization procedure. This can cause the entire localization process to break down due to a single point of failure if server issues occur.

In 2020, the same authors proposed another time-based joint synchronization and localization algorithm for asynchronous WSN which utilizes TDoA [109]. It also relies on a centralized server for launching the localization procedures. Simulation results show that this new approach is superior in scenarios where anchor positions are imperfectly known. However, its centralized structure means that there is a single point of failure.

In 2021, Ding et al. proposed an indoor localization algorithm based on Error variance and measurement Noise Weighted Least Squares, named ENWLS [13], which is a weighted algorithm for localization in 3D WSN based on RSS/AoA measurements. Simulation results show that ENWLS outperforms other existing hybrid RSS/AoA localization algorithms when there are more than three anchor nodes in the scenario. This implies that ENWLS is not reliable in case of malfunctioning of anchor nodes. Moreover, multipath effects and NLoS are not considered.

Range-based approaches aim at enhancing localization accuracy by measuring ranges between

unknown nodes and anchor nodes. However, to perform such measurements, they often require strict clock synchronization or special hardware. Additionally, multipathing or other environmental conditions may interfere with such systems. This makes them harder and more expensive to deploy.

### 4.2.2 Range-free approaches

As presented in Section 4.1, MCL [110] is a representative range-free localization approach that requires no additional hardware. More importantly, MCL allows all nodes including anchors in the network to move arbitrarily over time. Node mobility is leveraged to further increase localization accuracy through the use of a particle filter. Strong mobility support is one of the outstanding features of MCL-based approaches.

Inspired by MCL, many range-free approaches have been proposed, improving localization accuracy and sampling efficiency [113], [111]. However, these solutions cannot solve the problem of localization failure due to connectivity loss caused by dynamic changes in the network topology and low anchor node density. In order to solve this problem, Hartung et al. proposed the Sensor-Assisted Monte Carlo Localization (SA-MCL) method in [12]. The algorithm can specifically handle the temporary loss of all connectivity to anchor nodes in the network, which greatly improves the shortcomings of the original MCL algorithm. Low-cost 9-axis Inertial Measurement Unit (IMU) sensors are used to perform dead reckoning to bridge periods without connectivity to anchor nodes.

In order to improve localization accuracy under low anchor node density, Qin and Zhu [111] adapted MCL through the use of the Differential Evolution (DE) optimization algorithm (MCL-DE). Instead of using the regular sample filtering and resampling algorithms, MCL-DE selects the sample weight as the objective function for optimization and implements the differential evolution algorithm to obtain valid samples for location prediction. The authors found that MCL-DE has enhanced localization accuracy. However, the computational and communication costs of the proposed scheme are not studied and no security implications are considered.

Range-free localization approaches not based on MCL also exist. One common group of range-free approaches are those based on Distance Vector-Hop (DV-Hop). In 2020, Gui et al. [112] introduced a decentralized, range-free approach based on the DV-Hop family of localization approaches. Centralized Connectivity based DV-Hop (CCDV-Hop) and Distributed Connectivity based DV-Hop (DCDV-Hop) have relatively low computational complexity and achieve low localization error. However, malfunctioning or malicious anchor nodes are not considered and simulation results are provided only for small and medium-sized networks of up to 30 nodes. Additionally, it was only evaluated with relatively high numbers of anchor nodes (50 % of unknown nodes in the small network of 6 unknown nodes, 33.3 % in the medium network of 9 unknown nodes) with high communication ranges relative to the small simulation areas (20 m in a  $40 \times 40 \text{ m}^2$  or  $60 \times 60 \text{ m}^2$  areas), allowing each anchor node to cover a high percentage of the experimental area.

The authors do not provide an anchor node density measure.

Overall, range-free approaches trade off some localization accuracy for ease of deployment, cost-effectiveness and more generalized applicability. Since the only requirement to deploy common range-based localization approaches is the ability to decide whether a node is within radio range or not, this type of approach can be employed even with very basic hardware.

### 4.2.3 AI-based approaches

Along with range-based and range-free localization algorithms, researchers have also applied AI techniques to develop new indoor and outdoor localization solutions. Chen et al. proposed ConFi, the first Convolutional Neural Network (CNN)-based indoor WiFi localization method, in 2017 [114]. It uses Channel State Information (CSI) to build a time-frequency matrix that is utilized as the feature for localization. The authors conducted extensive experiments to select the parameters for the CNN and also show that ConFi outperforms the existing solutions. However, the amount of samples required to train the CNN is high. To apply ConFi in a new environment that is not comparable to the current one, the model must be trained with a new dataset from that environment.

Gharghan et al. [115] proposed an adaptive neural fuzzy inference system to estimate the distance between a moving bicycle (i.e., player) and a static coordinator node (i.e., coach) for the indoor and outdoor velodromes. Simulation results show that the proposed approach outperforms other state-of-the-art systems in terms of mean absolute error. However, the offline phase for Adaptive Neural Fuzzy Inference System (ANFIS) training is time-consuming and depends heavily on the complexity of the fuzzy inference system. Furthermore, the current approach requires the training of two ANFIS systems for the localization coordinates separately which increases the overall training cost.

Besides using CSI and RSS to train the neural model, researchers have also proposed approaches that apply AI techniques based on WiFi fingerprints, such as HybLoc [116], which is a hybrid indoor localization system for both room-level and latitude-longitude predictions. Simulation results show that HybLoc has better performance in terms of accuracy and precision. However, the hardware requirements for the sensor nodes to achieve a short response time in the prediction phase, which is very important to evaluate whether HybLoc is applicable in real scenarios, are not presented. In addition, without re-training the model, HybLoc is not applicable for localization in a new environment, such as a building that is not in the dataset.

Munadhil et al. [117] proposed a neural network-based localization system for WSNs in an indoor environment that is specially designed to determine the position of an Alzheimer's patient. It utilizes the RSS with respect to the anchor nodes. Simulation results show that the proposed approach outperforms other previous techniques in terms of mean localization error. However, in the experiment, the mobile node carried by the patient must be connected to a laptop to record

the RSS samples, configure the wireless connection, and supply power. This is an impractical hardware requirement for most real-world scenarios. Moreover, due to the static anchor nodes and the offline training phase, the approach cannot simply be transferred to other environments or mobile scenarios.

Overall, these AI-based approaches are computationally intensive and usually only apply to specific environments they were trained for. Furthermore, all anchor nodes in these approaches are positioned in static locations. Additionally, they do not consider security implications of malfunctioning or malicious anchor nodes, which may pose issues in real-world applications. In the end, the system complexity and the high offline training cost of these AI models should not be underestimated. However, more general and lightweight approaches also exist, which are more suitable for environments like WSNs.

#### 4.2.4 Security-aware approaches

Besides improving localization accuracy, sampling efficiency, and coping with transient connectivity loss, addressing security aspects of localization is also an active research topic. Many security-aware localization algorithms have been proposed. In [10], Xie et al. proposed a lightweight secure ToA-based localization algorithm in WSNs, exploiting the noise features caused by external distance attacks. This approach aims mainly to defend against impersonation attacks launched by external attackers.

Liu et al. proposed a Malicious Node Detection algorithm based on Clustering and consistency evaluation (MNDC) along with an enhanced secure localization version called EMDC, both of which are range-based localization schemes [118]. They use density-based spatial clustering to detect the abnormal clusters of nodes. A sequential probability ratio test is then used to identify malicious nodes that compromise the networks. The conducted simulations show that the proposed algorithms outperforms other state-of-art schemes in terms of detection accuracy and effectiveness. However, the computational overhead caused by the clustering algorithm and the sequential probability ratio test is not explained. In addition, there is no mechanism for anchor nodes that are identified as malicious to recover their reputations. As a range-based scheme, it also has additional requirements for deployment compared to range-free schemes.

Yuan et al. [15] proposed a secure Approximate Point in Triangle (APIT)-based range-free scheme in 2018, which attempts to detect Sybil nodes inside the network. SF-APIT is evaluated with a relatively low number of anchor nodes compared to unknown nodes (10%), but relatively high communication ranges (60 m in a  $300 \times 300 \text{ m}^2$  area). Under these conditions, it achieves low localization errors. However, SF-APIT is only applicable to static networks and no consideration is given to anchor nodes that behave in malicious ways without performing Sybil attacks. No anchor density measure as defined by Hu et al. [110] is provided.

Existing security-aware approaches often focus on network structure-based attacks (e.g. wormhole

or Sybil attacks), have high computational requirements, assume static networks or make strong assumptions (e.g. trustworthiness of anchor nodes) that may not be true in real-world scenarios. In many cases, where anchor or unknown nodes are assumed to behave maliciously, no specific attack model for the behavior of malicious nodes is given or such malicious nodes are only detected, but no further handling of the detected nodes is specified.

We propose RESA-MCL, which combines the advantages of range-free approaches, such as low cost and ease of deployment, with the enhanced robustness against attacks of secure localization schemes while still being very lightweight with respect to computation. Our scheme is fully decentralized, meaning that there are no infrastructure requirements beyond the nodes themselves and that no single point of failure exists. Malfunctioning or malicious anchor nodes can be detected and their effect on localization accuracy is mitigated. We define three different attack models for anchor nodes and evaluate our proposed scheme, as well as previous schemes, under these conditions. As the approach is based on MCL, RESA-MCL fully supports mobility for both unknown nodes and anchor nodes. There is no offline training phase or dependence on any location-specific data, meaning that RESA-MCL can be used in arbitrary environments, rather than being dependent on a fixed location. Our approach performs well with low numbers of anchor nodes (less than 5% of unknown nodes) and low communication ranges relative to the experimental area (50 m in a  $500 \times 500 \text{ m}^2$  area).

Overall, RESA-MCL can be deployed with minimal hardware requirements. It leverages sensor data from low cost and low powered 9-axis IMU sensors, similar to SA-MCL, to both enhance localization accuracy and allow the detection of malfunctioning or malicious anchor nodes. As RESA-MCL mainly consists of computationally inexpensive modifications to SA-MCL, the real-world power measurements presented by Hartung et al. [12] apply to RESA-MCL as well, demonstrating that its deployment on low powered IRIS sensor nodes [119] based on an 8 bit microcontroller and XBee (2.4 GHz 802.15.4) radio is feasible. In addition, since RESA-MCL does not make any assumptions about the area it is deployed in, it is suitable for both mobile and stationary networks as well as indoor and outdoor environments.

### 4.3 Localization Scheme

RESA-MCL uses the idea of the SA-MCL scheme introduced by Hartung et al. [12] as a basis and adds a number of improvements, which both increase its overall accuracy and make it more robust in networks with faulty or malicious nodes. SA-MCL itself is based on the MCL [110] approach. In the following, we shortly reiterate these approaches and finally detail the improvements made in RESA-MCL.



Symbol	Value	First use	Meaning
$V_{\text{Min}}$	10 m/s	S.4.3.2	Minimum speed of nodes
$V_{\text{Max}}$	20 m/s	L.4.1	Maximum speed of nodes
$t$		L.4.1	Time $t$
$L_t$		L.4.1	Set of particles at time $t$
$L_S, L_R$		L.4.1	Strict/relaxed set of particles
$N$	50	L.4.1	Target number of particles
$l_{t-1}^k$		L.4.1	Particle $k$ from $L_{t-1}$
$p_k$		L.4.1	Particle $k$
$r$	50 m	L.4.2	Radio range in meters
$\delta$	5 m	S.4.3.2	Increased radio range for relaxed acceptance
$r_D$	2.5	L.4.5	Anchor node plausibility check range factor (direct)
$r_I$	4.5	L.4.5	Anchor node plausibility check range factor (indirect)
$r_f$		L.4.5	A radio range factor
$\Delta\text{pos}$		L.4.4	Sensor measured movement distance of node since last MCL call
$c$		L.4.1	MEETCONDITION result
$A_D, A_I$		L.4.2	Set of anchor nodes in range (direct, indirect)
$a^i$		L.4.5	Anchor node $i$
$a_t^i$		L.4.5	Anchor node $i$ and position at time $t$
$a_{t_{\text{last}}}^i$		L.4.5	Anchor node $i$ 's last known position before time $t$
$P_i$	0	L.4.5	Anchor node $i$ 's distrust points
$d(a, b)$		L.4.2	Distance between two given particles or points a and b
$s_\phi$		S.4.3.4	Modulo value used for subsetting
$s_\lambda$		S.4.3.4	Threshold value used for subsetting

Table 4.1: Symbols with references to sections and listings.

### 4.3.1 Notation

This section provides an overview of all symbols used in the sections detailing our scheme. Table 4.1 provides short explanations, references to where each symbol is first used and, where applicable, values of constants. The "First use" column specifies which section (S.) or listing (L.) the symbol is used in first.

### 4.3.2 MCL

The MCL algorithm [110] is made up of two phases. The first phase is network communications and the second phase is a particle filter used for location estimation.

The necessary information to update location information is gathered during a phase of network

```

1: procedure MCL(attempts, enableSubset=false)
2:    $L_t \leftarrow \{\}$ 
3:   count  $\leftarrow 0$ 
4:   while size( $L_t$ ) <  $N$  and count < attempts do
5:     count  $\leftarrow$  count+1
6:      $L_S \leftarrow \{\}$ 
7:      $L_R \leftarrow \{\}$ 
8:     for all  $k \in [1, 2, \dots, N]$  do
9:        $p_k \leftarrow$  RESAMPLEINRADIUS( $l_{t-1}^k, V_{\text{Max}}$ )
10:       $c \leftarrow$  MEETCONDITION( $p_k, \text{enableSubset}$ )
11:      if  $c = \text{strict}$  then
12:         $L_S \leftarrow L_S \cup \{p_k\}$ 
13:      else if  $c = \text{relaxed}$  then
14:         $L_R \leftarrow L_R \cup \{p_k\}$ 
15:      end if
16:    end for
17:     $N_{\text{left}} \leftarrow N - \text{SIZE}(L_t)$ 
18:    if SIZE( $L_S$ )  $\geq N_{\text{left}}$  then
19:       $L_t \leftarrow L_t \cup \text{CHOOSE}(L_S, N_{\text{left}})$ 
20:    else
21:       $L_t \leftarrow L_t \cup \text{CHOOSE}(L_S \cup L_R, N_{\text{left}})$ 
22:    end if
23:  end while
24: end procedure

```

LISTING 4.1: MCL algorithm.

communications. Here, anchor nodes broadcast their locations. Unknown nodes that receive this information directly from an anchor node broadcast it again, marking it as a rebroadcast. Nodes that receive either type of broadcast store the received location data and node IDs. In regular intervals, the nodes use this collected information to update their location estimates by using a particle filter.

The particle filter itself is split into two steps: prediction and filtering. Initially, each unknown node initializes  $N$  particles. These particles are points in 2D space and represent possible locations of the unknown node. They are distributed randomly over the area of possible locations. These particles are an approximation of the probability distribution of possible node locations. During the prediction step, each particle is reassigned to a new location within the radius corresponding to the maximum speed of the node. This mechanism ensures that, if no observations are made to constrain the distribution, its uncertainty grows over time in accordance with the node's movement. In the filter step, the resampled particle is checked against received broadcasts from anchor nodes and forwarded 2-hop retransmissions of such broadcasts. If the particle is not within the radio range of all anchor nodes, from which transmissions are received, or within the ring between one and two radio ranges in the case of 2-hop retransmissions, it is discarded and a new particle is sampled instead, restarting the process. Since the original description of the algorithm leaves some

```

1: function MEETCONDITION( $p_k$ , enableSubset=false)
2:   result  $\leftarrow$  strict
3:   for all  $a_t^i \in A_D$  do
4:     if not enableSubset  $\vee ((k + t + i) \bmod s_\phi < s_\lambda)$  then
5:       if  $d(p_k, a_t^i) \geq r + \delta$  then
6:         return filtered
7:       else if  $d(p_k, a_t^i) \geq r$  then
8:         result  $\leftarrow$  relaxed
9:       end if
10:    end if
11:  end for
12:  for all  $a_t^i \in A_I$  do
13:    if not enableSubset  $\vee ((k + t + i) \bmod s_\phi < s_\lambda)$  then
14:      if  $d(p_k, a_t^i) < r - \delta \vee d(p_k, a_t^i) \geq 2r + \delta$  then
15:        return filtered
16:      else if  $d(p_k, a_t^i) < r \vee d(p_k, a_t^i) \geq 2r$  then
17:        result  $\leftarrow$  relaxed
18:      end if
19:    end if
20:  end for
21:  return result
22: end function

```

LISTING 4.2: Particle filter condition function.

room for interpretation, Listing 4.1 gives a more detailed description, following the actual code given in the simulator released by the authors, which contains additional detail, such as a relaxed acceptance criterium for particles during the filter step.

The "attempts" parameter determines how many iterations and thereby time should be spent on trying to find a set of particles fulfilling the range MCL criteria while resampling. When running the algorithm for the first time, the "attempts" parameter is given as 10000, while subsequent runs use an "attempts" value of 200. Additionally, during the more thorough initialization run, the while loop is run through first fully discarding relaxed samples and then once more, if necessary, keeping them. For the sake of simplicity, this additional behavior for the initialization is not described in Listing 4.1.

The original MCL algorithm uses a MEETCONDITION function (shown in Listing 4.2) without the additional particle subsetting (see Section 4.3.4) introduced for RESA-MCL. To avoid having two slightly different copies of the pseudocode, we introduce the "enableSubset" parameter to MCL and MEETCONDITION. For the original MCL approach the "enableSubset" parameter is always "false" in both MCL and MEETCONDITION. The MEETCONDITION function determines whether a given sample passes the filter criterium fully, only in a relaxed manner or not at all. If enough samples are found to fully fulfill the strict range criterium, the relaxed condition is not used. Otherwise, the range is extended by  $\delta$  meters and particles fulfilling this relaxed condition

```

1: procedure SA-MCL(attempts)
2:   if  $A_D = \emptyset \wedge A_I = \emptyset$  then
3:     DEADRECKONING
4:   else
5:     MCL(attempts, enableSubset=false)
6:   end if
7: end procedure

```

LISTING 4.3: SA-MCL algorithm.

are accepted as well.

For a given point  $p$  and radius  $r$ , the function  $\text{RESAMPLEINRADIUS}(p, r)$  returns a random, new point around that point  $p$  within a radius of  $r$ , but still within the bounds of the experimental area. Given a set  $S$  and  $N \in \mathbb{N}$ , the  $\text{CHOOSE}(S, N)$  function randomly selects at most  $N$  elements from a set  $S$ , but never returns more than  $|S|$  elements.

### 4.3.3 SA-MCL

The main addition to MCL contributed by SA-MCL is that dead reckoning is used to update node locations when no anchor nodes are within communication range. This allows it to bridge the time between encountering anchor nodes with enhanced localization accuracy compared to the original approach. However, since this process is only run when no anchor nodes are within range, in denser networks its performance is not improved over that of MCL.

As shown in Listing 4.3, in SA-MCL, particle positions are updated according to the direction of the node as measured by the IMU sensor installed on it. The `DEADRECKONING` procedure described in Listing 4.4 performs these updates to the particle positions. This procedure is run instead of the regular particle resampling and filtering if both  $A_D$  and  $A_I$  are empty sets.

In a real-world test bed implementation, SA-MCL uses low-cost MPU-9150 9-axis IMU sensors with low power consumption [12]. The authors show that the use of this type of sensor is feasible in WSN localization approaches, both from a cost (monetary and power) as well as from an accuracy perspective.

### 4.3.4 RESA-MCL

RESA-MCL introduces three modifications to the SA-MCL algorithm, making it more accurate and improving its resilience to adversarial network conditions. Each of the improvements by itself is both effective and can be implemented efficiently even on low powered hardware such as 8 bit microcontroller-based IRIS sensor nodes.

```

1: procedure DEADRECKONING
2:    $L_t \leftarrow \{\}$ 
3:    $\Delta x \leftarrow \text{GETMOVEMENTXFROMSENSORS}$ 
4:    $\Delta y \leftarrow \text{GETMOVEMENTYFROMSENSORS}$ 
5:    $\Delta \text{pos} \leftarrow (\Delta x, \Delta y)$ 
6:   for all  $l_{t-1} \in L_{t-1}$  do
7:      $l_{t-1}.x \leftarrow l_{t-1}.x + \Delta x$ 
8:      $l_{t-1}.y \leftarrow l_{t-1}.y + \Delta y$ 
9:   end for
10: end procedure

```

LISTING 4.4: Particle dead reckoning.

### Motion-based particle updates

As detailed in Section 4.3.3, SA-MCL leverages dead reckoning to update particle positions when no anchor nodes are within communication range. RESA-MCL goes one step further and also applies these motion-based particle updates even when anchor nodes are within range. In RESA-MCL, the DEADRECKONING procedure is executed before the loop of prediction and filtering operations, leading to two differences. Firstly, RESA-MCL always updates its position estimate according to the sensed motion data. Secondly, if no anchor nodes are within range, both 1-hop or 2-hop, particle resampling is still performed after the motion-based position updates.

### Particle subsetting

To prevent a single malicious node from completely throwing off the position estimate of a node by providing misleading positional information, in RESA-MCL anchor node information is only applied to a subset of particles.

The particle subsetting is implemented in the form of an additional condition that allows skipping the MCL range criterium check with respect to a certain anchor node. In the original MCL, a particle that is not within the range of an anchor node that was heard by the unknown node is filtered. With RESA-MCL's particle subsetting, the particle may instead be kept, even if it is not within the radio range of that specific anchor node. More specifically, a particle  $p_k$  is subject to filtering at time  $t$  with respect to anchor  $a_t^j$ , only if  $k + t + j \bmod s_\phi < s_\lambda$ . Here  $k + t + j \bmod s_\phi$  functions in a hash-like manner to pseudo-randomly select different particles at each time step and for each anchor node. If this condition is not true, the given particle  $p_k$  is not affected by the anchor  $a_t^j$ 's positional information at the given time step.

Figure 4.1 shows an example at time  $t = 5$  with eight particles  $p_k, k \in \{1, 2, \dots, 8\}$ , two anchor nodes  $a_t^j, j \in \{1, 2\}$  and an unknown node  $U$ . The particles represent the position estimate of the unknown node. For each particle, the MCL range condition is checked with respect to both anchor nodes. Particles are color-coded according to whether they are kept (blue), filtered (red) due to not fulfilling the MCL range criterium. Green means that particle subsetting prevented them from

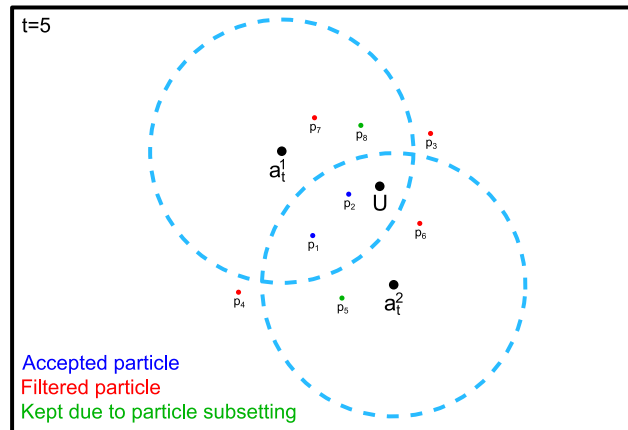


Figure 4.1: Example illustrating the particle subsetting process (S.4.3.4) with two anchors, one unknown node  $U$  and  $k \in \{1, 2, \dots, 8\}$ ,  $t = 5$ ,  $j \in \{1, 2\}$ .

being filtered despite the fact that they would fail an MCL range condition check. Particles  $p_1$  and  $p_2$  fall into the intersection of  $a_t^1$ 's and  $a_t^2$ 's radio ranges and are therefore kept. The filtered particles  $p_3, p_4, p_6, p_7$  are removed, because they fail the range MCL criterium;  $p_3, p_4$  are outside of both radio ranges,  $p_6$  is not inside the radio range of  $a_t^1$  and  $p_7$  is not inside the radio range of  $a_t^2$ .

Finally,  $p_5$  and  $p_8$  are kept only because of the particle subsetting. Specifically,  $p_5$  fulfills the range condition for  $a_t^2$ , but would be filtered due to failing it with respect to  $a_t^1$ . This is because  $5 + 5 + 1 \bmod 4 = 3$ , which is not less than  $s_\lambda = 3$ . Since the result of the modulo operation is not less than the threshold value  $s_\lambda$ , the range criterion is not checked for  $p_5$  with respect to  $a_t^1$  and the particle is not filtered out. In the case of  $p_8$ , it fulfills the range criterium for  $a_t^1$ , but it would be filtered due to failing it with respect to  $a_t^2$ . However, for this particle holds that  $8 + 5 + 2 \bmod 4 = 3$ , which is again not less than the threshold value  $s_\lambda$ . Therefore the particle  $p_8$  is kept because the range criterium is ignored.

The particle subsetting functionality is added in RESA-MCL's MEETCONDITION function, which is shown in Listing 4.2. Its general operation matches that of MCL's MEETCONDITION, other than the addition of the particle subsetting mechanism through the additional conditions on lines 4 and 13. The additional "enableSubset" parameter allows disabling this new functionality and makes the given function equivalent to the original MCL MEETCONDITION function.

Particle subsetting can prevent potential malicious nodes in the network from strongly affecting location estimates, but it also leads to a lower rate of information use in the case that all provided information is legitimate. However, the use of dead reckoning already improves the accuracy of the particle filter-based location estimation significantly, making up for the lower rate of positional correction through anchor node information.

```

1: function CHECKANCHOR( $a^i, r_f$ )
2:   if no  $a_{last}^i$  : return true
3:   return  $d((a_t^i - a_{last}^i), \Delta pos) < r \cdot r_f$ 
4: end function

```

LISTING 4.5: Anchor position plausibility check function.

```

1: procedure UPDATEPOINTS( $A_{bad}, A_{good}$ )
2:   for all  $a^i \in A_{bad}$  do
3:      $P_i \leftarrow \begin{cases} 20 & \text{if } P_i < 20 \\ P_i + 5 \end{cases}$ 
4:   end for
5:   for all  $a^i \in A_{good}$  do
6:      $P_i \leftarrow \begin{cases} 0 & \text{if } P_i < 1 \\ P_i - 1 \end{cases}$ 
7:   end for
8: end procedure

```

LISTING 4.6: Anchor distrust point update function.

### Anchor position plausibility check

RESA-MCL nodes also apply a plausibility check to received positional information, using the CHECKANCHOR function (Listing 4.5). When receiving position information from an anchor node, from which previously position information has already been received, a movement vector is calculated from the difference in positions ( $a_t^i - a_{last}^i$ ) and compared with the movement of the node ( $\Delta pos$ ) trying to localize itself according to its IMU sensed movement data. The difference between both movement vectors is then calculated and compared to the radio range multiplied by a factor  $r_f$ , which relaxes the condition to allow for inaccuracies in radio range and motion detection. The idea is that, if the anchor node and unknown node can hear each other before and after moving, and have a radio range of  $r$ , then  $r_f$  is chosen, such that they cannot have moved by a distance greater than  $r \cdot r_f$ . Specifically, the factor  $r_f$  is chosen as  $r_f = 2 \cdot hops + 0.5$ , where the hop count  $hops$  determines the maximum distance between nodes and both the factor of 2 and the additive term of 0.5 are a safety margins to reduce false positives with respect to the plausibility check. Therefore, in the 1-hop case,  $r_f = 2.5$ , which corresponds to the diameter of the circles representing the radio ranges around nodes, plus the safety margin of 0.5. As the total distance between nodes may be doubled in the 2-hop case,  $r_f$  is chosen as 4.5 following the same formula and reasoning. In the 2-hop case, the maximum distance between anchor and unknown node can be twice that of the 1-hop case. Due to that,  $r_f$  is also chosen to be higher to make up for the increased variance in movement and range introduced by the retransmission of anchor information.

If implausible information is detected, the anchor node is removed from the sets of anchor nodes used in MEETCONDITION and added to a list of untrustworthy anchor nodes. The UPDATEPOINTS

```

1: procedure RESA-MCL(attempts)
2:   DEADRECKONING
3:    $A'_D \leftarrow \{a^i | a^i \in A_D \text{ if } \text{CHECKANCHOR}((a^i, r_D))\}$ 
4:    $A'_I \leftarrow \{a^i | a^i \in A_I \text{ if } \text{CHECKANCHOR}((a^i, r_I))\}$ 
5:   UPDATEPOINTS( $((A_D \setminus A'_D) \cup (A_I \setminus A'_I), A'_D \cup A'_I)$ )
6:    $A_D \leftarrow \{a_i | a_i \in A'_D \wedge P_i = 0\}$ 
7:    $A_I \leftarrow \{a_i | a_i \in A'_I \wedge P_i = 0\}$ 
8:   MCL(attempts, enableSubset=true)
9: end procedure

```

LISTING 4.7: RESA-MCL algorithm.

(Listing 4.6) procedure assigns distrust points to such anchor nodes with implausible positions and reduces distrust points for anchor nodes with plausible positions. This allows trustworthy anchors, which were mistakenly classified as non-trustworthy, to recover over time and allows its location information to be used for localization, once its distrust points fall back to 0. At the same time, nodes that consistently misbehave are not be used for localization.

This functionality synergizes well with particle subsetting. If an anchor node broadcasts positional information which would greatly impact localization accuracy, this is likely to be detected the second time location information is received. At the same time, particle subsetting limits the impact of the inaccurate information received in the first time step.

### Plausibility checking of 2-hop forwarding

Applying the same plausibility check described for anchor nodes in the 2-hop case to unknown nodes is also considered, but not used in RESA-MCL. The idea of this approach is to disable all 2-hop data reception when implausible forwarded data is received following a similar points system as described for anchor plausibility checking. However, during the ablation experiments in Section 4.4.5, we find limited benefits of this mitigation strategy in our considered attack models and therefore do not employ it as a part of RESA-MCL. We plan to reevaluate this method in the future under different attack models.

### Putting everything together

The full algorithm can be described as given in Listing 4.7. First particle positions are updated according to dead reckoning, then heard anchor lists are filtered according to the positional plausibility check and their distrust points are updated. Finally, the basic MCL algorithm is run with particle subsetting enabled and anchor sets  $A_D, A_I$  containing only anchor nodes with zero distrust points.

The estimated position is the average position of the particles in  $L_t$  after running the MCL function.



## 4.4 Evaluation

In this section, we introduce our methodology for evaluating RESA-MCL. First, we introduce our experimental setup and simulation parameters. Following that, we show and discuss our experimental results with respect to baseline performance, motivate parameter choices and demonstrate RESA-MCL’s robust behavior under three different types of attacks.

### 4.4.1 Experimental setup

All experiments are done using an extended and improved version of the Java-based simulator originally developed and used for the evaluation MCL and also used in the evaluation of SA-MCL. The source code of our version of the simulator as well as simulation results are publicly available on Code Ocean<sup>1</sup>. This simulator is used to ensure the comparability of results to other MCL-based algorithms (e.g. SA-MCL), which commonly also use the same code base.

All experiments are run with 300 nodes. Initial positions are chosen randomly at the start of each run. Of these, unless otherwise specified, 10 are used as anchor nodes. The experimental area is  $500 \times 500 \text{ m}^2$  with a radio communication range of 50 m. Simulations are run for 1000 steps, with each step representing a time interval of 1 s and one iteration of the localization algorithm. Each such run is repeated 10 times with a different random seed. Nodes follow a modified random waypoint movement model, where a minimum movement speed of  $V_{\text{Min}}$  and a maximum path segment duration constraint are applied to prevent movement from degrading to low average speeds as described by Yoon et al. [120]. Specifically, movement speeds are randomly selected from the range of 10 m/s to 20 m/s and a limit of at maximum five time steps per path segment is imposed. An error of 20 % is applied to both the sensed speed and direction to include the effect of noisy sensors in the evaluation of our scheme and enable a fair comparison to SA-MCL. RSS or link quality between nodes is not considered beyond basic connectivity, which is all that is required in range-free localization schemes.

Hu et al. [110] define the anchor density as the average number of anchor nodes within a 1-hop distance to unknown nodes. Due to node mobility and random initialization, it is not possible to give a fixed anchor density with this definition. With the given experimental parameters, we measure a mean anchor density of 0.327 with a standard deviation of 0.054 over ten runs of 1000 steps each. Localization error in all figures is given as the error in meters divided by the radio range, which is the common measure for range-free approaches.

Due to the results shown in Section 4.4.3, the particle subsetting parameters are chosen as  $s_\lambda = 3, s_\phi = 4$  (equivalent to  $s_\lambda = 12, s_\phi = 16$ ) for all experiments other than those specifically varying those parameters. This mostly avoids impacting the localization performance in attack-free scenarios while still mitigating the impact of malicious nodes.

<sup>1</sup><https://codeocean.com/capsule/5799c9e7-22c4-450a-8783-a3e3eb2b5829/tree/v1>

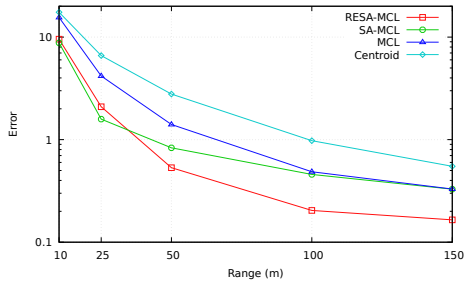


Figure 4.2: Comparison at different communication ranges without attack.

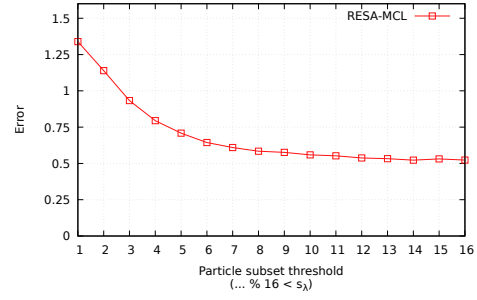


Figure 4.3: Evaluation of particle subset sizes without attacks and  $s_\phi = 16$ .

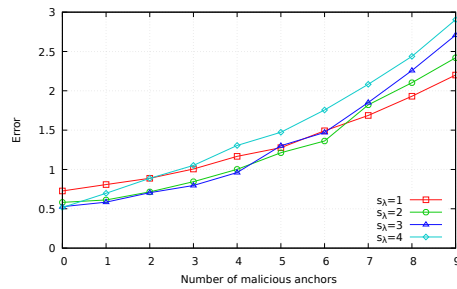


Figure 4.4: Evaluation of particle subset sizes under fixed position attack with  $s_\phi = 4$ .

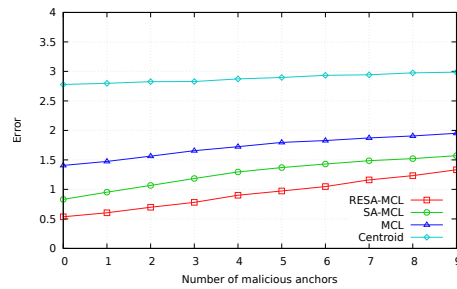


Figure 4.5: Biased position attack.

The behavior of four range-free localization algorithms (Centroid, MCL, SA-MCL, RESA-MCL) under three different attack models is evaluated. In the following, we analyze the results of our experiments.

#### 4.4.2 Baseline

Figure 4.2 shows a comparison of RESA-MCL with previous approaches in a scenario with no attacks over different radio ranges. Since only 10 anchor nodes are used for an area of  $500 \times 500 \text{ m}^2$ , in scenarios with very low communication ranges, nodes rarely encounter an anchor node. In this case, RESA-MCL and SA-MCL behave quite similarly, as both use sensor data for dead-reckoning. Since SA-MCL has no built-in detection of malfunctioning or malicious nodes, it can make full use of the very limited information it receives, letting it perform slightly better than RESA-MCL in scenarios with extremely low communication ranges and thus anchor density.

As communication range increases, the performance of SA-MCL approaches that of MCL, because nodes almost always have at least one anchor node within range, making it disregard its motion sensor data. RESA-MCL meanwhile keeps employing dead-reckoning to improve its localization accuracy even further in comparison to other approaches.

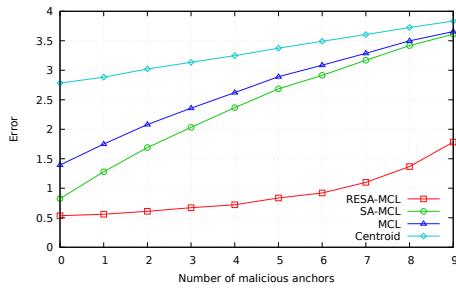


Figure 4.6: Random position attack.

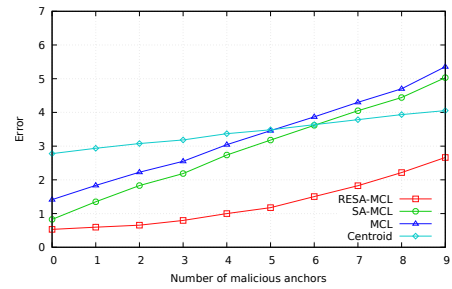


Figure 4.7: Fixed position attack.

### 4.4.3 Optimal particle subsetting

In this section we determine the optimal parameters for the particle subsetting process given in Section 4.3.4. Figure 4.3 shows the performance of RESA-MCL in a scenario without attacks, different values for  $s_\lambda$  (threshold) with  $s_\phi = 16$  (modulus). The case of  $s_\lambda = s_\phi = 16$  is equivalent to no particle subsetting, while  $s_\lambda = 1$  means that only 1 in 16 particles is affected by an anchor node. It can be seen that for  $s_\lambda < 8$ , the localization error sharply increases with declining  $s_\lambda$ , while above  $s_\lambda = 12$  the localization performance is barely affected by particle subsetting.

Figure 4.4 shows that more restrictive particle subsetting makes RESA-MCL more resilient against attacks. The fixed position attack, which is shown to be the most effective attack in Section 4.4.4, is chosen for this evaluation. Due to the results from Figure 4.3 and to increase the legibility of the figure, we divide  $s_\lambda$  and  $s_\phi$  by 4 and show results for  $s_\lambda \in \{1, 2, 3, 4\}$ , representing the most relevant tradeoff points. As per Section 4.4.2, the least subsetting performs the best in the scenario with no attacks, but  $s_\lambda = 3$  performs almost equally as well. In scenarios with 10% to 40% malicious anchor nodes,  $s_\lambda = 3$  performs the best with more aggressive subsetting being more effective in scenarios where half or more anchor nodes are malicious. As a network where the majority of nodes is malicious seems less likely, we decide on  $s_\lambda = 3$  as a reasonable trade-off between robustness and localization accuracy in networks without attacks. Therefore,  $s_\lambda = 3$  is used for all further experiments.

### 4.4.4 Performance under attacks

In the following, we present results showing the performance of RESA-MCL under three different types of attack models.

#### Biased position attack

In the biased position attack, malicious anchor nodes send their true locations, with an offset of (50, 50) (in meters) added to it. This type of attack is harder to detect for RESA-MCL, because the movement vectors of anchor nodes match their true movement, thereby usually satisfying the plausibility check. At the same time, since only an offset is added on top of the real position, the

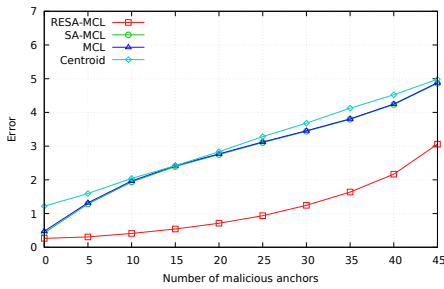


Figure 4.8: Fixed position attack with 50 anchor nodes.

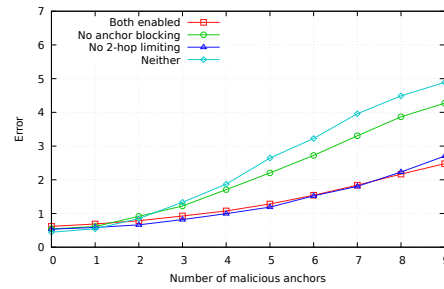


Figure 4.9: Ablation experiments.

effect of the attack is also lower than that of other attacks, as the malicious data still contains real information.

Figure 4.5 shows the performance of MCL, SA-MCL and RESA-MCL, as well as Centroid, which serves as a baseline. It can be seen that all approaches are affected by the attack, but as expected, the effect of the attack is relatively low. RESA-MCL performs the best in all cases, due to its overall superior localization accuracy capabilities granted through its use of dead reckoning.

### Random position attack

Malicious anchor nodes send out completely random, freshly chosen positions in the random position attack. The effect of this attack is stronger than that of the biased position attack as no real data is part of the malicious broadcasts.

The results in Figure 4.6 show that in this attack, the same relative ordering between approaches is maintained as with the biased position attack. However, RESA-MCL performs much better, with only a slight increase in its location estimation error up until about 50% malicious anchor nodes. The other approaches are affected much more strongly. This shows that anchor position plausibility checks and particle subsetting are effective at mitigating this attack.

### Fixed position attack

The fixed position attack is found to have the strongest effect of all three attacks. Malicious anchor nodes broadcast their position as a fixed point of (70, 70) (in meters) on the map. This pulls location estimates towards one corner of the map. Its effect is stronger than that of the random position attack because in the random position attack positions closer to the actual position of a node may be sent at random.

One obvious difference to the previous attacks, which is clearly visible in Figure 4.7, is that after 50% of anchor nodes are malicious, MCL and SA-MCL start being affected more strongly by the attack than Centroid. This is likely the case due to the low anchor density (0.327) in this scenario. When Centroid localization is not within range of any anchor nodes, which according to the

anchor density measurement is the case for 67 % of location estimates, it will assume a location in the center of the map, which is an averagely bad estimate for all points of the map. The other approaches are stateful and their current estimate is influenced by past information received from anchor nodes. With a majority of anchor nodes being malicious, these estimates will be negatively affected at nearly all times, rather than just in 32.7 % of the time as is the case with Centroid.

RESA-MCL’s robustness features can once again be seen to be effective in this scenario, with its error rate remaining way below Centroid even at 90 % of malicious anchor nodes. Up until 30 % of anchor nodes being malicious, the increase in localization error is very small.

Figure 4.8 shows the same type of results for a scenario with 300 nodes in total, of which 50 nodes are chosen as anchor nodes, leading to a much higher anchor node density. As most unknown nodes are in the range of at least one anchor node almost all the time, SA-MCL and MCL fall onto one line here, because SA-MCL will not perform its dead reckoning while within range of an anchor node. RESA-MCL outperforms all other approaches despite being optimized for low anchor node densities.

#### 4.4.5 Ablation experiments

Taking inspiration from the field of machine learning, we perform ablation experiments to evaluate the performance of RESA-MCL with certain components disabled. This experiment includes the 2-hop plausibility check described in Section 4.3.4 as one component, and the anchor position plausibility check from Section 4.3.4 as the second component. Particle subsetting is not included as results from it being disabled can also be seen in Section 4.4.3.

Figure 4.9 shows the results of disabling either or both of the components under a fixed position attack in comparison with full RESA-MCL. It can be seen that 2-hop plausibility checks only provide a benefit with 70 % or more malicious anchor nodes and otherwise worsen performance. For this reason, we do not include this component in RESA-MCL, although it may be useful under specific circumstances. Blocking malicious anchor nodes however, is shown to be highly beneficial under attack.

Approach	Anchor density	Error
RESA-MCL	0.33	0.54
	1.63	0.26
MCL-DE	0.5	1.0
	1.5	0.5

Table 4.2: Error comparison between RESA-MCL and MCL-DE without attacks.

#### 4.4.6 Comparison under anchor density

In the earlier parts of this section, we have thoroughly compared the localization accuracy of Centroid, MCL, SA-MCL and RESA-MCL. In the following, we provide a comparison of RESA-MCL and MCL-DE [111], another recent MCL-based approach.

For RESA-MCL, we measure an anchor density of 0.327 with a standard deviation of 0.054 over ten runs of 1000 steps each in scenarios with 300 nodes, of which 10 nodes are anchor nodes. In the scenario is presented in Figure 4.8, with 300 nodes in total, of which 50 nodes are anchor nodes, we measure an average anchor density of 1.63 with a standard deviation of 0.09. As can be seen in Table 4.2, MCL-DE achieves a localization error of approximately 1.0 at an anchor density of 0.5 and an error of approximately 0.5 at an anchor density of 1.5.

Our measured anchor densities in RESA-MCL do not perfectly match up with those of MCL-DE. However, with an error of 0.54 at anchor density 0.33, RESA-MCL achieves an error comparable to that of MCL-DE at anchor density 1.5 and 45 % lower than the error of 1.0 achieved by MCL-DE at the closer anchor density value of 0.5. Similarly, with an error of 0.26 at an anchor density of 1.63, RESA-MCL's error is 48 % lower than the error of 0.5 at MCL-DE's comparable anchor density of 1.5. In both cases, RESA-MCL achieves significantly lower localization errors (up to 48 %) at comparable anchor densities or, conversely, requires significantly less anchor nodes to achieve a given localization accuracy.

### 4.5 Conclusion and future works

In this work, we introduce RESA-MCL, a novel MCL-based, range-free, security-aware localization algorithm for WSNs and the IoT that strongly outperforms comparable approaches both in safe situations and under attack by malicious anchor nodes. Without attacks, it outperforms a recent comparable approach with 48 % lower localization error at similar anchor densities. RESA-MCL employs three techniques to both enhance general localization accuracy and robustness against malicious anchor nodes. Localization accuracy is enhanced both in very sparse networks with low numbers of anchor nodes and in very dense networks with high numbers of anchor nodes.

Additional contributions include the introduction of three different attack models against localization approaches in WSNs and the IoT, based on the assumption of malicious or malfunctioning anchor nodes. We evaluate the previous approaches Centroid, MCL and SA-MCL under these attack models and find them to be strongly affected by the attacks. In contrast, we demonstrate RESA-MCL's resilience against attacks under all three attack models and show that its localization error only increases slightly even with 30 % malicious anchor nodes under the most effective "fixed position" attack model.

Furthermore, we introduce the idea of 2-hop plausibility checking, which may increase resilience against malicious unknown nodes and provide a highly detailed reformalization of the original

MCL approach through pseudo-code. This reformalization includes implementation details previously found only in the original author's simulation. To facilitate easier future investigations of MCL-based approaches, we also publish an optimized version of the simulator with performance optimizations and bug fixes.

To ease future works comparing these approaches, we optimize and improve the original simulation software provided by Hu et al. [110] and Hartung et al. [12] and make it available on CodeOcean (see Section 4.4).

In the future, we plan to investigate further attack models, such as mixed attack strategies, malicious unknown nodes, as well as the effectiveness of refined versions of the 2-hop plausibility check functionality. We also plan to investigate ways of increasing the effectiveness of implausible location data checking while reducing false positives. Furthermore, we also plan to evaluate our approach in a real-world testbed, to validate simulation results. Finally, making the approach topology-aware, allowing it to exclude impassable terrain from particle locations is another avenue of enhancing localization accuracy that we plan to explore.





## Chapter 5

# Conclusions and future works

In the previous three chapters, **RQ1** through **RQ3** from Section 1.3 have been addressed.

Unchained, presented in Chapter 2, addresses **RQ1** by describing a decentralized scheme for the creation of cryptographically secure identities that monetarily disincentivizes the execution of Sybil attacks. This is achieved by outsourcing a PoW puzzle to existing Blockchain networks, instead of having the user, who is creating the identity, perform the PoW locally. Honesty is promoted by ensuring that attempting to circumvent the scheme would incur opportunity costs that are at least as high as the costs incurred by creating identities in an honest manner. Verification of identities can be performed offline without actually accessing the Blockchain network or even the Internet. Chapter 3 presents the Rechained protocol, an extension of Unchained, which refines certain aspects and addresses previously still open issues, such as the revocation of identities and eclipse attacks, which are mainly relevant in distributed P2P networks.

Chapter 4 introduces RESA-MCL, a range-free localization scheme for WSNs. RESA-MCL addresses **RQ2** by improving localization accuracy, both in scenarios with low and high anchor densities. Without attacks, it achieves up to 48 % lower localization error at comparable anchor densities to other comparable schemes. **RQ3** is addressed by the introduction of a novel particle subsetting mechanism, which reduces the impact of successfully disseminated incorrect location data, as well as a new anchor node plausibility check mechanism, which excludes malfunctioning or malicious anchor nodes from the network.

Three different attack models based on malicious anchor nodes are defined and RESA-MCL's performance under these attacks with up to 90 % malicious anchor nodes is evaluated and compared to previous approaches. In all scenarios, RESA-MCL is found to strongly outperform the previous approaches under all attacks. Ablation studies are used to evaluate the actual benefit of each new technique employed in RESA-MCL. Additionally, an optimized version of the simulator used for the experiments is provided to allow reproduction of the results in this thesis as well as ease future experiments in this field. The simulator includes implementations of the Centroid [121],

MCL [110], SA-MCL [12] and RESA-MCL algorithms.

A combination of the approaches from the previous chapters provides a framework for secure localization in WSNs.

Both the works addressing **RQ1** and RESA-MCL also open up numerous paths for future works.

In the future, investigating the feasibility of implementing the Unchained and Re chained approaches on Blockchains using consensus mechanisms other than PoW would be beneficial as PoW is often considered wasteful or environmentally harmful. Furthermore, doing studies of real-world deployments of these approaches, for example in a WSN testbed, should help demonstrate their feasibility further. Another possible extension is to employ Blockchain analysis techniques to augment the approach with a way of identifying possible Sybil attacks. Finally, formally specifying and implementing the different update mechanisms should aid in making the approaches deployable in the real world.

With respect to RESA-MCL, in the future, it is planned to design additional attack models, covering an even wider range of scenarios, also including malicious unknown nodes. Another avenue of future research is investigating whether the location data plausibility check mechanism can be improved further for enhanced accuracy and lower false-positive rates, improving the approach's performance in scenarios without attacks. Real-world testbed implementations are often very helpful in uncovering issues or further clarifying various properties of localization approaches. SA-MCL [12] has been evaluated in a real-world testbed. Based on this, it would be beneficial to do the same for RESA-MCL. Exploiting properties of the terrain and making the approach topology-aware may also further improve localization accuracy and allow for novel ways of detecting implausible data sent by malfunctioning or malicious nodes.

Furthermore, it would also be of interest to analyze MCL-based approaches with respect to certain theoretical properties such as upper and lower bounds of error, as well as average error rates theoretically achievable under given anchor densities. Determining practical deployment costs of nodes in various configurations (e.g. unknown nodes compared to anchor nodes with GPS and correspondingly bigger batteries) would also be beneficial.

# Bibliography

- [1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," <https://bitcoin.org/bitcoin.pdf>, p. 28, 2008, (Accessed 2022-04-26).
- [2] Bitcoincharts, "Bitcoincharts API, Price data (MtGox, BTC-e, BitStamp, Coinbase)," (Accessed 2022-04-26). [Online]. Available: <https://api.bitcoincharts.com/v1/csv/>
- [3] Blockchain.info, "Bitcoin Blockchain, Difficulty," (Accessed 2022-04-26). [Online]. Available: <https://api.blockchain.info/charts/difficulty?format=csv>
- [4] Etherscan, "Ethereum Charts and Statistics," <https://etherscan.io/charts>, 2017, (Accessed 2022-04-26).
- [5] B. Leiding, "The M2X Economy – Concepts for Business Interactions, Transactions and Collaborations Among Autonomous Smart Devices," PhD Thesis, University of Göttingen, Göttingen, Germany, 2020.
- [6] W.-Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. LaHusen, "Air-Dropped Sensor Network for Real-Time High-Fidelity Volcano Monitoring," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 305–318.
- [7] P. Sommer, J. Liu, K. Zhao, B. Kusy, R. Jurdak, A. McKeown, and D. Westcott, "Information Bang for the Energy Buck: Towards Energy- and Mobility-Aware Tracking," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '16. USA: Junction Publishing, 2016, p. 193–204.
- [8] M. Liu, K. Yang, N. Zhao, Y. Chen, H. Song, and F. Gong, "Intelligent Signal Classification in Industrial Distributed Wireless Sensor Networks Based Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4946–4956, 2021.
- [9] Y. I. Wu, H. Wang, and X. Zheng, "WSN Localization Using RSS in Three-Dimensional Apace—A Geometric Method With Closed-Form Solution," *IEEE Sensors Journal*, vol. 16, no. 11, pp. 4397–4404, 2016.

- [10] N. Xie, Y. Chen, Z. Li, and D. O. Wu, "Lightweight Secure Localization Approach in Wireless Sensor Networks," *IEEE Transactions on Communications*, vol. 69, no. 10, pp. 6879–6893, 2021.
- [11] J. Yin, Q. Wan, S. Yang, and K. Ho, "A Simple and Accurate TDOA-AOA Localization Method Using Two Stations," *IEEE Signal Processing Letters*, vol. 23, no. 1, pp. 144–148, 2015.
- [12] S. Hartung, A. Bochem, A. Zdziarstek, and D. Hogrefe, "Applied Sensor-Assisted Monte Carlo Localization for Mobile Wireless Sensor Networks," in *International Conference on Embedded Wireless Systems and Networks (EWSN), 2016*, Graz, Austria, February 2016.
- [13] W. Ding, S. Chang, and J. Li, "A Novel Weighted Localization Method in Wireless Sensor Networks Based on Hybrid RSS/AoA Measurements," *IEEE Access*, vol. 9, pp. 150 677–150 685, 2021.
- [14] J. R. Douceur, "The Sybil Attack," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 251–260.
- [15] Y. Yuan, L. Huo, Z. Wang, and D. Hogrefe, "Secure APIT Localization Scheme Against Sybil Attacks in Distributed Wireless Sensor Networks," *IEEE Access*, vol. 6, pp. 27 629–27 636, 2018.
- [16] O. Alfandi, A. Bochem, A. Kellner, and D. Hogrefe, "Simple secure PKI-based scheme for wireless sensor networks," *Proceedings of the 2011 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2011*, pp. 359–364, 2011.
- [17] B. Prünster, D. Ziegler, C. Kollmann, and B. Suzic, "A Holistic Approach Towards Peer-to-Peer Security and Why Proof of Work Won't Do," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2018, pp. 122–138.
- [18] O. Alfandi, A. Bochem, A. Kellner, C. Göge, and D. Hogrefe, "Secure and authenticated data communication in wireless sensor networks," *Sensors (Switzerland)*, vol. 15, no. 8, pp. 19 560–19 582, 2015.
- [19] A. Bochem, B. Leiding, and D. Hogrefe, "Unchained Identities: Putting a Price on Sybil Nodes in Mobile Ad hoc Networks," in *14th EAI International Conference on Security and Privacy in Communication Networks*. Springer, 2018, pp. 358–374, (Accessed 2022-04-26). [Online]. Available: [http://dx.doi.org/10.1007/978-3-030-01701-9\\_20](http://dx.doi.org/10.1007/978-3-030-01701-9_20)
- [20] A. Bochem and B. Leiding, "Rechained: Sybil-Resistant Distributed Identities for the Internet of Things and Mobile Ad Hoc Networks," *Sensors*, vol. 21, no. 9, 2021, (Accessed 2022-04-26). [Online]. Available: <http://dx.doi.org/10.3390/s21093257>
- [21] A. Bochem and H. Zhang, "Robustness Enhanced Sensor Assisted Monte Carlo Localization for Wireless Sensor Networks and the Internet of Things," *IEEE Access*, vol. 10, pp. 33 408–33 420, 2022, (Accessed 2022-04-26). [Online]. Available: <http://dx.doi.org/10.1109/ACCESS.2022.3162288>

- [22] R. van der Meulen, "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016," <https://www.gartner.com/newsroom/id/3598917>, 2017, (Accessed 2022-04-26).
- [23] A. Nordrum, "Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated," <https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated>, 2016, (Accessed 2022-04-26).
- [24] H. A. Horst and D. Miller, *Digital Anthropology*. A&C Black, 2013.
- [25] K. Su, J. Li, and H. Fu, "Smart City and the Applications," in *Electronics, Communications and Control (ICECC), 2011 International Conference on*. IEEE, 2011, pp. 1028–1031.
- [26] J. Macker, "Mobile Ad-Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, RFC 2501," 1999.
- [27] N. Raza, M. U. Aftab, M. Q. Akbar, O. Ashraf, and M. Irfan, "Mobile Ad-Hoc Networks Applications and Its Challenges," 2016.
- [28] U. S. R. K. Dhamodharan and R. Vayanaperumal, "Detecting and Preventing Sybil Attacks in Wireless Sensor Networks Using Message Authentication and Passing Method," *The Scientific World Journal*, vol. 2015, 2015.
- [29] R. John, J. P. Cherian, and J. J. Kizhakkethottam, "A Survey of Techniques to Prevent Sybil Attacks," in *Soft-Computing and Networks Security (ICSNS), 2015 International Conference on*. IEEE, 2015, pp. 1–6.
- [30] S. Abbas, M. Merabti, D. Llewellyn-Jones, and K. Kifayat, "Lightweight Sybil Attack Detection in MANETs," *IEEE systems journal*, vol. 7, no. 2, pp. 236–248, 2013.
- [31] A. Tangpong, G. Kesidis, H.-y. Hsu, and A. Hurson, "Robust Sybil Detection for MANETs," in *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th International Conference on*. IEEE, 2009, pp. 1–6.
- [32] G. Wood, "Ethereum: A Secure Decentralized Generalised Transaction Ledger," <http://gavwood.com/paper.pdf>, 2014, (Accessed 2022-04-26).
- [33] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: Defending Against Sybil Attacks Via Social Networks," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 267–278.
- [34] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A Near-Optimal Social Network Defense Against Sybil Attacks," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 3–17.

- [35] B. Xiao, B. Yu, and C. Gao, "Detection and Localization of Sybil Nodes in VANETs," in *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*. ACM, 2006, pp. 1–8.
- [36] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil Attack in Sensor Networks: Analysis & Defenses," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 259–268.
- [37] P. Dai, N. Mahi, J. Earls, and A. Norta, "Smart-Contract Value-Transfer Protocols on a Distributed Mobile Application Platform," [https://www.researchgate.net/publication/314190216\\_Smart-Contract\\_Value-Transfer\\_Protocols\\_on\\_a\\_Distributed\\_Mobile\\_Application\\_Platform](https://www.researchgate.net/publication/314190216_Smart-Contract_Value-Transfer_Protocols_on_a_Distributed_Mobile_Application_Platform), 2017, (Accessed 2022-04-26).
- [38] S. Popov, "The Tangle - Version 1.3," [https://web.archive.org/web/20171226231533/https://iota.org/IOTA\\_Whitepaper.pdf](https://web.archive.org/web/20171226231533/https://iota.org/IOTA_Whitepaper.pdf), 2017, (Accessed 2022-04-26).
- [39] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
- [40] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT," in *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Springer, 2017, pp. 523–533.
- [41] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-Managed and Blockchain-Based Vehicular Ad-Hoc Networks," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, 2016, pp. 137–140.
- [42] O. Bussmann, "The Future of Finance: FinTech, Tech Disruption, and Orchestrating Innovation," in *Equity Markets in Transition*. Springer, 2017, pp. 473–486.
- [43] Q. K. Nguyen, "Blockchain - A Financial Technology for Future Sustainable Development," in *Green Technology and Sustainable Development (GTSD), International Conference on*. IEEE, 2016, pp. 51–54.
- [44] B. Leiding, C. H. Cap, T. Mundt, and S. Rashidibajgan, "Authcoin: Validation and Authentication in Decentralized Networks," in *The 10th Mediterranean Conference on Information Systems - MCIS 2016*, Cyprus, CY, September 2016.
- [45] P. McCorry, S. F. Shahandashti, D. Clarke, and F. Hao, "Authenticated key exchange over bitcoin," in *International Conference on Research in Security Standardisation*. Springer, 2015, pp. 3–20.
- [46] Bitcoin Project, "Bitcoin Developer Guide," <https://web.archive.org/web/20171221092534/https://bitcoin.org/en/developer-guide#proof-of-work>, 2017, (Accessed 2022-04-26).

- [47] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies," in *2015 IEEE Symposium on Security and Privacy*, May 2015, pp. 104–121.
- [48] P. Todd, "BIP 65 - OP\_CHECKLOCKTIMEVERIFY," 2014, (Accessed 2022-04-26). [Online]. Available: <https://github.com/bitcoin/bips/blob/6295c1a095a1fa33f38d334227fa4222d8e0a523/bip-0009.mediawiki>
- [49] "Mining - Bitcoin Wiki," (Accessed 2022-04-26). [Online]. Available: <https://en.bitcoin.it/w/index.php?title=Mining&oldid=64115#Reward>
- [50] "Transaction - Bitcoin Wiki," (Accessed 2022-04-26). [Online]. Available: <https://en.bitcoin.it/w/index.php?title=Transaction&oldid=63712>
- [51] Ethereum Team, "Byzantium HF Announcement," <https://blog.ethereum.org/2017/10/12/byzantium-hf-announcement/>, 2017, (Accessed 2022-04-26).
- [52] P. A. Barro, M. Zennaro, and E. Pietrosemoli, "TLTN–The Local Things Network: On the Design of a LoRaWAN Gateway with Autonomous Servers for Disconnected Communities," in *2019 Wireless Days (WD)*. IEEE, 2019, pp. 1–4.
- [53] S. Laso, D. Flores-Martín, J. L. Herrera, C. Canal, J. M. Murillo, and J. Berrocal, "Providing Support to IoT Devices Deployed in Disconnected Rural Environment," in *International Workshop on Gerontechnology*. Springer, 2019, pp. 140–150.
- [54] H. Rajadurai and U. D. Gandhi, "Fuzzy Based Collaborative Verification System for Sybil Attack Detection in MANET," *Wireless Personal Communications*, vol. 110, no. 4, pp. 2179–2193, 2020.
- [55] K. Jensen, "Coloured Petri Nets," in *Discrete Event Systems: A New Challenge for Intelligent Control Systems, IEE Colloquium on*. IET, 1993, pp. 5–1.
- [56] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3-4, pp. 213–254, 2007.
- [57] Y. Permpoontanalarp and A. Changkhanak, "Security Analysis of the TMN Protocol by Using Coloured Petri Nets: On-the-fly Trace Generation Method and Homomorphic Property," in *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2011, pp. 63–68.
- [58] R. Drummond, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, and J. Holt, "Decentralized Identifiers (DIDs) v1.0 – Core Architecture, Data Model, and Representations," <https://w3c.github.io/did-core/>, 2020, (Accessed 2022-04-26).

- [59] A. Back, "Hashcash - A Denial of Service Counter-Measure," <http://www.hashcash.org/papers/hashcash.pdf>, 2002, (Accessed 2022-04-26).
- [60] Y. Yao, B. Xiao, G. Wu, X. Liu, Z. Yu, K. Zhang, and X. Zhou, "Multi-Channel based Sybil Attack Detection in Vehicular Ad Hoc Networks Using RSSI," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 362–375, 2018.
- [61] M. Jamshidi, E. Zangeneh, M. Esnaashari, and M. R. Meybodi, "A Lightweight Algorithm for Detecting Mobile Sybil Nodes in Mobile Wireless Sensor Networks," *Computers & Electrical Engineering*, vol. 64, pp. 220–232, 2017.
- [62] M. A. Jan, P. Nanda, X. He, and R. P. Liu, "A Sybil Attack Detection Scheme for a Forest Wildfire Monitoring Application," *Future Generation Computer Systems*, vol. 80, pp. 613–626, 2018.
- [63] U. Asfia, V. Kamuni, S. Sutavani, A. Sheikh, S. Wagh, and N. Singh, "A Blockchain Construct for Energy Trading Against Sybil Attacks," in *2019 27th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2019, pp. 422–427.
- [64] S. Huh, S. Cho, and S. Kim, "Managing IoT Devices Using Blockchain Platform," in *19th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2017, pp. 464–467.
- [65] Y. Zhang and J. Wen, "The IoT Electric Business Model: Using Blockchain Technology for the Internet of Things," *Peer-to-Peer Networking and Applications*, vol. 10, no. 4, pp. 983–994, 2017.
- [66] J. Lin, Z. Shen, A. Zhang, and Y. Chai, "Blockchain and IoT based Food Traceability for Smart Agriculture," in *Proceedings of the 3rd International Conference on Crowd Science and Engineering*, 2018, pp. 1–6.
- [67] X. Li, X. Wu, X. Pei, and Z. Yao, "Tokenization: Open Asset Protocol on Blockchain," in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2019, pp. 204–209.
- [68] A. Garba, A. Bochem, and B. Leiding, "BlockVoke – Fast, Blockchain-Based Certificate Revocation for PKIs and the Web of Trust," in *Information Security*, W. Susilo, R. H. Deng, F. Guo, Y. Li, and R. Intan, Eds. Cham: Springer International Publishing, 2020, pp. 315–333.
- [69] N. Cam-Winget, R. Housley, D. Wagner, and J. Walker, "Security Flaws in 802.11 Data Link Protocols," *Communications of the ACM*, vol. 46, no. 5, pp. 35–39, 2003.
- [70] U. Carlsen, "Cryptographic Protocol Flaws: Know Your Enemy," in *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings*. IEEE, 1994, pp. 192–200.



- [71] F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman, "Strand Spaces: Why is a Security Protocol Correct?" in *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*. IEEE, 1998, pp. 160–171.
- [72] S. Vaudenay, "Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS..." in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2002, pp. 534–545.
- [73] A. Stubblefield, J. Ioannidis, and A. D. Rubin, "A Key Recovery Attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 2, pp. 319–332, 2004.
- [74] C. Brook, "Nuclear Power Plant Disrupted by Cyber Attack," <https://threatpost.com/nuclear-power-plant-disrupted-by-cyber-attack/121216/>, 2016, (Accessed 2022-04-26).
- [75] C. A. Petri, "Kommunikation mit Automaten," PhD Thesis, Technical University of Darmstadt, 1962.
- [76] R. Milner, J. Parrow, and D. Walker, "A Calculus of Mobile Processes – I," *Information and Computation*, vol. 100, no. 1, pp. 1–40, 1992.
- [77] C. A. R. Hoare, "Communicating Sequential Processes," in *The Origin of Concurrent Programming*. Springer, 1978, pp. 413–443.
- [78] M. Abadi and A. D. Gordon, "A Calculus for Cryptographic Protocols: The Spi Calculus," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*. ACM, 1997, pp. 36–47.
- [79] F. Crazzolaro and G. Winskel, "Events in Security Protocols," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*. ACM, 2001, pp. 96–105.
- [80] S. Aly and K. Mustafa, "Protocol Verification and Analysis Using Colored Petri Nets," URL: <http://facweb.cs.depaul.edu/research/techreports/tr04-003.pdf>, 2003, (Accessed 2022-04-26).
- [81] K. Edwards, "Cryptographic Protocol Specification and Analysis Using Coloured Petri Nets and Java." Ph.D. dissertation, Queen's University Kingston, 1999.
- [82] A. Norta and L. Kutvonen, "Safeguarding Trusted eBusiness Transactions of Lifecycles for Cross-Enterprise Collaboration," Tech. Rep., 2012.
- [83] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented Requirements Analysis and Reasoning in the Tropos Methodology," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 159–171, 2005.

- [84] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia Methodology for Agent-oriented Analysis and Design," *Autonomous Agents and Multi-agent Systems*, vol. 3, no. 3, pp. 285–312, 2000.
- [85] L. Padgham and M. Winikoff, "Prometheus: A Methodology for Developing Intelligent Agents," in *International Workshop on Agent-Oriented Software Engineering*. Springer, 2002, pp. 174–185.
- [86] B. Moulin and L. Cloutier, "Soft computing," F. Aminzadeh and M. Jamshidi, Eds. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994, ch. Collaborative Work Based on Multiagent Architectures: A Methodological Perspective, pp. 261–296.
- [87] B. Moulin and M. Brassard, "A Scenario-based Design Method and an Environment for the Development of Multiagent Systems," in *Australian Workshop on Distributed Artificial Intelligence*. Springer, 1995, pp. 216–232.
- [88] S. A. DeLoach, "Analysis and Design using MaSE and agentTool," DTIC Document, Tech. Rep., 2001.
- [89] M. Mahunnah, A. Norta, L. Ma, and K. Taveter, "Heuristics for Designing and Evaluating Socio-technical Agent-Oriented Behaviour Models with Coloured Petri Nets," in *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*. IEEE, 2014, pp. 438–443.
- [90] L. Sterling and K. Taveter, *The Art of Agent-Oriented Modeling*. MIT Press, 2009.
- [91] A. M. Davis, *Software Requirements: Objects, Functions, and States*. Prentice-Hall, Inc., 1993.
- [92] IEEE Computer Society. Software Engineering Technology Committee and Institute of Electrical and Electronics Engineers, *IEEE Recommended Practice for Software Requirements Specifications*, ser. IEEE Std. Institute of Electrical and Electronics Engineers, 1994.
- [93] A. Norta, P. Grefen, and N. C. Narendra, "A Reference Architecture for Managing Dynamic Inter-Organizational Business Processes," *Data & Knowledge Engineering*, vol. 91, pp. 52–89, 2014.
- [94] J. Marshall, "Agent-Based Modelling of Emotional Goals in Digital Media Design Projects," *International Journal of People-Oriented Programming (IJPOP)*, vol. 3, no. 1, pp. 44–59, 2014.
- [95] B. Leiding and A. Norta, "Mapping Requirements Specifications Into a Formalized Blockchain-Enabled Authentication Protocol for Secured Personal Identity Assurance," in *4th International Conference on Future Data and Security Engineering - FDSE 2017*. Ho Chi Minh City, Vietnam: Springer, 2017, pp. 181–196.
- [96] B. Leiding, "Securing the Authcoin Protocol Using Security Risk-Oriented Patterns," Master's thesis, University of Göttingen, Göttingen, Germany, 2017.

- [97] Privacy Pros, "Bitcoin Transaction Fee Estimator & Calculator," <https://privacypros.io/tools/bitcoin-fee-estimator/>, 2021, (Accessed 2022-04-26).
- [98] CoinMarketCap, "Historical Data for Bitcoin," <https://coinmarketcap.com/currencies/bitcoin/historical-data/>, 2021, (Accessed 2022-04-26).
- [99] Lombrozo, Eric and Lau, Johnson and Wuille, Pieter, "BIP: 141 – Segregated Witness (Consensus layer)," <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>, 2021, (Accessed 2022-04-26).
- [100] Wuille, Pieter and Maxwell, Greg, "BIP: 173 – Base32 address format for native v0-16 witness outputs," <https://github.com/bitcoin/bips/blob/master/bip-0173.mediawiki>, 2021, (Accessed 2022-04-26).
- [101] James Ray and Vitalik Buterin et al., "Ethereum Design Rationale," (Accessed 2022-04-26). [Online]. Available: <https://github.com/ethereum/wiki/wiki/Ethereum-Design-Rationale/6e97c9cea49605264c6f4d1dc9e1939b1f89a5a3>
- [102] Mobility Open Blockchain Initiative - VID Working Group, "MOBI Vehicle Identity Standard - Version 1.0 Preview," URL: <https://dlt.mobi/wp-content/uploads/2020/04/Preview-MOBI-Vehicle-Identity-Standard-v1.0.pdf>, 2019, (Accessed 2022-04-26).
- [103] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," 2015.
- [104] A. Kanev, A. Nasteka, C. Bessonova, D. Nevmerzhitsky, A. Silaev, A. Efremov, and K. Niki-forova, "Anomaly Detection in Wireless Sensor Network of the "Smart Home" System," in *2017 20th Conference of Open Innovations Association (FRUCT)*, 2017, pp. 118–124.
- [105] N. Castell, F. R. Dauge, P. Schneider, M. Vogt, U. Lerner, B. Fishbain, D. Broday, and A. Bartonova, "Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates?" *Environment International*, vol. 99, pp. 293–302, 2017.
- [106] L. Tönisson, J. Voigtländer, M. Weger, D. Assmann, R. Käthner, B. Heinold, and A. Macke, "Knowledge Transfer with Citizen Science: Luft-Leipzig Case Study," *Sustainability*, vol. 13, no. 14, 2021.
- [107] Q. Luo, Y. Peng, J. Li, and X. Peng, "RSSI-Based Localization Through Uncertain Data Mapping for Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 16, no. 9, pp. 3155–3162, 2016.
- [108] T. Wang, H. Ding, H. Xiong, and L. Zheng, "A Compensated Multi-Anchors TOF-Based Localization Algorithm for Asynchronous Wireless Sensor Networks," *IEEE Access*, vol. 7, pp. 64 162–64 176, 2019.

- [109] T. Wang, H. Xiong, H. Ding, and L. Zheng, "TDOA-Based Joint Synchronization and Localization Algorithm for Asynchronous Wireless Sensor Networks," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 3107–3124, 2020.
- [110] L. Hu and D. Evans, "Localization for Mobile Sensor Networks," in *10<sup>th</sup> Annual International Conference on Mobile Computing and Networking (MobiCom 2004)*, Philadelphia, USA, 2004, pp. 45–57.
- [111] M. Qin and R. Zhu, "A Monte Carlo localization method based on differential evolution optimization applied into economic forecasting in mobile wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, pp. 1–9, 2018.
- [112] L. Gui, F. Xiao, Y. Zhou, F. Shu, and T. Val, "Connectivity Based DV-Hop Localization for Internet of Things," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8949–8958, 2020.
- [113] Y. Zhang, L. Cui, and S. Chai, "Energy-efficient localization for mobile sensor networks based on RSS and historical information," in *Control and Decision Conference (CCDC), 2015 27th Chinese*. IEEE, 2015, pp. 5246–5251.
- [114] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional Neural Networks Based Indoor Wi-Fi Localization Using Channel State Information," *IEEE Access*, vol. 5, pp. 18 066–18 074, 2017.
- [115] S. K. Gharghan, R. Nordin, A. M. Jawad, H. M. Jawad, and M. Ismail, "Adaptive Neural Fuzzy Inference System for Accurate Localization of Wireless Sensor Network in Outdoor and Indoor Cycling Applications," *IEEE Access*, vol. 6, pp. 38 475–38 489, 2018.
- [116] B. A. Akram, A. H. Akbar, and O. Shafiq, "HybLoc: Hybrid Indoor Wi-Fi Localization Using Soft Clustering-Based Random Decision Forest Ensembles," *IEEE Access*, vol. 6, pp. 38 251–38 272, 2018.
- [117] Z. Munadhil, S. K. Gharghan, A. H. Mutlag, A. Al-Naji, and J. Chahl, "Neural Network-Based Alzheimer's Patient Localization for Wireless Sensor Network in an Indoor Environment," *IEEE Access*, vol. 8, pp. 150 527–150 538, 2020.
- [118] X. Liu, S. Su, F. Han, Y. Liu, and Z. Pan, "A Range-Based Secure Localization Algorithm for Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 19, no. 2, pp. 785–796, 2018.
- [119] *IRIS OEM datasheet*, MEMSIC Inc., 2010, Accessed on 2022-04-26. [Online]. Available: [http://static6.arrow.com/arrowpdfconversion/8f126c9f83e22f80e2c01b513ad4960db1ea/6020-0123-02\\_a\\_iris\\_oem\\_edition-t.pdf](http://static6.arrow.com/arrowpdfconversion/8f126c9f83e22f80e2c01b513ad4960db1ea/6020-0123-02_a_iris_oem_edition-t.pdf)
- [120] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*. IEEE Societies, vol. 2, March 2003, pp. 1312–1321 vol.2.

- [121] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low-Cost Outdoor Localization for Very Small Devices," *Personal Communications, IEEE*, vol. 7, no. 5, pp. 28–34, 2000.



# Arne Bochem

Curriculum Vitae

Adress:                                 Johannisstr. 7, 37073 Göttingen  
Phone:                                     +49 551 486803  
Email:                                     arne.bochem@cs.uni-goettingen.de  
Date of birth:                             1984-01-04  
Place of birth:                             Bad Mergentheim  
Citizenship:                               German

## Education

2016–2022                                 PhD Programme in Computer Science (GAUSS PCS)  
Georg-August-Universität Göttingen  
Telematics Research Group

2011–2015                                 Applied Computer Science Programme  
Georg-August-Universität Göttingen  
Degree: Master of Science  
Thesis:  
    Efficient Localization for Mobile Wireless  
    Sensor Networks using extended Monte Carlo  
    Localization

2006–2011                                 Applied Computer Science Programme  
Georg-August-Universität Göttingen  
Degree: Bachelor of Science  
Thesis:  
    Evaluations of a Simple Secure PKI-based  
    approach for Wireless Sensor Networks

2004–2006                                 Mathematics Diploma Programme  
Georg-August-Universität Göttingen  
(Switched to Applied Computer Science.)

# Teaching experience

<b>2016–present</b>	Georg-August-Universität Göttingen Institute for Computer Science Involved courses: Informatik I (Computer Science I) (yearly) Informatik II (Computer Science II) (yearly) (Grundlagen der Praktischen Informatik) Mobile Communications (yearly)
2021–present	Proseminar Network Security and Privacy (twice a year) Emerging Technologies for the Circular Economy (yearly)
2018–2020	Introduction to Blockchain Technology (yearly) Advanced Blockchain (yearly)
2016–2020	Seminar Self-Organized Networks (twice a year)
2016–2019	Practical Course in Wireless Sensor Networks (twice a year; paused due to COVID-19)
2016–2018	Key competencies for PhD students of the Faculty of Mathematics and Computer Science (yearly)
<b>2011–2015</b>	Student teaching assistant Georg-August-Universität Göttingen Institute for Computer Science Telematics Research Group, Sensorlab Courses: Practical Course in Wireless Sensor Networks (twice a year)

# Other activities

2016–2022	Student speaker of the GAUSS PCS and PEI: PhD Programme for Computer Science PhD Programme for Environmental Informatics
2016–2019	Supported organization of the Girls' Day at the Institute of Computer Science (on hold due to COVID-19)



## Supervised student thesis (MSc)

**Formalizing and Securing the BlockVoke protocol for fast, cost-efficient and privacy respecting revocations.** Anant Sujatanagarjuna, 2021.

**Threat Analysis of the Rechained Protocol by Applying Security Risk-oriented Patterns for Security Improvements.** Mehmed Mustafa, 2021.

**An Autonomous Self-Localizing Vehicular Testbed for Wireless Sensor Network Applications.** Andreas Zdziarstek, 2019.

**Digital Investigations on Biomedical Wireless Sensor – Zephyr BioHarness 3.** Argianto Rahartomo, 2017.

## Supervised student thesis (BSc)

**Erstellung und Analyse einer Bibliothek zur Entropiesammlung für Wireless Sensor Nodes nach NIST SP800-90B.** Oliver Buse, 2016.

## Supervised student projects

**Project: Extending TLS for WSN.** Leon Tan, 2016.

**Project: Cargo tracking via RFID.** Lars Runge and Sebastian Schrage, 2016.

## Publications

A. Bochem & H. Zhang (2022). **Robustness Enhanced Sensor Assisted Monte Carlo Localization for Wireless Sensor Networks and the Internet of Things.** IEEE Access, vol. 10, pp. 33408-33420.

A. Sujatanagarjuna, A. Bochem & B. Leiding (2021). **Formalizing the Blockchain-Based BlockVoke Protocol for Fast Certificate Revocation Using Colored Petri Nets.** Information, 12(7), 277.

A. Bochem & B. Leiding (2021). **Rechained: Sybil-Resistant Distributed Identities for the Internet of Things and Mobile Ad Hoc Networks.** Sensors, 21(9), 3257.

- A. Garba, A. Bochem & B. Leiding (2020). **BlockVoke–Fast, Blockchain–Based Certificate Revocation for PKIs and the Web of Trust**. Lecture Notes in Computer Science, 315–333.
- B. Leiding, A. Bochem & L. H. Acosta (2019). **Automated Sensor–Fusion Based Emergency Rescue for Remote and Extreme Sport Activities**. In 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC) (pp. 1233–1238).
- H. Zhang, A. Bochem, X. Sun & D. Hogrefe (2018). **A Security Aware Fuzzy Enhanced Ant Colony Optimization Routing in Mobile Ad hoc Networks**. International Conference on Wireless and Mobile Computing, Networking and Communications, 2018–October.
- H. Zhang, A. Bochem, X. Sun & D. Hogrefe (2018). **A Security Aware Fuzzy Enhanced Reliable Ant Colony Optimization Routing in Vehicular Ad hoc Networks**. IEEE Intelligent Vehicles Symposium, Proceedings, 2018–June, 1071–1078.
- M. Gurabi, O. Alfandi, A. Bochem & D. Hogrefe (2018). **Hardware based Two–Factor User Authentication for the Internet of Things**. 2018 14th International Wireless Communications and Mobile Computing Conference, IWCMC 2018, 1081–1086.
- A. Bochem, B. Leiding & D. Hogrefe (2018). **Unchained identities: Putting a price on sybil nodes in mobile Ad Hoc networks**. Lecture Notes of the Institute for Computer Sciences, Social–Informatics and Telecommunications Engineering, LNICST, 254, 358–374.
- A. Bochem, H. Zhang & D. Hogrefe (2017). Poster abstract: **Streamlined anomaly detection in web requests using recurrent neural networks**. 2017 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPs 2017, 1016–1017.
- A. Bochem, K. Freeman, M. Schwarzmaier, O. Alfandi & D. Hogrefe (2016). **A privacy–preserving and power–efficient bicycle tracking scheme for theft mitigation**. IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 – Proceedings.
- S. Hartung, A. Bochem, A. Zdziarstek & D. Hogrefe (2016). **Applied Sensor–Assisted Monte Carlo Localization for Mobile Wireless Sensor Networks**. In Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (pp. 181–192). Junction Publishing.

- O. Alfandi, A. Bochem, M. Gurabi, A. Diaz, M. Mehedi & D. Hogrefe (2016). **Calculating the speed of vehicles using Wireless Sensor Networks**. Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016, 1043-1047.
- A. Bochem, A. Zdziarstek, Hartung, S. & D. Hogrefe (2016). Demo: **Dead Reckoning for Monte Carlo Localization in Low Seed Density Scenarios**. In Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (pp. 227-228). Junction Publishing.
- A. Bochem, Y. Yuan & D. Hogrefe (2016). **Tri-MCL: Synergistic Localization for Mobile Ad-Hoc and Wireless Sensor Networks**. Proceedings - Conference on Local Computer Networks, LCN, 333-338.
- O. Alfandi, A. Bochem, K. Bulert, D. Hogrefe & A. Maier (2015). **Received signal strength indication for movement detection**. 2015 8th International Conference on Mobile Computing and Ubiquitous Networking, ICMU 2015, 82-83.
- O. Alfandi, A. Bochem, A. Kellner, C. Göge & D. Hogrefe (2015). **Secure and authenticated data communication in wireless sensor networks**. Sensors (Switzerland), 15(8), 19560-19582.
- S. Adhatarao, O. Alfandi, Bochem, A. & D. Hogrefe (2015). **Smart parking system for vehicles**. IEEE Vehicular Networking Conference, VNC 2015 (January), 189-190.
- O. Alfandi, A. Bochem, A. Kellner & D. Hogrefe (2013). **Improving energy efficiency of data communication in a hybrid PKI-based approach for WSNs**. 2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013, 697-700.
- O. Alfandi, A. Bochem, A. Kellner & D. Hogrefe (2011). **Simple secure PKI-based scheme for wireless sensor networks**. Proceedings of the 2011 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2011, 359-364.