PRIVACY-ENHANCING MECHANISMS FOR DECISION-MAKING DAPPS

vorgelegt von
M. Sc.
Robert Muth

an der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Stefan Tai
Gutachter:	Prof. Dr. Florian Tschorsch
Gutachter:	Prof. Dr. Axel Küpper
Gutachter:	Prof. Dr. Arthur Gervais

Tag der wissenschaftlichen Aussprache: 14. September 2023

Berlin 2023

ABSTRACT

Blockchain-based decentralized applications (DApps) promise trust and transparency as their execution can be verified publicly and do not rely on trusted third parties. To this end, DApps leverage smart contracts and immutable blockchain storage that allow all participants to verify their correct execution. This makes it possible to verify all DApp interactions and corresponding blockchain transactions, including past ones, and thus prevents subsequent manipulations without being noticed. Unlike traditional legal contracts and centralized web applications, a smart contract ensures that agreements must be executed as implemented, and invalid interactions will be rejected. In particular, there are no designated third parties or other intermediary instances that can violate a smart contract agreement without being explicitly permitted to do so by its implementation. Moreover, smart contracts will automatically execute agreements once their terms and conditions are met.

These inherent properties make DApps particularly attractive for decision-making use cases, such as decentralized autonomous organizations (DAOs) and governance platforms, allowing stakeholders to decide on funding and change proposals. Such decision-making DApps implement regulated processes for stakeholders who may have different interests but still want to make a consensual decision, e.g., by majority or stakes. However, DApps can only work if their interactions and available data are reliable and accurate. To avoid potentially negative outcomes for the stakeholders, DApps must implement mechanisms to ensure that only trusted data is processed. For instance, they must authenticate their data and users; otherwise, binding decision-making processes could be influenced or exploited by malicious parties. On the one hand, DApps can ask for user credentials to ensure that only authorized users can interact with them. On the other hand, DApps put users' privacy at risk by requesting personal data that is processed on-chain, e.g., user names, addresses, or other personal information. This is particularly important because all data from all interactions with a smart contract will be available to everyone on the blockchain and can no longer be removed. For this reason, privacy-enhancing mechanisms that limit access to personal data and keep it private from the public become relevant for decision-making DApps. At the same time, the secure and transparent on-chain execution allows users to verify how their personal data is processed securely and privately, without being disclosed to third parties.

In this thesis, we study how privacy-enhancing mechanisms, i.e., zero-knowledge proofs, anonymous credentials, and secure off-chain

communication channels, can be implemented in decision-making DApps. In particular, we focus on how to implement as much of the cryptographic program logic as possible in a smart contract and execute it on-chain, including routines for privacy protection and proof verification. Furthermore, we evaluate the capabilities of privacy-enhancing mechanisms for DApps in a real-world deployment by implementing a new DApp for urban participation. To this end, we evaluate practical feasibility requirements, i.e., cost and scalability, through the decision-making processes of real-world construction projects. We use these insights to identify privacy challenges and constraints in order to design feasible on-chain privacy solutions for decision-making DApps. Specifically, we contribute privacy-focused mechanisms and proof-of-concept implementations for secure on-chain key exchange, anonymous credentials verification, and anonymous voting for DApps.

ZUSAMMENFASSUNG

Blockchain-basierte dezentrale Anwendungen (DApps) versprechen Vertrauen und Transparenz, da ihre Ausführung öffentlich überprüft werden kann und nicht von Dritten abhängig ist, denen vertraut werden muss. Zu diesem Zweck nutzen DApps Smart Contracts und den unveränderlichen Blockchain-Speicher, welche es allen Teilnehmern ermöglichen, die korrekte Ausführung zu überprüfen. Dies ermöglicht es, alle DApp-Interaktionen und die entsprechenden Blockchain-Transaktionen, auch schon vergangene Transaktionen, zu verifizieren und so unbemerkte, nachträgliche Manipulationen zu verhindern. Im Gegensatz zu herkömmlichen juristischen Verträgen und zentralisierten Webanwendungen stellt ein Smart Contract sicher, dass Vereinbarungen wie implementiert ausgeführt werden müssen und ungültige Interaktionen zurückgewiesen werden. Insbesondere gibt es keine designierten Dritten oder andere zwischengeschaltete Instanzen, die eine Smart-Contract-Vereinbarung verletzen könnten, ohne dass dies durch die Implementierung ausdrücklich erlaubt wäre. Außerdem führen Smart Contracts Vereinbarungen automatisch aus, sobald ihre Bedingungen erfüllt sind.

Diese inhärenten Eigenschaften machen DApps besonders attraktiv für Entscheidungsfindung-Anwendungsfälle, wie z. B. dezentrale autonome Organisationen (DAOs) und Governance-Plattformen, die es den Beteiligten ermöglichen, über Finanzierungs- und Änderungsvorschläge zu entscheiden. Solche DApps zur Entscheidungsfindung implementieren geregelte Prozesse für Interessengruppen, die zwar unterschiedliche Interessen haben, aber dennoch eine einvernehmliche Entscheidung treffen wollen, z. B. durch Mehrheit oder Beteiligungen. DApps können jedoch nur funktionieren, wenn ihre Interaktionen und verfügbaren Daten zuverlässig und genau sind. Um potenziell negative Ergebnisse für alle Beteiligten zu vermeiden, müssen DApps Mechanismen implementieren, die sicherstellen, dass nur vertrauenswürdige Daten verarbeitet werden. So müssen sie beispielsweise ihre Daten und Nutzer authentifizieren, da sonst verbindliche Entscheidungsprozesse von böswilligen Parteien beeinflusst oder ausgenutzt werden könnten. Einerseits können DApps z. B. Benutzerdaten abfragen, um sicherzustellen, dass nur autorisierte Benutzer interagieren können. Andererseits gefährden DApps die Privatsphäre der Nutzer, wenn sie persönliche Daten abfragen, die auf der Blockchain verarbeitet werden, z. B. Namen der Benutzer, Adressen oder andere persönliche Informationen. Insbesondere, weil alle Daten aus allen Interaktionen mit einem Smart Contract für jeden auf der Blockchain verfügbar sind und nicht mehr entfernt werden können. Aus diesem Grund werden Datenschutz-Mechanismen für entscheidungsrelevante

DApps relevant, die den Zugriff auf persönliche Daten begrenzen und diese vor der Öffentlichkeit schützen. Gleichzeitig ermöglicht die sichere und transparente Ausführung auf der Blockchain den Nutzern zu überprüfen, wie ihre persönlichen Daten sicher und privat verarbeitet werden, ohne dass sie an Dritte weitergegeben werden.

In dieser Dissertation untersuchen wir, wie Mechanismen zur Verbesserung der Privatsphäre, z. B. mit Zero-Knowledge-Proofs, dem Konzept von Anonymous Credentials und sicheren Off-Chain-Kommunikationskanälen, in Entscheidungsfindung-DApps implementiert werden können. Insbesondere konzentrieren wir uns auf die Frage, wie möglichst viel der kryptographischen Programmlogik in einem Smart Contract implementiert und auf der Blockchain ausgeführt werden kann, einschließlich Routinen für den Schutz der Privatsphäre und die Überprüfung von Beweisen. Ferner evaluieren wir die Möglichkeiten von Technologien zur Verbesserung der Privatsphäre für DApps in einem realen Einsatz, indem wir eine neue DApp für die städtische Partizipation implementieren. Zu diesem Zweck bewerten wir die Anforderungen an die praktische Durchführbarkeit, also Kosten und Skalierbarkeit, anhand von Entscheidungsprozessen bei tatsächlichen Bauprojekten. Wir nutzen diese Erkenntnisse, um Herausforderungen und Einschränkungen für den Datenschutz zu identifizieren, um praktikable On-Chain-Datenschutzlösungen für DApps zur Entscheidungsfindung zu entwickeln. Insbesondere erarbeiten wir datenschutzorientierte Mechanismen und Proof-of-Concept-Implementierungen für einen sicheren On-Chain-Schlüsselaustausch, anonyme Identitäts-Verifizierung und anonyme Abstimmungen für DApps.

PUBLICATIONS

Publications that are an integral part of this thesis:

- [MT23]: Robert Muth and Florian Tschorsch. "Tornado Vote: Anonymous Blockchain-Based Voting." In: *International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2023, pp. 1–9
- [Mut+22a] Robert Muth, Tarek Galal, Jonathan Heiss, and Florian Tschorsch. "Towards Smart Contract-based Verification of Anonymous Credentials." In: *Financial Cryptography Workshop on Trusted Smart Contracts*. Vol. 13412. Lecture Notes in Computer Science. Springer, 2022, pp. 481–498
- [Mut+22b]: Robert Muth, Beatrice Ietto, Kerstin Eisenhut, Jochen Rabe, and Florian Tschorsch. "Lessons Learned: Transparency in Urban Participation Utilizing Blockchains." In: *Eurasian Studies in Business and Economics*. In publication. Springer, 2022
- [MT21]: Robert Muth and Florian Tschorsch. "Empirical Analysis of On-chain Voting with Smart Contracts." In: *Financial Cryptography Workshop on Trusted Smart Contracts*. Vol. 12676. Lecture Notes in Computer Science. Springer, 2021, pp. 397–412
- [MT20]: Robert Muth and Florian Tschorsch. "SmartDHX: Diffie-Hellman Key Exchange with Smart Contracts." In: *International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE, 2020, pp. 164–168
- [Mut+19]: Robert Muth, Kerstin Eisenhut, Jochen Rabe, and Florian Tschorsch. "BBBlockchain: Blockchain-Based Participation in Urban Development." In: *International Conference on eScience*. IEEE, 2019, pp. 321–330

Other works by the author may be cited, but are not included in this thesis:

- [Iet+23]: Beatrice Ietto, Jochen Rabe, Robert Muth, and Federica Pascucci. "Blockchain for citizens' participation in urban planning: the case of the city of Berlin. A value sensitive design approach." In: *Elsevier Cities Journal* 140 (2023), p. 104382
- [Hei+22]: Jonathan Heiss, Robert Muth, Frank Pallas, and Stefan Tai. "Non-disclosing Credential On-chaining for Blockchain-Based Decentralized Applications." In: *International Conference on*

Service-Oriented Computing (ICSOC). Vol. 13740. Lecture Notes in Computer Science. Springer, 2022, pp. 351–368

- [GMF22]: Marcel Gregoriadis, Robert Muth, and Martin Florian. "Analysis of Arbitrary Content on Blockchain-Based Systems using BigQuery." In: *Companion of The Web Conference*. WWW '22. ACM, 2022, pp. 478–487

- [Iet+22]: Beatrice Ietto, Kerstin Eisenhut, Robert Muth, Jochen Rabe, and Florian Tschorsch. "Transparency in Digital-Citizens Interfaces Through Blockchain Technology: BBBlockchain for Participation Processes in Urban Planning." In: *European Technology and Engineering Management Summit (E-TEMS)*. IEEE, 2022, pp. 65–71

- [Rab+21]: Jochen Rabe, Beatrice Ietto, Robert Muth, Kerstin Eisenhut, and Federica Pascucci. "Citizens' Engagement in Urban Development through Blockchain: a Human-centered Design Approach." In: *International Conference on Technology Management, Operations and Decisions (ICTMOD)*. IEEE, 2021, pp. 1–6

- [Bra+19]: Samuel Brack, Robert Muth, Stefan Dietzel, and Björn Scheuermann. "Recommender Systems on Homomorphically Encrypted Databases for Enhanced User Privacy." In: *Local Computer Networks Conference (LCN) Symposium*. IEEE, 2019, pp. 74–82

- [Bra+18]: Samuel Brack, Robert Muth, Stefan Dietzel, and Björn Scheuermann. "Anonymous Datagrams over DNS Records." In: *Local Computer Networks Conference (LCN)*. IEEE, 2018, pp. 536–544

ACKNOWLEDGMENTS

My entire academic journey has been accompanied by so many incredible people. First and foremost, Florian Tschorsch, whose expertise and endless patience allowed me to explore and pursue my personal research interests. Special thanks to Prof. Dr. Arthur Gervais, Prof. Dr. Axel Küpper, and Prof. Dr.-Ing. Stefan Tai for their time, effort, and all the feedback. And all of my fellow colleagues, Christoph Döpmann, Elias Rohrer, Erik Daniel, and Saskia Nuñez von Voigt, for putting up with me in the same office. As well as the new colleagues, Carolin Brunn, Charmaine Ndolo, Lukas Gehring, and Maximilian Weisenseel. And my project colleagues Beatrice Ietto, Jochen Rabe, and Kerstin Eisenhut. Not to forget, the colleague from upstairs, Jonathan Heiss. Also, Björn Scheuermann and his former team at the Humboldt-Universität zu Berlin, especially Samuel Brack. Furthermore, all the students I had the honor to support: Aljoscha F., Carolin N., Elio D., Florens B., Jonathan P., Jorrit P., Natalie P., Nathan D., Stanislav T., Thore W., and Tarek Galal. This fruitful collaboration was only possible thanks to all colleagues at the Technische Universität Berlin and the Einstein Center Digital Future.

I have no doubt that this work would not have been possible without the support of my entire family. I cannot express my gratitude enough to my parents, Christine Muth and Thomas Muth, who have been so incredibly dedicated to my personal success and well-being, as well as my grandparents, Anneliese Sailer, Bernd Muth, Franz Sailer, and Waldtraut Muth. And last but not least, my partner Alexander—*je t'aime*.

Thank you all,
$\sim$ Robert

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Ethereum introduced full-fledged smart contracts in 2017, which are custom, Turing-complete programs that can be deployed, verified, and executed independently by all blockchain peers [@But23b; Woo22]. Ethereum therefore allows anyone to access and execute a deployed smart contract without the need for a trusted third party (TTP), such as an external server or another user [Mil19]. Smart contracts can also be deployed by anybody, however, they cannot be modified or deleted anymore once they are deployed [Woo22]. To this end, smart contracts promise greater trust than traditional written contracts because their program logic is binding, not open to vague interpretation, and not necessarily dependent on TTPs [AM17; Sza97]. As a result, once deployed, smart contract-based agreements must be followed and will eventually be executed in a virtually automated process [@But23b].

With Ethereum, smart contracts can be used to develop decentralized applications (DApps) [Met20]. A DApp leverages one or more smart contracts and provides a frontend for users to conveniently interact with the DApp's smart contracts. Furthermore, DApps promise to be more reliable, available, and censorship-resistant than centralized web applications because their smart contracts do not necessarily depend on servers or TTPs that can fail or cause errors. To this end, several popular DApps have emerged, such as decentralized autonomous organizations (DAO), decentralized financial applications (DeFi), a decentralized name server (ENS), and more [@Jen16; Sch21; Xia+21; Wu19].

In fact, because of the blockchain-inherent properties, such as trust and transparency, it is particularly attractive for DAOs and governance DApps to implement *decision-making* processes. For example, DAO and governance DApps therefore allow users to make decisions in a consensual and systematic process, e.g., by majority or stakes. Such decision-making DApps implement proposal-based votings on predefined outcomes that, if approved, are automatically executed by the smart contracts, e.g., cryptocurrency transfers. The powerful technical capabilities of smart contracts can implement even complex decision-making processes. However, while such decision-making processes may involve complex rules and logic, they can still be trusted to make and execute a decision in a transparent and trustworthy manner.

Hence, at first glance, DApps benefit from their smart contracts' promised trust and transparency; however, many DApps must make

trade-offs in their trust assumptions and decentralization to protect their users' personal data. Processing personal data with smart contracts requires special attention to privacy and security, as all data on a public permissionless blockchain is publicly exposed and cannot be deleted. For example, established financial DApps require their users to identify themselves to prevent money laundering [MSS22], but any kind of personal identity data on a public blockchain compromises user privacy. The same issue applies to decision-making DApps that rely on reliable user identification for trusted voting processes. Therefore, many DApps use *oracles* to delegate personal data processing to centralized TTPs, such as external servers, and therefore rely on trusted, external infrastructures [HET19; CE21; Müh+20]. However, while oracles may be a solution for data protection, such dependencies jeopardize the decentralization and transparency of a DApp.

Nevertheless, privacy-enhancing off-chain computation techniques have emerged that trade off the decentralization and trust assumptions of a DApp while ensuring on-chain verifiability. With such techniques, privacy can be achieved with off-chain computation that is on-chain verifiable. Therefore, personal data can be processed by a DApp without relying on oracles or TTPs. For example, verifiable off-chain computation with on-chain verifiable zero-knowledge proofs (ZKP) [ET18], trusted execution environments (TEE) [Kar+21], and homomorphic encryption (HE) [Ste+19; Ste+22]. Unfortunately, these techniques are still not very practical for many DApp use cases, or are difficult to implement [Len+22]. For example, ZKP generation becomes memory and computationally intensive as soon as it has to prove complex off-chain calculations [ET18; Gro16]. Homomorphic encryption also becomes very computationally intensive as soon as it is used for complex calculations [Bra+19; Arm+15]. Furthermore, while TEEs promise performant execution and could help overcome scalability and privacy issues, they require proprietary, trusted hardware, which currently prevents their widespread use for permissionless DApps [Erw+20; GY19].

The main problem with many privacy-enhancing techniques and mechanisms for DApps is that they cannot perform resource-intensive privacy operations on-chain, and their artifacts (e.g., keys and proof routines) cannot be stored economically due to the scalability constraints of the blockchain [Hei+22]. However, many existing DApps already successfully implement privacy-enhancing techniques by performing resource-intensive operations off-chain while remaining verifiable on-chain [@Ale19; GAC19; SHS20]. For this purpose, the smart contracts do not perform the privacy operations on-chain, but expect predefined routines to be executed off-chain on the client computers and return the results and a corresponding proof [Gro16; ET18]. The results can then be cryptographically verified to confirm that the intended operations were performed as expected, without having to repeat the entire computation on-chain with a smart contract. For example, a ZKP can be

generated off-chain to attest the validity of a cryptographically signed date of birth as proof of legal age from a trusted instance without revealing the exact date in clear-text. A smart contract therefore only verifies the validity of the proof on-chain, and the date of birth is not published on the blockchain [Hei+22]. The most important difference between this approach and an oracle is that it does not rely on a centralized authority whose processing cannot be publicly verified. Unfortunately, such ZKPs may consist of large key files or other artifacts that are expensive to store on the blockchain; however, with certain ZKP techniques it is possible to distribute these artifacts off-chain and verify their execution results with an efficient on-chain verifier [Gro16]. In summary, hence, decision-making DApps face the challenge of reconciling the constraints of a blockchain with the needs of privacy-enhancing techniques, and of finding suitable mechanisms that do not compromise the trust assumption of DApps. To this end, decision-making DApps have to ensure that trust-relevant processes remain on-chain and that resource-intensive operations can be outsourced without any loss of trust.

1.2 RESEARCH QUESTIONS

In this thesis, we study the potentials, capabilities, and barriers of privacy-enhancing techniques and mechanisms for public decision-making DApps. To this end, we design and develop a new Ethereum DApp for urban participation that inherently depends on privacy for successful decision-making outcomes. With this DApp, we identify current challenges for privacy-enhancing techniques with public, permissionless blockchains in practice. Therefore, this thesis addresses the following two research questions:

- How to implement verifiable privacy protection mechanisms for decision-making DApps? In particular, how to verify personal data without exposing it to the public for reliable decision-making processes?
- Which decision-making processes can be executed off-chain while keeping their results on-chain verifiable to maintain trust in smart contract execution? Specifically, how to implement as many cryptographic procedures as possible in a smart contract with respect to the capabilities and principles of the blockchain?

More specifically, this thesis focuses on the implementation of privacy protection mechanisms for anonymous voting, anonymous authentication, and secure key exchange for DApps on permissionless blockchains with practical, real-world requirements. In contrast to theoretical privacy research, this raises a number of practical issues, such as limited blockchain scalability, costs, and accessibility. In this context, we identified a gap in the existing research on decision-making DApps, which we address as follows.

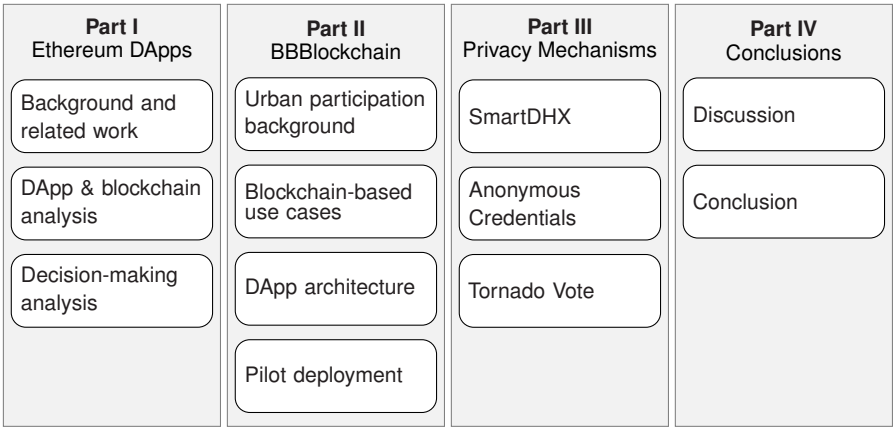


Figure 1. Organization of this thesis.

1.3 OUTLINE

First, this thesis begins with an introduction to DApps and the relevant technical background. Additionally, we analyze existing DApps to point out their relevance. Second, we develop a new decision-making DApp for urban participation to analyze the current state of DApp development with a focus on privacy. In preparation for answering our first research question, we identify technical limits for implementing on-chain verifiable privacy mechanisms based on the experience with our DApp. Third, we contribute new privacy mechanisms to enhance privacy for decision-making DApps: A solution for establishing trusted secure communication channels to address blockchain scalability capabilities, an anonymous credentials verifier for Ethereum, as well as an anonymous DApp-based voting system. Therefore, each contribution presents its own conclusions and key findings. Finally, we conclude this thesis with an open discussion about the role of decision-making for DApps and recap our research questions.

To this end, the main body of this thesis is divided into four parts after this introduction, as shown in Figure 1: Part I focuses on Ethereum DApps. Chapter 2 introduces technical concepts and background knowledge relevant to all following parts. Chapter 3 provides additional background, focusing on blockchain data analysis, and presents an overview of key metrics and the relevance of DApps. Chapter 4 extends this analysis to an empirical exploration of the current state of decision-making DApps.

Part II focuses on a decision-making DApp for urban participation processes. In Chapter 5, we introduce *BBBlockchain*. Chapter 5.1 introduces the basics of urban participation and civic tech. Building on this, Chapter 5.2 proposes suitable DApp-based use cases for urban participation. Chapter 5.3 presents the architecture of BBBlockchain to implement the use cases and specifies requirements for a participation decision-making DApp. In Chapter 5.4, we design and implement

BBBlockchain for a pilot deployment by introducing it in two real construction projects to gain practical experience with a decision-making DApp and evaluate our key findings during the deployment.

Part III presents novel mechanisms for on-chain privacy in decision-making DApps based on the experiences with BBBlockchain. Chapter 6 presents a key exchange protocol whose communication and protocols are completely secured by the blockchain. Chapter 7 presents an on-chain anonymous credentials verifier to improve voting reliability while keeping the voters anonymous. Lastly, Chapter 8 presents a new anonymous voting system, *Tornado Vote*, which is based on a well-established cryptocurrency mixer.

Part IV concludes this thesis. In Chapter 9.1, we discuss the role of our contributions in achieving on-chain privacy for decision-making DApps. Finally, in Chapter 9.2, we recap the research questions and conclude this thesis.

1.4 CONTRIBUTIONS

In summary, we present the following contributions of this thesis:

- An empirical analysis of existing decision-making DApps to explore the current state and to reveal the limitations of on-chain voting with a model-based scalability evaluation.
- *BBBlockchain*, a DApp for public participation processes. Therefore, we investigate and study possible blockchain-based use cases for participation processes in order to implement them in existing building projects. As a result, we propose blockchain-based voting, token-based incentivization, and content timestamping use cases to inform and consult citizens. In an interdisciplinary collaboration between an urban development and a computer science research team, we explore how blockchains can meet real-world requirements and gather practical experience to implement new technical approaches and mechanisms to overcome the barriers identified. To this end, we introduced BBBlockchain in two participation projects in Berlin, Germany from 2019 – 2022.
- *SmartDHX*, a multi-party Diffie–Hellman key exchange protocol to establish secure communication channels between blockchain participants. Unlike existing key exchange implementations, we develop a novel technique to store all execution logic on-chain, thus securing the entire key exchange process on-chain. The main goal is to implement and execute as many cryptographic operations as possible on-chain. To this end, we design two different Diffie–Hellman key exchange protocols, i.e., for two-party and for multi-party key exchanges. The key feature is that the SmartDHX smart contract contains off-chain executables that control the smart contract itself.

- An EVM-compatible, on-chain *anonymous credentials* verifier. Technically, anonymous credentials allow to prove the possession of personal identity features without having to disclose personal data. For example, a person can prove that she or he is eligible to vote and has not previously voted without revealing personal information such as name, address, or date of birth. Anonymous credentials are particularly useful because they can be verified by anyone without third-party involvement. To this end, we implement a smart contract-based, on-chain anonymous credentials verifier which is compatible to existing identity systems.

- *Tornado Vote*, an anonymous voting system to decouple voters' accounts from their votes. Therefore, we are adapting the Tornado Cash cryptocurrency mixer protocol, and ensure the security of our adaptations through formal verification, vulnerability scanners, and unit tests. While we found that Tornado Cash could be used directly to implement anonymous voting with coins, this approach would not allow a fair voting process. Therefore, our implementation adapts the Tornado Cash protocol and implements an additional commitment phase to privately collect all votes before revealing them all-together. We therefore implement a proof-of-concept to evaluate the feasibility of Tornado Vote.

Part I

DECENTRALIZED APPLICATIONS

2 BACKGROUND AND RELATED WORK

In this part, we introduce the technical concepts of decision-making DApps and present related work in Chapter 2. In Chapter 3, we introduce the concepts of blockchain analysis in order to show the importance and relevance of existing DApps. Lastly, in Chapter 4, we focus on decision-making DApps, their underlying technical concepts, and analyze their relevance and technical capabilities.

In general, it is difficult to define blockchain technologies and their concepts in a consistent and universal valid form, as there are too many differences between the different types of blockchains. We therefore introduce the relevant blockchain concepts in the following chapters, i.e., DApps, smart contracts, decision-making, and blockchain-based voting, primarily for Ethereum and EVM-compatible blockchains. Since there may be differences between our definitions and those in other works, the following definitions serve to provide consistency throughout all chapters of this thesis.

2.1 TECHNICAL BACKGROUND

The Ethereum blockchain [@But23b] allows the execution of arbitrary code in the user transactions with *smart contracts* [Woo22]. This enables Ethereum and similar capable blockchains to deploy, host, and execute smart contracts that implement powerful logic and routines. While it can be argued that Bitcoin first introduced smart contracts with *Bitcoin Script* in 2008, Ethereum introduced much more powerful smart contract capabilities in 2015. To this end, the Ethereum blockchain executes smart contracts in the so-called *Ethereum Virtual Machine* (EVM) [Woo22]. Technically, smart contracts consist of assembly-like EVM commands and have access to the current blockchain state. Therefore, smart contracts can hold and transfer cryptocurrencies based on their implemented logic. All users on the Ethereum blockchain can deploy new smart contracts, interact with them, and verify their correct execution. Furthermore, smart contracts can hold cryptocurrencies and transfer them based on their implemented logic. Because of the immutability of the blockchain, a smart contract cannot be removed or changed once it is deployed.

Smart contracts provide a technical interface for interactions, i.e., an application binary interface (ABI). An ABI specifies the available functionality of a smart contract, but is not designed for user interaction. To interact with smart contracts, users typically use a *DApp*. DApps leverage one or more smart contracts and provide a graphical frontend.

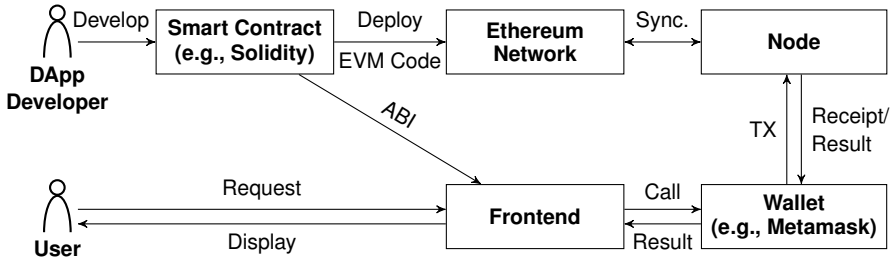


Figure 2. Exemplary DApp setup with a smart contract deployment on the Ethereum blockchain by a DApp developer, and user interactions with a frontend connected to a blockchain node via a wallet.

Usually, DApps provide frontends that are implemented as an interactive web page for standard browsers. The frontend can be hosted on a central web server, following a typical view and data separation model, so it only implements the view and does not store or control any user data. For data access, browser-based frontends require an additional plugin to connect the web page with the blockchain, e.g., Metamask[1], to connect to a synchronized local blockchain node or an external node, e.g., Infura.[2]

Figure 2 shows an exemplary setup for a DApp that can be interacted with using a web frontend. First, the developer deploys the DApp's smart contracts by compiling the source codes and submitting a transaction with the resulting EVM code to the Ethereum blockchain, i.e., the Ethereum Mainnet or a test network. Additionally, the developer generates an ABI that specifies the smart contract functions which have been stripped by the compiler for the deployment. This ABI is provided to the frontend, which uses it to interact with the smart contracts. Next, a user interacts with the frontend, for example, to call a smart contract function or to send a transaction. Alternatively, it is also possible to interact directly with a DApp (without a frontend), but in both cases the developer must provide an ABI or the smart contract source codes. The user can then call a smart contract function without submitting a transaction to the blockchain, allowing the DApp to only display the function's return value; alternatively, a user can also call a function and submit a new transaction to persist the function execution on the blockchain and receive a corresponding receipt.

To date, DApps have been implemented for many different use cases [MM18]. With a current market capitalization of more than 2 billion USD, Ethereum is the second most successful blockchain and the leading platform for hosting DApps [@Coi23]. In the following, we therefore focus primarily on Ethereum and decision-making use cases.

1 https://metamask.io
2 https://infura.io

2.2 DECISION-MAKING DAPPS

The definition of *decision-making DApps* encompasses many blockchain concepts. Therefore, in this section, we introduce the relevant technical background and related work for decision-making DApps. Since there is no universal and precise definition for a decision-making DApp, we introduce each of the related concepts separately. We therefore distinguish between the concepts of a DApp and a smart contract, decision-making and governance, and lastly, voting and e-voting. Furthermore, we introduce the technical background for building a decision-making DApp and analyze deployed DApps. Lastly, we present related work on privacy for decision-making DApp use cases.

Decision-Making, DAO, and Governance DApps

With blockchain-based decision-making, users agree on rules that can be implemented with smart contracts or in the blockchain protocol. The term *decision-making* is not well defined and applies to a broad range of DApp use cases [Zio+19; Yan+19; EAA22]. However, most decision-making applications have one thing in common, usually some type of voting process. Therefore, for this thesis, we define decision-making DApps as a general term for DApps that implement *user-driven* decision-making processes, such as voting by majority or stakes. One of the most prominent examples of a decision-making DApp is *The DAO* [@Jen16], which attracted a lot of attention in the Ethereum community when it launched in 2016. Decentralized autonomous organizations (DAO) allow stakeholders to manage funds in a decentralized manner without trusted intermediaries or management entities. To this end, DAOs typically issue their own tokens [@Jen16] to their users as voting rights, which decouples voting rights from fluctuations in cryptocurrency exchange rates. Users can then make organizational proposals, similar to fundraising and crowdfunding campaigns. Other users use their tokens to signal acceptance and attach funding in the form of cryptocurrency, or they reject a proposal. Finally, accepted proposals can then transfer the funds to a dedicated smart contract and return the profits (i.e., tokens or cryptocurrency) to the supporters [@Jen16].

Another advantage of DAO is that blockchains allow them to transfer funds globally without relying on banks or other third-party institutions with their cryptocurrencies. In fact, DApps can implement asset transfers easily and without much administrative overhead, which would not be possible with traditional bank transfers. Combined with the immutability of an underlying public, permissionless blockchain, payment transfers can be confirmed and tracked internationally. But this also has its advantages and disadvantages: Transactions are reliable once they have been sent, verified, and confirmed by a sufficient

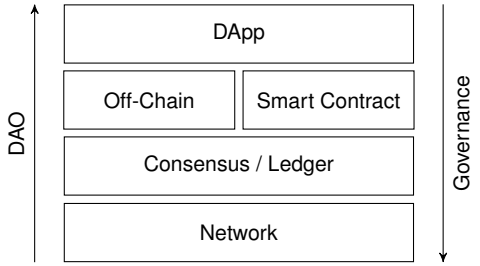


Figure 3. Simplified blockchain layer model showing the direction of impact for DAO and governance DApps.

number of blocks. On public, permissionless blockchains, however, transactions are open to everyone, which means there is no privacy in the first place. The immutability also led to a hard fork of the Ethereum Mainnet in 2016 [MS19; @Wil16], when the smart contracts of *The DAO* were exploited. However, many other successful DAOs have been implemented, such as MakerDAO [@The17], to realize a token-based stable coin on Ethereum.

Another example of decision-making DApps are *governance* DApps. From a technical point of view, governance DApps and DAOs are not too different. In fact, the technical overlap is significant, as both implement voting processes for their users. The main difference is, that governance use cases operate in and for their ecosystems, such as users and developers voting together on technical protocol changes. For example, Bitcoin implements version bits for its governance on the protocol level [@Pan17], and the Dash blockchain provides a full-fledged DApp platform for its governance (see Section 4.3 for more details). In comparison, as shown in Figure 3, DAOs use blockchain technologies as a means for trusted decision-making processes that can technically have an impact beyond the blockchain, while governance DApps are used to improve the technical ecosystem. Also, while the underlying technology of blockchains must constantly make decisions in order to operate, we mainly build on the DApp layer of blockchains in this thesis.

Ultimately, for this thesis, the term decision-making DApps encompasses DAOs, governance DApps, and other DApps that implement user-driven voting. However, we distance ourselves from blockchain-based e-voting for elections, as we will explain in the following related work.

2.3 RELATED WORK

In this section, we present related work, which is structured along the lines of our different research directions. To this end, we first present related work on blockchain-based voting and draw a distinction between it and blockchain-based e-voting for elections. Second, we present

related work on privacy with public blockchains, and introduce established techniques and DApps. Next, we present related work on blockchain scalability and transaction fees to pave the way for the research problems and requirements that follow. Finally, we present related work on off-chain computation as a solution to scalability, costs, and privacy issues.

Blockchain-Based Voting and E-Voting

There are many other DApps that implement voting as utility for their primary use case (as we will see in Chapter 4), as well as specialized voting DApps [AKW20; MSH17; SGY20]. However, while decision-making DApps also encompass e-voting for elections (e.g., for nation-wide elections), we exclude this use case in this thesis, mainly for yet unresolved privacy and security reasons [Hei+18; BV16; ECW21].

In this thesis, we primarily focus on blockchain-based voting for DAO and governance use cases. Furthermore, while blockchain-based voting is a commonly implemented concept [Gar+19; @Jen16], we specifically focus on *anonymous* voting. For example, McCorry, Shahandashti, and Hao [MSH17] implement an anonymous voting protocol based on the *Open Vote Network*. It is implemented for Ethereum and allows up to 40 voters. A self-tallying mechanism computes the final results, but also constitutes a computational bottleneck which prevents practical implementation in larger-scale voting DApps. In order to improve scalability, Seifelnasr, Galal, and Youssef [SGY20] introduce an off-chain tallying process for the voting implementation in [MSH17]. Therefore, they rely on at least one trusted instance.

Killer et al. [Kil+22] implement an anonymous blockchain-based voting that is receipt-free. However, their proposal requires its own permissioned blockchain and therefore cannot be deployed on Ethereum. By relying on a proof-of-authority consensus, Killer et al. achieve significant better scalability and performance than with a permissionless blockchain.

Other blockchain-based voting protocols defer to completely private networks [Yu+18; Har+18; Hja+18].

Privacy on Public Blockchains

Public blockchains, whether permissionless or permissioned, enable verifiability and thus promise trust. However, this also puts privacy at risk, and not just for anonymous voting. If all transactions are public, all user interactions can be monitored. Although it is possible to create new accounts without much technical effort and thus achieve pseudonymity, it remains difficult to keep interactions between accounts and smart contracts secret. Therefore, several public permissionless blockchains have already taken measures to address privacy. For example, Mon-

ero [@Van13] and Zcash [@Hop+16] implement privacy mechanisms to protect or obfuscate cryptocurrency transfers directly in their protocol. Also Ethereum recently proposed the concept of *stealth addresses* to achieve privacy [@But23a].

Other approaches have also emerged that do not require blockchain protocol changes, but instead build on the technical capabilities of the blockchains. For example, CoinShuffle [RMK14] for Bitcoin, or Tornado Cash [@Ale19] for Ethereum. Both examples build on the principle of mixing coins in an *anonymity set*. To this end, multiple coin owners deposit the same amount of coins into a shared wallet, and then withdraw the coins with a different account at a later time. As long as there is no traceable connection between the deposit and withdrawal transactions, an observer cannot be certain that the deposit and withdrawal accounts belong to the same person. More specifically, an observer can only relate two transactions with a probability of one to the number of other participants in the anonymity set. Nevertheless, advanced clustering techniques, time correlations, and technical metadata could be used to reduce the size of the anonymity set [Mös+18; Vic20; Bér+21] and thus compromise the privacy of the participants.

The same privacy issues apply to smart contract interactions processed through public transactions. Decision-making DApps, in particular, must protect their users' privacy for reliable outcomes. For example, the reliability of a voting can be improved by making the entire voting process anonymous. This can give voters a greater sense of democratic freedom and confidence to make their decision without outside pressure or influence. Alternatively, for more complicated decision-making processes, participants may need to discuss details via a secure communication channel. Currently, most of the communication for DAOs takes place off-chain in web forums. Either publicly in public discussion forums or privately via direct messaging. Therefore, such communication platforms do not offer the same trust and transparency properties as blockchain-based DApps. However, they are easily scalable and do not incur significant costs, which are aspects that we will inspect in more detail in the following.

Scalability and Transaction Fees

Besides privacy, there is another major obstacle to widespread adoption of public permissionless blockchains at the current stage of development: scalability or transactions throughput, respectively. Due to limited block sizes and static block generation rates, the transaction throughput of a blockchain is limited [Woo22]. The limited scalability, however, leads to sometimes high transaction costs and longer transaction processing times.

For decision-making DApps, this can occasionally lead to two major problems: First, the cost of DApp interactions is significantly higher

than for centralized applications. Second, DApp interactions may not be executed due to the scalability limitations, which may compromise the inherent censorship-resilience of the blockchain.

Blockchains use *transaction fees* for Sybil and spam protection, and to reward active miners to keep the network up and running [TS16]. To this end, different blockchains implement different techniques for the calculation of transaction fees. For instance, as the pioneer of public permissionless blockchains, i.e., Bitcoin, implements a fee system based on supply and demand [Nak08]. In principle, the senders of new transactions determine themselves how much they are willing to pay for their transactions, before they submit them to the miners. The miners then decide for themselves whether they are willing to accept a transaction when they generate the next block. Since the size of each block is limited, the fees naturally increase when there are a lot of transactions, i.e., the blocks are filled more. Other blockchains, e.g., Ethereum [@But23b] and Dash [@DD18], implement similar approaches for transaction fees.

Nevertheless, one of the biggest problems with such a supply-and-demand control system is that it allows miners to disproportionately reward themselves beyond their efforts. Especially if the blockchain network experiences a lot of traffic, transaction costs will increase significantly, and miners will only include the transactions with the highest reward for them. Ethereum, therefore, was looking into alternatives for calculating the transaction fees. First, Ethereum already implements a different fee calculation than Bitcoin, that is based on the complexity and resource consumption of transactions [Woo22]. To this end, Ethereum uses variable costs for executing commands in a transaction, and charges them in the form of *Gas*, an Ethereum specific pseudo-unit. For example, simple arithmetic operations are much cheaper than storage allocations. However, Ethereum started with an approach similar to other permissionless blockchains: with a supply and demand regulation. A transaction sender specified how much Ether (i.e., the native cryptocurrency) she or he is willing to pay per Gas [Woo22]. As a result, when demand increases significantly, transaction fees on Ethereum suffer from the same issues.

EIP-1559 In 2019, Buterin et al. proposed a new transaction fee calculation mechanism for Ethereum with EIP-1559 [@But+19]. The authors' intention is that miners will no longer be able to benefit disproportionately due to much traffic. In summary, the Gas costs remained the same, but the exchange rate, i.e., the *base fee per Gas*, will be regulated with a demand controlling mechanism. To this end, the base fee increases if the most recent block required too much Gas to process all transactions, but decreases if there is enough residual capacity. EIP-1559 therefore specifies a target threshold of 15M Gas, however, blocks can actually process up to 30M Gas. Whenever the threshold is crossed, the base fee per Gas will increase or decrease proportionally

by $\pm 12.5\,\%$. However, the base fee does not go to the miners. Instead, it is multiplied by the amount of Gas used in the transactions and burned. It is still possible, but not mandatory, for the transaction sender to add a small tip to the miners for prioritized processing.

As a result, transaction fees can no longer increase abruptly, and miners no longer have an incentive to favor the most expensive transactions [@But+19]. In addition, participants gain a degree of predictability about transaction costs in the near future, as they can increase by a maximum of 12.5 % per block. This prevents transactions from unexpectedly disappearing into the mempool, which is a pool of pending transactions.

PROOF-OF-WORK & PROOF-OF-STAKE EIP-1559 was not the only major protocol change for Ethereum. In September 2022, Ethereum switched from a proof-of-work (PoW) consensus algorithm to proof-of-stake (PoS). The main goals for switching to PoS were to save energy by not wasting computing power on solving random puzzles [@But17b], and to improve the security of the network [@But20]. With this hard fork (also known as *The Merge*), the miners' leader election no longer depends on randomness and available computing power, but on individual cryptocurrency stakes. When the miners follow the protocol correctly, they are rewarded with the block reward and priority tips. However, if they fail to comply with the protocol, they will receive a penalty.

Ultimately, while changing the consensus algorithm had a huge impact on the consumption of energy resources due to the elimination of PoW, the available capabilities of the Ethereum network remained the same. Therefore, the impact on network congestion and transaction fees can be considered moderate, as we show in the following chapter. Nevertheless, the switch from PoW to PoS can be considered a success, both technically and operationally [Cor+23].

Off-Chain Solutions and Computations

Blockchains implement leader election mechanisms and transaction fees to prevent Sybil attacks and too much traffic in general [TS16]. Unfortunately, transaction fees can be very high depending on the current transaction load and the complexity of a transaction. However, *off-chain solutions* can be used to reduce transaction fees, e.g., to enable micro-payments. For example, the Lightning network [@PD16] enables cheaper and faster Bitcoin transfers by establishing off-chain payment channels between participants. As a result, only the final result of one or more payments will be recorded on the blockchain. Similarly, Raiden [@Est21] was developed to improve the scalability of Ethereum.

DApps can also leverage off-chaining to reduce transaction costs. With verifiable off-chain computations and ZKPs, DApps can run

programs on client machines that would be very expensive to run on-chain [ET18]. However, the correct execution of the program can be verified on-chain at lower cost and without repeating the entire program execution [Gro16]. In addition, off-chain computations can process personal data on local machines, so it does not need to be stored on the blockchain. Therefore, off-chain computation can also be seen as a suitable solution for achieving privacy. We will look into off-chain computations and ZKPs in detail in Sections 7 and 8, where we leverage them to achieve privacy.

Blockchain Analysis

In this thesis, we analyze the variety of decision-making DApps and on-chain voting, independent of any specific use case or property. A series of papers explore blockchain data in terms of several other aspects, including privacy [Bér+21; RH11], data storage [Mat+18; GMF22], and smart contract metrics [Pin+19]. There is, moreover, model-based analysis on the security [KDF13] and scalability [Cro+16] of blockchains available. Specifically for voting, Heiberg et al. [Hei+18] evaluate the trade-offs of blockchain-based voting on a qualitative level. They discuss aspects such as complexity, costs, and scale, which go in a similar direction as our paper. However, we will complement their discussion with an empirical analysis and provide new insights, for example, on the magnitude of on-chain voting. Methodologically similar to our approach are [FFB19; RYM19; VL19; Pin+19]. Victor and Lüders [VL19] inspect the Ethereum blockchain for token implementations, which are managed by the ERC-20 [@VB15] smart contract template. While EIP-1202 [@ZEX18] proposes a similar standard for voting smart contracts, it is not as established as the ERC-20 compatible token standards are. Fröwis, Fuchs, and Böhme [FFB19] search for token-related behavior with symbolic execution analysis techniques and compare the effectiveness of both methodologies. The diversity of voting schemes, features, and privacy mechanisms make it more difficult to identify voting smart contracts by their EVM bytecode. In contrast to automated smart contract inspection, the authors of [RYM19; Pin+19] present approaches that are based on manually collected exchange listings and corresponding source code publications on CoinMarketCap[3] and Etherscan.

3 https://coinmarketcap.com

3

DAPP ANALYSIS

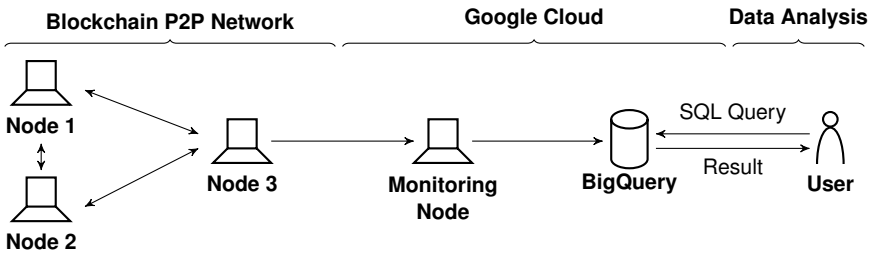


Figure 4. Data analysis infrastructure with Google BigQuery.

In this chapter, we scan the Ethereum Mainnet to analyze its key metrics in order to show its relevance. Therefore, we present background knowledge and related work to analyze the blockchain and related data, such as historical exchange rates. In the following chapter, we use this knowledge to specifically analyze decision-making DApps on the Ethereum Mainnet.

Blockchain data can be analyzed in several ways. A fully synchronized blockchain client downloads and verifies all block data, so it can be used for data analysis right away. Unfortunately, most blockchain client implementations are so optimized for their cryptocurrency use case that efficient data analysis becomes very difficult. For example, most blockchain clients do not need to maintain database indexes to quickly access arbitrary historical data. This makes accessing and analyzing the data very slow and requires a lot of memory. Therefore, blockchain clients can download block data and export it to a more powerful database via an extract, transform, load (ETL) pipeline. By doing so, it is possible to insert block data, transactions, etc. into a relational database (RDB). The data can then be easily managed and analyzed using SQL queries.

Therefore, the *Ethereum ETL*[4] project provides an automated process to convert the Ethereum blockchain into CSV files that can be imported into an RDB. Unfortunately, inserting entire blockchains into a database requires a lot of storage, and analyzing them still requires a lot of memory as well. For comparison, the Ethereum client Geth required approximately 1 TB of storage in May 2023. Fortunately, there are data warehouse platforms that already synchronize the current blockchain into an RDB and provide read-only access to their customers. For example, Google Cloud BigQuery[5] provides access to multiple syn-

4 https://github.com/blockchain-etl/ethereum-etl (Accessed: 2023-06-08)
5 https://console.cloud.google.com/bigquery

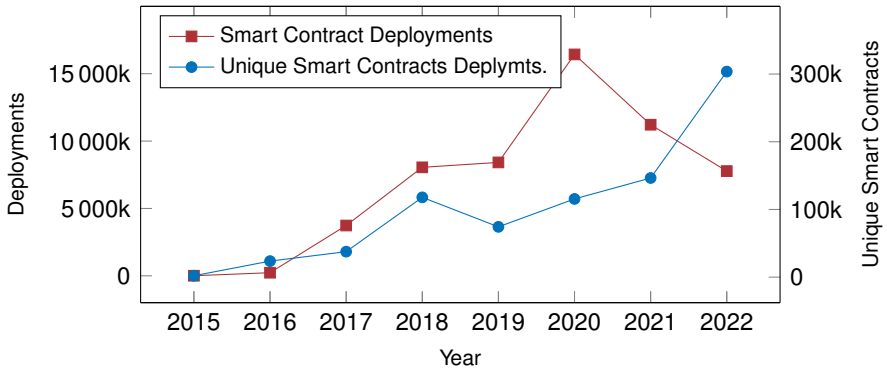


Figure 5. The number of all new smart contract deployments and the number of new smart contracts with unique EVM byte code on the Ethereum Mainnet per year.

chronized blockchains, including Bitcoin [Nak08], Ethereum [Woo22], Dash [@DD18], Dogecoin[6], and others. As shown in Figure 4, BigQuery uses a monitoring node that is synchronized with the current state of the blockchain networks. Once blocks can be considered confirmed, the latest blocks, transactions, receipts, smart contracts, and other blockchain data are inserted into public, read-only RDB tables. Users can then use SQL queries to analyze the blockchains without worrying about any computational requirements. BigQuery therefore orchestrates powerful server resources in the background, which are charged according to the amount of data processed. In addition, more blockchain-specific data analytics platforms, such as Etherscan[7], have emerged. Such platforms analyze transactions and enrich the raw data with insights from source codes, exchanges, and user inputs.

In this thesis, we use the different approaches to gain insights into the blockchain, especially for decision-making use cases (as shown in Chapter 4). We thus combine multiple sources for an extensive data analysis pipeline. We specifically focus on Ethereum, but we will also look at other blockchains for comparison where relevant. However, in the following, we first analyze general blockchain and smart contract metrics to highlight the overall relevance of DApps.

3.1 KEY METRICS OF ETHEREUM

Deployments

To get an overview of past demand, we analyzed the number of smart contract deployments on the Ethereum Mainnet. Figure 5 shows the number of all smart contract deployments by year. Therefore, we identified the date of all successful smart contract deployments and summed them by year. Since one and the same smart contract implementation

6 https://dogecoin.com
7 https://etherscan.io

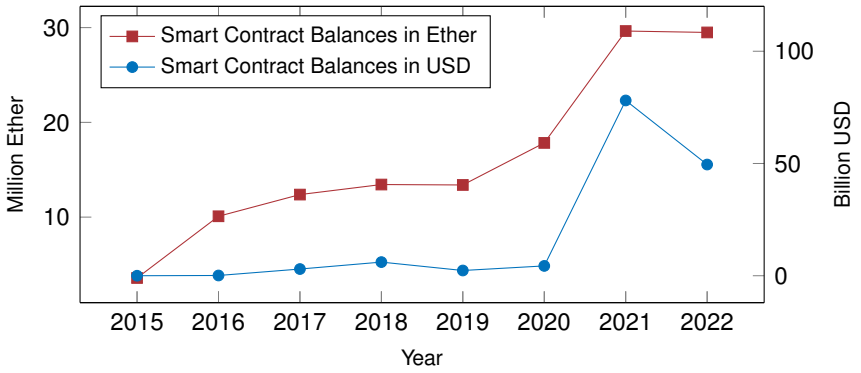


Figure 6. The balance of all smart contracts on the Ethereum Mainnet per year in Ether and USD. The exchange rate between both balances corresponds to the median of all exchange rates of the year [@Eth23].

can be deployed multiple times, we also summed all smart contract deployments with unique EVM byte codes. While the number of deployments increased from 2015 to 2020, it has steadily decreased since then. In contrast, the number of smart contracts with unique byte codes also increased until 2018, temporarily decreased for one year, and then actually reached an all-time high in 2022. In both cases, we see a consistent high demand and popularity for smart contracts on the Ethereum Mainnet. This could be due to the high demand for crypto tokens implemented in smart contracts, and the declining popularity of the native Ether cryptocurrency.

Funds

Similar to the number of deployments, we use the balances of smart contracts to get an overview of their past demand and relevance. For this purpose, we analyzed the balances of all smart contracts per year, as shown in Figure 6. To this end, the balance does not belong to a user's Ethereum account, but is completely managed by a smart contract. To better understand the funds, we also determined the median exchange rate of Ether to USD for each year. While both metrics show a moderate increase from 2015–2020, smart contract balances increased significantly in 2021. In 2022, the value of Ether decreased, and therefore the smart contract balances in USD, but the balances in Ether remained constant. Although the exchange rate of USD/Ether is basically regulated by supply and demand, the introduction of EIP-1559 and the switch from PoW to PoS could be a reason for less interest in Ether. Therefore, we provide a more detailed analysis of the exchange rate in the following.

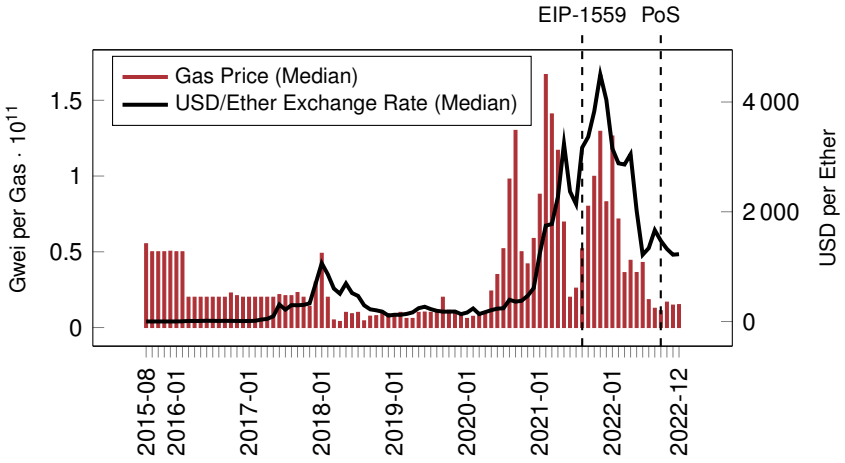


Figure 7. Median Gas price in Gwei per Gas and the median exchange rate from Ether to USD [@Eth23], grouped by month. The dashed lines indicate when the Ethereum Mainnet implemented EIP-1559 and PoS, respectively.

Transaction Fees

Analyzing the fluctuations in Gas prices and USD/Ether exchange rates provides valuable insights into the relevance of Ethereum and the deployed smart contracts over time. Figure 7 therefore shows the median of both metrics per month since Ethereum's launch. While technical adjustments and advancements lead to decreasing Gas prices at first, the interest in Ethereum started to increase by the beginning of 2018. In fact, Ethereum has since made numerous protocol changes to revise Gas costs for various EVM commands [@But16; @Akh+19; @BS20]. Most notably, EIP-1559 [@But+19] changed not only how Gas charges are calculated, but how they are charged to the transaction sender, as described in Chapter 2.3. However, this peak in 2018 is primarily a result of the sudden success of the blockchain game *CryptoKitties* [SSH21].

Since 2020, Ethereum also experienced increased traffic due to various other events. On the one hand, this led to a rise in demand and traffic, resulting in an increase in the USD/Ether exchange rate. On the other hand, the increasing value of Ether made smart contracts more relevant as they managed more valuable assets. Also since 2020, we can see the increasing interest in Defi DApps and tokens. Especially the trend of non-fungible tokens (NFT) brought a lot of attention to Ethereum. However, we can expect this trend to end soon, like other hype-based trends before.

In addition, in 2021, the introduction of EIP-1559 and the shift from PoW to PoS have noticeably affected Gas prices and the value of Ether. Nonetheless, it is important to note that the prices are largely influenced by investment and speculation, and may not accurately reflect technical advancements.

3.2 RELEVANCE AND TRENDS OF BLOCKCHAIN-BASED DAPPS

In conclusion, since its launch in 2015, Ethereum has seen a steady increase in interest, both from a technical perspective and in the value of its cryptocurrency, Ether. In particular, there have been several significant events that have pushed the network to its limits, including CryptoKitties in 2018 [SSH21], Defi since 2020 [BCL21; Ten+22], and NFTs since 2021 [WMP22]. It is important to note that miners have also entered the investment business in an eager attempt to earn Ether through block rewards and maximize transaction fees. Fortunately, as we can see with both metrics in Figure 7, EIP-1559 has prevented this trend from continuing since the end of 2022, and the hype around blockchains can also be considered to have subsided since then.

However, we have not specifically addressed decision-making DApps and their relevance in our analysis. It is important to note that decision-making DApps handle significant amounts of monetary assets as well, although they have not yet caused a significant surge in Ethereum. Therefore, in the following chapter, we are going to focus specifically on analyzing decision-making DApps.

4

DECISION-MAKING DAPPS

This chapter is based on joint work published in [MT21] by Robert Muth and Florian Tschorsch. "Empirical Analysis of On-chain Voting with Smart Contracts." In: Financial Cryptography Workshop on Trusted Smart Contracts. Vol. 12676. Lecture Notes in Computer Science. Springer, 2021, pp. 397–412

In general, smart contracts offer transparency, trust, verifiability, and automated execution, making them an attractive option for implementing blockchain-based voting [KV18; Hei+18; Hja+18] as everyone can verify the correct execution. As a result, decentralized governance DApps have emerged that leverage smart contracts to implement decision-making processes. Against this background, numerous decentralized governance platforms have arisen with decision-making processes, such as proposal-based voting by majority or stakes. In this chapter, we therefore build an analysis pipeline to empirically identify decision-making DApps and investigate the most relevant DApps based on the number of interactions, i.e., votes and their funding.

To that end, we first develop an analysis pipeline to reveal decision-making smart contracts on the Ethereum blockchain and present an overview of key metrics, which emphasize the relevance of on-chain voting. Second, we assess the technical limitations of on-chain voting with a model-based comparison of blockchain specifications as well as an analysis using historic block data. More specifically, we evaluate the DApps voting capabilities in small-scale and large-scale scenarios. We additionally give an outlook on other relevant blockchains with on-chain voting, i.e., Bitcoin and Dash.

Permissionless blockchains like Ethereum [Woo22] inherently allow anyone to verify all blockchain transactions. Therefore, DApps running on Ethereum allow anyone to verify that a vote was stored and counted correctly in a transaction. It has been, however, shown and argued that blockchain-based voting has fundamental issues [Hei+18; Par+21], including security [SKW20; Nat18] and privacy [HHK18] problems. While blockchain-based online voting certainly polarizes, on-chain voting is still being used for reasons such as the decentralized governance of funds. Most prominently, decentralized autonomous organizations, e.g., *The DAO* [@Jen16], allow fundraising and enable stakeholders to manage the distribution of funds with on-chain voting. Smart contracts render the decision-making process transparent and self enforcing. Since its debut in 2016, The DAO raised approximately 150 million USD, but at the same time lost about 60 million USD due to an exploit [ABC17]. While we distance ourselves from the idea of blockchain-based online e-voting, e.g., for official elections, we argue that *on-chain voting* still requires attention and further research.

We show the relevance of on-chain voting and derive limitations in terms of scalability and transaction costs. To this end, we scan the Ethereum Mainnet for smart contracts with voting functionality and

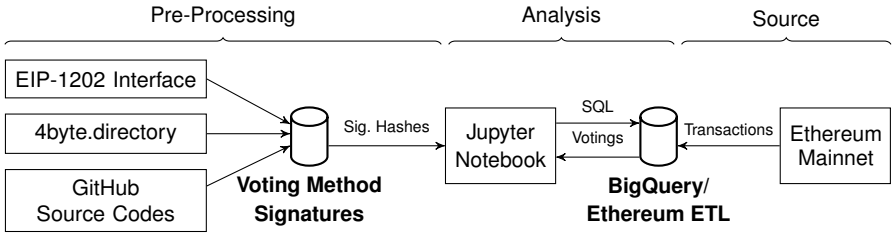


Figure 8. Our blockchain-based voting analysis toolchain with a Jupyter Notebook and BigQuery (or another Ethereum ETL pipeline) based on given pre-processed method signatures.

analyze their usage with respect to submitted votes, Gas costs, and fundings. In order to understand the scalability potential of on-chain voting, we analyze past residual blockchain capacities of Ethereum and evaluate the feasibility of small and large-scale votings. We also look beyond Ethereum and discuss other leading blockchains, including Bitcoin [Nak08] and the governance network of Dash [@DD18]. We provide a publicly available repository with the collected data sets and our analysis pipeline. Our presented database driven analysis approach is compatible with Google BigQuery and therefore does not require any advanced setup.

4.1 EMPIRICAL ANALYSIS

In this section, we reveal the magnitude of on-chain voting in Ethereum. We are particularly interested in the diversity of voting smart contracts with respect to cost and fundings.

Toolchain and Methodology

As described in Chapter 3, there are different technical approaches that can be used to receive historical blockchain data and analyze it in more detail. Typically, analyzing blockchains requires a synchronized node with all valid transactions. With Geth, the Ethereum foundation provides such a node, which has been optimized to save computational resources and memory. As it turns out, the very data-efficient data structures make it difficult to quickly analyze historic data; especially for our smart contract analytics requirements.

For this reason, we use Google BigQuery as a source for Ethereum Mainnet transactions instead. As shown in Figure 8, we use BigQuery as data source and to execute complex SQL queries for analysis. We additionally develop a Jupyter Notebook, which manages the analysis process, i.e., preparing input data from pre-processing, compiling SQL statements, monitoring the execution, and preparing the results.

While smart contracts are generally stored publicly on the Ethereum blockchain, only the compiled bytecode, i.e., EVM code, is available.

Similar to high-level programming languages, the original source code compiles to an assembly-like language. To this end, compilers remove comments and substitute identifiers, which render the bytecode difficult to understand without the original source code. In addition, method signatures of smart contracts, i.e., method name and parameter list, are represented by a hash pointer. More specifically, the first for 4 bytes of a method signature's Keccak (basically SHA-3) hash value are used to point to the respective stack code position. Since Keccak is a cryptographic hash function, it is not possible to infer the method signature from the hash value directly. Hence, it is neither straight forward to search for a certain type of smart contract nor for a partial method signature.

In order to analyze the Ethereum blockchain, we search for hash values of method signatures that are usually part of voting smart contracts. As shown in Figure 8 as part of the pre-processing, we collect the hashed method signatures of the EIP-1202 voting interface, which provides a standardized set of methods for voting. In addition, we use the Ethereum Function Signature Database[8], which provides a list of method signatures and their corresponding hash values based on known smart contract source codes and user submissions. We use the database's RESTful API to search for methods containing 'vote', 'voting', or 'ballot'. As a result, we get a list of method signature and hash value tuples, which are related to voting. We use these tuples to retrieve the smart contracts that actually implement the respective method. Finally, we analyze the source code of the DAO smart contract on GitHub for identifying transactions to the original instance and deployed copies with them same interface methods.

Inevitably, the approach can lead to some positives as well as false negatives. For example, generic method signatures lead to misclassification of some smart contracts, e.g., `setStatus(...)` of the EIP-1202 or `dropVotes(...)`. We also encounter hash collisions that indicate voting methods in a smart contract, but upon closer inspection do not belong to voting. For example, the method signatures `voting_var(address,uint256,int128,int128)` and `totalSupply()` share the same hash value `0x18160ddd` and lead to false-positives. In an attempt of manual inspection, we exclude these instances from our analysis. In order to prioritize precision (over sensitivity), we consider smart contracts that implement at least two method signatures related to voting only. However, since the bytecode in the blockchain remains a black box, we cannot completely exclude false classifications.

The described methodology enables analyses of Ethereum smart contracts in general and can be used to reveal multitudes in the blockchain. We used it to analyze voting smart contracts with respect to scale and Gas cost in general and the interaction with these contracts in particu-

8 https://4byte.directory

	Calls	Hash	Signature	Avg. Gas	Avg. Gas Price
1	80 682	`0x0121b93f`	`vote(uint256)`	210k	2.4 Gwei
2	7 694	`0x15373e3d`	`castVote(uint256,bool)`	87k	42.9 Gwei
3	7 120	`0xb384abef`	`vote(uint256,uint256)`	56k	29.8 Gwei
4	6 420	`0xfc36e15b`	`vote(string)`	449k	3.2 Gwei
5	4 534	`0xddb6e116`	`vote(uint16)`	77k	3.7 Gwei
6	4 283	`0x3850f804`	`castVote(uint256,↩ uint256[],uint256,uint256)`	239k	46.8 Gwei
7	2 934	`0x6cbf9c5e`	`commitVote(uint256,↩ bytes32,uint256,uint256)`	642k	4.1 Gwei
8	2 624	`0x5e8254ea`	`commitVoteOnProposal(↩ bytes32,uint8,bytes32)`	204k	7.0 Gwei
9	2 161	`0x9ef1204c`	`vote(bytes32,uint256)`	283k	9.6 Gwei
10	2 124	`0xcff9293a`	`vote(uint32,uint32)`	107k	12.1 Gwei

Table 1. Top ten voting methods with respect to their number of calls.

lar. We inspected the Ethereum blockchain for the timespan between October 16, 2017 and December 31, 2022. Moreover, we developed a Jupyter Notebook[9] which connects to BigQuery, our own local data records (e.g., historical exchange rates), and other external data sources. A data dump of the following results, the implementation to gather the data set independently, and our full analysis pipeline to reproduce the results is publicly available on GitHub.[10]

Implementation Complexity

In total, we found 1 472 relevant method signatures related to voting, which are implemented in 3 512 smart contracts. Overall, 168 758 086 transactions interacted with these smart contracts and called 140 396 times one of the voting methods. After data cleaning, 3 512 voting smart contracts remained and are subject of the following analysis.

In Table 1, we show the ten most often called voting methods and their average consumed Gas. None of the deployed voting smart contracts implemented EIP-1202 [@ZEX18] completely, but only a subset of its standardized method signatures. While most of the method signatures in Table 1 are not surprising, methods 7 and 8 lead us to expect a *commit-and-reveal* voting scheme, where voters submit their votes cryptographically concealed, e.g., by using a hash function, and reveal their individual votes later with another transaction. Since such a scheme is more complex, it typically requires more Gas.

Method signatures with more than one parameter mostly belong to smart contracts that conduct multiple votings and allow to specify a proposal. For example, most calls with method signature 9 belong to a DAO smart contract that conducts multiple votings, where the `byte32`

9 https://colab.research.google.com/drive/1r9VxRHurM0PaZkf7vNElvvTisAlL0mVC
10 https://github.com/robmuth/blockchain-voting-analysis/tree/dissertation

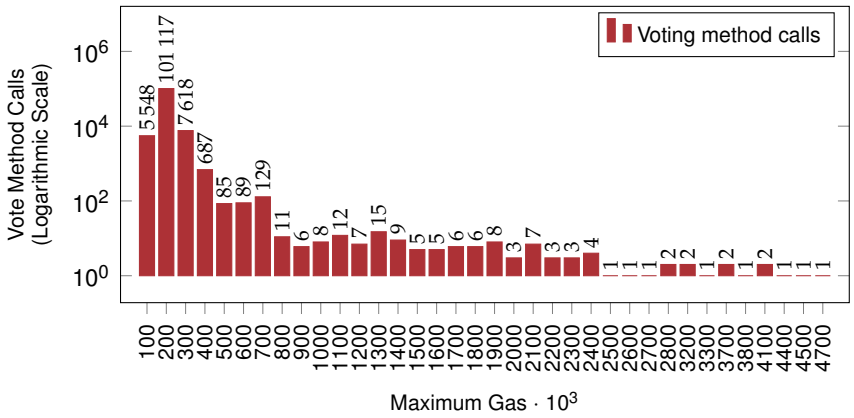


Figure 9. Complexity of voting methods (measured in Gas) in comparison to the number of calls (in total 115 407 calls).

		Funds in Ether	
	Smart Contract	**Received**	**Balance**
1	N/A (Congress Contract)	5 030	4 979 ($ 8 360k)
	`0x3de0c040705d50d62d1c36bde0ccbad20606515a`		
2	HONG / hongcoin	3 936	1 004 ($ 1 686k)
	`0x9fa8fa61a10ff892e4ebceb7f4e0fc684c2ce0a9`		
3	The DAO	12 456 352	43 ($ 72k)
	`0xbb9bc244d798123fde783fcc1c72d3bb8c189413`		
4	Unicorn Token (Congress Contract)	5 924	32 ($ 52k)
	`0xfb6916095ca1df60bb79ce92ce3ea74c37c5d359`		

Table 2. Top four smart contracts with respect to their funds.

parameter references the proposal and the `uint256` parameter encodes the user's choice.

In Figure 9, we compare the complexity of voting methods to the number of method calls. The required Gas (on the x axis) is an indicator of the computational complexity. We grouped Gas values in buckets of $100 \cdot 10^3$ Gas. The consumed Gas ranges from 18 120 Gas to a maximum of 4 590 068 Gas with an average of 84 419 Gas. The figure also shows that most voting method calls consume between 100 000 and 300 000 Gas (mind the log scale).

Acquired Funds

Many smart contracts combine voting with fund management in one way or another. In Table 2, we therefore show the four most used voting smart contracts in terms of their funds. We distinguish between the total funds received and their current balance (as of December 31, 2023). For example, the Unicorn Token uses the Ethereum Foundation DAO Congress contract, which allows members to deposit Ether and submit proposals for funding; other members can then vote on whether the

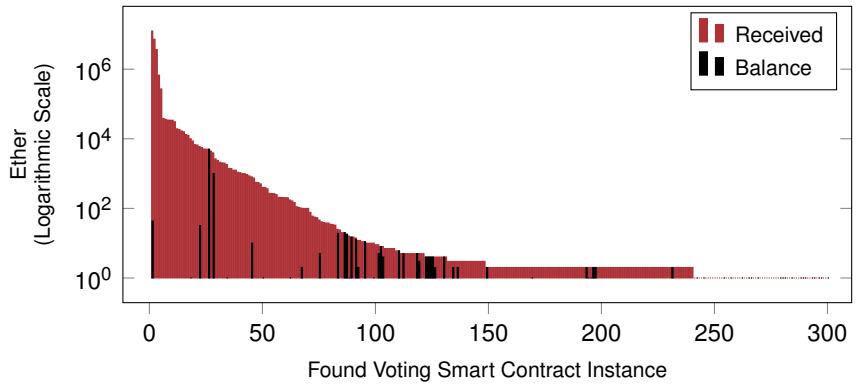


Figure 10. Received and current balance of Ether per voting smart contract, limited to 300 of 3 454 smart contracts in total (as of 2022-12-31).

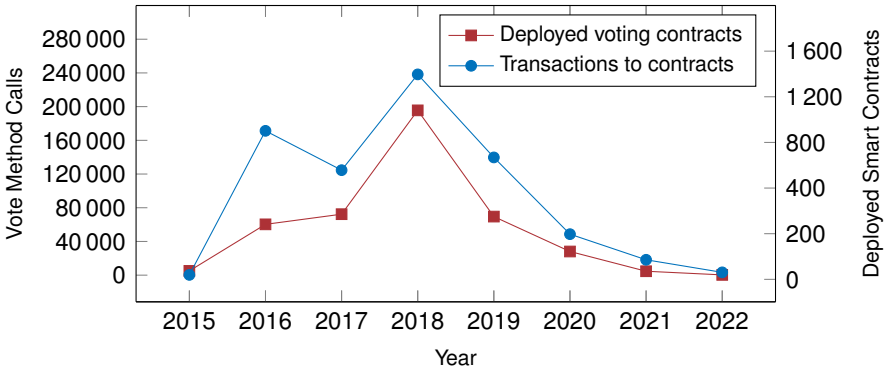


Figure 11. Number of newly deployed voting smart contracts and transactions to them by year (2015-09-06 – 2022-12-31).

proposal is accepted. After the voting period ends and a pre-defined quorum has accepted the proposal, the Ether is automatically transferred to the proposer.

In Figure 10, we show the distribution of funds (limited to 300 of 3 454 voting smart contracts which have received Ether). We can clearly observe a long tail distribution (log scale). However, many of the originally acquired funds are already withdrawn. From all of the funds received, 0.03 % are still deposited. All analyzed voting smart contracts together have a balance of more than 6 307 Ether, which equals approximately 10.59 million USD.[11] The amount of acquired funds can be considered as an indicator for the relevance of on-chain voting.

Trend

In order to get an understanding of the trend, we analyze the transactions as a time series over the last eight years since Ethereum's release in 2015. We particularly focus on the interest and relevance of votings in Ethereum over time.

11 Median exchange rate in 2022 was 1 679 $\frac{\text{USD}}{\text{Ether}}$ [@Eth23]

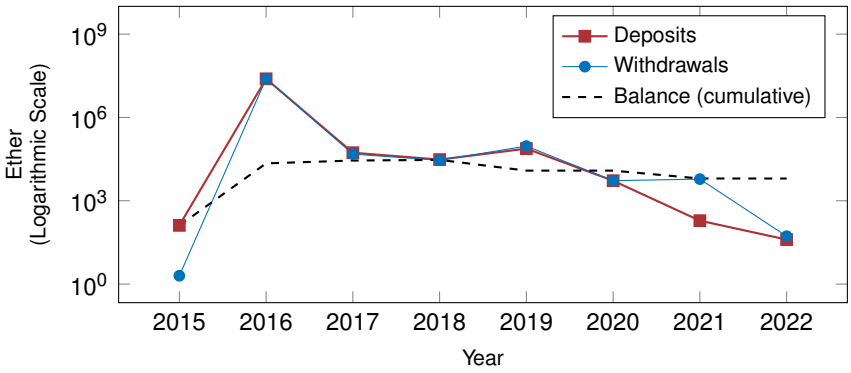


Figure 12. Ether deposits to and withdrawals from voting smart contracts and corresponding balances of voting smart contracts per year (2015-09-06 – 2022-12-31).

In Figure 11, we show the number of voting method calls (left y axis) as well as the number of deployed smart contracts related to voting (right y axis). Once deployed, smart contracts remain active and are not counted again in the following years, i.e., the figure shows the deployment of new smart contracts. In addition, we analyzed the deposits, withdrawals, and corresponding balances of each voting smart contract over time, which are shown in Figure 12.

We generally observe that with the debut of the DAO [@Jen16] in 2016, the number of smart contracts with voting functionalities as well as the number of transactions that interact with voting contracts increases with a peak in 2018. After 2018, we observe a decline of both metrics. While the trend might suggest a decline in interest, the balances remain stable over time. Upon closer examination, comparing Figure 10 and Figure 12, the total balance in 2022 is almost entirely contributed by the top two voting contracts (with more than 1 000 Ether). That is, while the balances were previously spread across many smart contracts, we can infer that the funds are more centralized now. We conclude that the dynamics and interactions of voting smart contracts have decreased over time, but on-chain voting is still relevant in terms of funding.

Summary

In our empirical analysis, we found 3 512 deployed Ethereum smart contracts related to voting, which held 6 243 Ether at the end of 2022, or more than 10.48 million USD (using the median exchange rate of 2022). From these smart contracts, we identified 88 instances of the DAO (deployed smart contracts that are based on the original DAO source code), which in total received 5 929 votes, so far. Over the past years, voting smart contracts in general accumulated and processed 12 094 377 Ether. Our analysis suggests a continuously high amount of monetary investments in and interaction with voting smart contracts, indicating a high popularity and relevance.

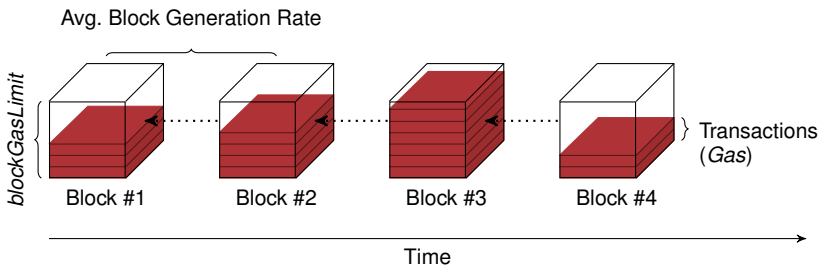


Figure 13. Blockchain partially filled with transactions, leaving residual Gas.

4.2 FEASIBILITY ANALYSIS

In the following, we present a feasibility analysis of on-chain voting. In particular, we analyze scalability limitations using a model-based analysis as well as an empirical analysis based on historical blockchain data.

Block Capacities

One of the central scalability parameters is the maximum number of transactions per block interval, i.e., transaction throughput, which ultimately limits the possible number of votes. Since the switch from PoW to PoS, Ethereum has been aiming for a block generation rate of 12 seconds [@RB20] and a block Gas limit of 15 million Gas, with the flexibility to temporarily max it out to 30 million Gas [@But+19]. The notion of *Gas* was introduced to measure the total computational complexity of transactions. Ethereum accordingly charges transaction fees based on the transaction's complexity. The sender of a transaction sets a price in Ether, which determines the amount they are willing to pay per computational unit, i.e., the *Gas price*.

Depending on the number of transactions per block and their complexity, transactions might not make use of the available block Gas limit and leave *residual Gas*. In Figure 13, we visualize the concept of the Gas consumption and residual Gas. The residual Gas determines the space for additional transactions on top of the baseline activities of Ethereum. Later, we make use of the notion of residual Gas to evaluate the feasibility and scale of on-chain voting.

Model-Based Scalability Analysis

Our analysis is based on overly optimistic model-based assumptions to reveal upper limits, which allow us to make fundamental statements about the (in)feasibility of on-chain voting. To do so, we start with a number of votes μ that we would like to cast. We are then interested in the number of blocks n that are necessary to cast μ votes. Given the block generation rate, we can approximate the time it takes to mine

n blocks, which we denote with Δ. For a block i with a $blockGasLimit(i)$ and a certain *gasCost* per vote, we can calculate the maximum number of votes per block by $blockGasLimit(i)/gasCost$.

Based on our blockchain analysis results from Section 4.1, we evaluate two different scales of voting. Since our measurements show that most voting methods were called between 2k–7k times, we consider $\mu = 2\,000$ to be a small-scale voting, and $\mu = 100\,000$ to represent future large-scale votings. Moreover, we introduce three on-chain voting "schemes", which are either overly simple or taken from our previous analysis. Please note that these simple voting schemes are not meant to facilitate general voting principles, e.g., anonymity and secrecy.

The *naive voting* provides different addresses, each representing a voting option. Voters can transfer coins to the respective address until the voting ends, where the balances determine the final voting result. This naive approach can basically be implemented in every cryptocurrency. In Ethereum, the Gas costs are 21 000 Gas.

The *minimal voting* uses a smart contract for counting votes. To this end, we implemented a synthetic voting smart contract that only consists of a single method for counting votes (available in our GitHub repository). We are aware, though, that the Solidity compiler does not generate perfectly optimized bytecode. While an optimized voting smart contract with a completely assembly-style built bytecode would need less Gas, we consider the Solidity compiler the most prevalent way to compile smart contract code. After deployment, the minimal voting requires at least 41 897 Gas per method call.

For the purpose of more realistic statements, we also analyzed the median Gas costs of votes to *the DAO*. To this end, we used our analysis pipeline described in the previous section, which yields 150k Gas per DAO voting call. As expected, this is more complex than our minimal voting as it also manages funds and quorum regulations.

In Table 3, we show the minimum duration of small-scale and large-scale votings for the various voting schemes (see "Model" columns). For Ethereum, we assumed a block generation rate of 12 seconds [@RB20] and two different block Gas limits w.r.t. EIP-1559: $12 \cdot 10^6$ Gas and $30 \cdot 10^6$ Gas. For comparability, we also included the naive voting for Bitcoin and Dash, which we will discuss later in Section 4.3. Based on this initial evaluation, we can expect that small-scale on-chain voting is generally feasible in reasonable bounds. At the same time, with 12 million Gas, large-scale votings require under idealistic circumstances more than six hours for the naïve voting scheme, or even about 51 h for the DAO voting smart contract. With 30 million Gas, it takes about 30 minutes and 4 h, respectively.

	Implementation	Gas	Blocks n			Duration Δ		
		Limit	Model	Median	MAD	Model	Median	MAD
Small-scale	Ethereum Naïve	12M	4	21	12	00:01	00:05	00:03
	Ethereum Minimal Voting	12M	7	48	30	00:01	00:13	00:09
	Ethereum *The DAO*	12M	25	219	142	00:05	00:51	00:35
	Ethereum Naïve	30M	2	2	0	00:01	00:01	00:00
	Ethereum Minimal Voting	30M	3	6	1	00:01	00:01	00:01
	Ethereum *The DAO*	30M	10	20	1	00:02	00:04	00:01
	Bitcoin Naïve	—	1	4	3	00:10	00:45	00:45
	Dash	—	1	1	0	00:03	00:00	00:00
Large-scale	Ethereum Naïve	12M	175	1 533	1 042	00:35	06:23	04:29
	Ethereum Minimal Voting	12M	350	3 447	2 389	01:10	14:08	09:58
	Ethereum *The DAO*	12M	1 250	13 192	9 105	04:10	50:45	34:43
	Ethereum Naïve	30M	70	141	2	00:14	00:29	00:02
	Ethereum Minimal Voting	30M	140	285	3	00:28	01:02	00:04
	Ethereum *The DAO*	30M	500	1025	3	01:40	03:49	00:07
	Bitcoin Naïve	—	3	41	36	00:30	06:30	05:51
	Dash	—	10	10	0	00:25	00:21	00:05

Table 3. Required blocks n and duration Δ [HH:MM] for different voting implementations; median and median absolute deviation (MAD) are based on residual block capacities (monthly intervals between 2015-12-28 and 2022-12-31).

Residual Capacities Analysis

In the following, we enrich our model-based evaluation with historic blockchain data to determine the residual Gas limits in Ethereum. This approach provides a more realistic assessment of limitations. More specifically, we define $residualGas(i) = blockGasLimit(i) - usedGas(i)$ for a block i. Please note that in Ethereum the block Gas limit is block specific and changed over time before the London fork (c.f. EIP-1559 [@But+19]). Since after, the maximum block Gas limit is fixed to $30 \cdot 10^6$ Gas, however, the protocol's aim is to target $15 \cdot 10^6$ Gas. The residual Gas is therefore determined by the used Gas at a certain point in time.

In Table 3, we show the median number of blocks n as well as the duration Δ for historical data in addition to our model-based evaluation. We calculated n and Δ starting with the last mined block of 2022-12-31 and repeated the process for each preceding month until the genesis block of Ethereum (2015-07-30). In general, our measurements yield values under the (unlikely) condition that all voters submit their votes in a perfectly aligned and coordinated order. We use this approach to provide an (optimistic) understanding for the minimum Gas needed to deploy and cast a single vote. Since we repeated the evaluation multiple times by shifting starting points in monthly intervals, we present the median absolute deviation (MAD).

The results show that simple small-scale and large-scale voting yield reasonable performance with approx. 1–13 min or less for 2k votes, and between 32 min–14 h for 100k votes (w.r.t. the different Gas limits due to EIP-1559). The exception is the more complex DAO implementation, which takes more than 4–50 hours (w.r.t. the Gas limits).

Economic Analysis

Since Gas cost can be directly translated to Ether, we can also estimate the economic efficiency of on-chain voting. As a first impression, we consider a median Gas price 2.4 $\frac{\text{Gwei}}{\text{Gas}}$ (SD $= 55.78$) for the 137 435 voting method calls from our data set. We used an exchange rate of 1 679.00 USD per Ether as before. Hence, we can approximate the price of a vote for our minimal voting scheme that approximately yields 0.17 USD per vote. For more realistic Gas cost, i.e., the most called voting methods require between 100–200k Gas, our price approximation ranges between 0.40 USD and 0.81 USD per vote.

Voting costs are a relevant factor for high reachability and inclusive participation. While fees for casting a vote might serve as Sybil protection, they might also deter voters. In general, fees set a higher participation threshold. In order to maximize participation, transaction costs should be as low as possible for submitting votes—or just not be charged, at all. Unfortunately, smart contracts in Ethereum are not able to pay the transaction fees for the senders, e.g., for calling chosen voting methods. It is possible to implement smart contracts that refund transaction fees within the same transaction, but it still requires voters to own initial Ether for paying the transaction fee in advance. Voters who do not own any Ether hence face a greater hurdle to participate.

Interestingly enough, we want to point out an approach that is able to store and release Gas to cover some of the Gas costs itself. Projects like the GasToken[12] exploit Gas reserving opcodes (i.e., `SSTORE` and `CREATE`/`SELFDESTRUCT`) for saving Gas when the Gas price is low and releasing it when Gas is more expensive. Unfortunately, releasing reserved Gas requires Gas itself. That is, the transaction costs can be reduced but not covered completely, which leaves us back to the original problem that voters need an initial Ether fund. For enabling future-oriented use cases that require broad involvement, e.g., participatory budgeting or crowd funding, we believe new solutions are required to open on-chain voting.

4.3 VOTING BEYOND ETHEREUM

In the following, we consider other well-established cryptocurrencies, namely Bitcoin [Nak08] and Dash [@DD18], that can also be used for voting.

12 https://github.com/projectchicago/gastoken (Accessed: 2023-06-07)

Bitcoin

There are several proposals for Bitcoin-based voting [Bis+17; ZC15; TFH17]. Unfortunately, due do the lack of a full-fledged scripting language, Bitcoin heavily relies on external infrastructure to conduct votings, making it difficult to inspect the blockchain and reliably extracting information with respect to voting. While we have found indications for on-chain voting, infrastructures have been shut down and therefore prevent analysis. However, it is worth mentioning that Bitcoin miners implement voting functionality for governance directly in the blockchain protocol to agree on improvement proposals [@Wui+21].

We can however assume that voting would have at least the same transaction requirements (w.r.t. transaction size and cost) as transferring coins. On this basis, we analyze residual transaction capacities of past blocks and derive the maximum of possible votes over that time span. To this end, we need to consider the specifics and changes of the *segregated witness* proposal [@LLW15], which tackles signature malleability issues and therefore separates signature data from the transaction's hashes. As a result, the maximum block size is then limited by the notion of *block weight*, i.e., $\textit{block weight} < 4 \cdot 10^6$, which corresponds approximately to a block size of 4 MB. A standard Bitcoin transaction for transferring coins from one address to another (P2WSH) with segregated witness requires a block weight of approximately 110 (median over all corresponding transactions until December 2022 with a standard deviation of 0.070). Other parameters include a target block generation rate of 10 minutes.

Evaluation

We analyzed Bitcoin for small-scale and large-scale scenarios with minimal transaction weights, which corresponds to our naive voting implementation. In addition to a model-based evaluation, we also investigated the residual block capacities. Table 3 shows the minimum amount of blocks as well as the time span it would take to cast μ votes. We assumed a transaction weight of 110 per vote. While Ethereum requires at least 1–5 minutes (w.r.t. the Gas limit), Bitcoin requires 45 minutes for small-scale votings. Please note that Bitcoin indicates very high MAD for n and Δ. Hence, Bitcoin's residual capacities fluctuate significantly compared to Ethereum, which makes it more difficult to make predictions. For large-scale voting, Bitcoin requires significantly less blocks (due to the larger block size) and despite its slower block generation rate is faster than Ethereum.

Dash Governance Platform Analysis

Dash [@DD18] was released in 2014, initially named *Xcoin* and later *Darkcoin*. Dash does not support smart contracts in the same way as

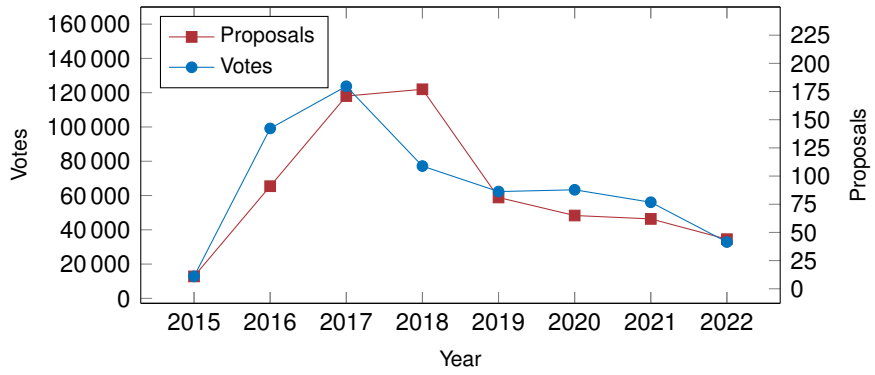


Figure 14. Dash governance proposals and votes, by year (2015-08-27–2022-12-31).

Ethereum, but instead implements its own governance mechanisms directly into its protocols. During the mining process, new coins will be split and distributed over three stakeholders: master nodes and miners each receive 40 %, and the remaining 20 % go to Dash's Decentralized Governance by Blockchain (DGBB) funding platform. Master nodes then can vote on public proposals for distributing the collected funds.

The Dash Governance Platform (DGP) is natively implemented in Dash's application protocols and can therefore be monitored by all nodes that have joined the network (also at DashCentral[13]). After a pre-defined voting phase, the number of yes-votes minus the no-votes must exceed 10 % of the total number of master nodes for a proposal to pass. Otherwise, the proposal will be rejected.

Evaluation

We analyzed 702 proposals between 2015-08-27 and 2022-12-31. During that time, 532 proposals were funded. In Figure 14, we show the total number of votes and the number of proposals per year. Dash's governance proposals started at the same year as the first voting smart contracts with Ethereum in 2015. Dash shows an increase and peak of newly created proposals and votings between 2016 and 2018, and similar to Ethereum, a steady decrease of interest afterwards. Dash's number of proposals at the peak is approximately 8 times lower compared to Ethereum (c.f. Figure 11). Note that the analysis of Dash is more precise and does not suppress any false-positives, which means that the difference to Ethereum is probably even higher. The number of votes at peak times is approximately 1–3 times smaller, when compared to Ethereum. All successful proposals collected 181 767 DASH, which equals approximately 20.90 million USD according to the corresponding exchange rates at the time of funding.[14] Even though the presented votings were not conducted on-chain, the blockchain's protocol auto-

13 https://dashcentral.org
14 https://coinmarketcap.com/currencies/dash/historical-data (Accessed: 2023-04-24)

matically pays out fundings with Dash's cryptocurrency and therefore supports the role of on-chain voting.

Additionally, we evaluated the residual capacities of Dash. While Dash is based on Bitcoin, it does not support segregated witness and aims for a block generation rate of 2.5 minutes with a maximum block size of 2 MB. As shown in Table 3, Dash has a lower transaction load than Bitcoin or Ethereum, which directly leads to high residual capacities and therefore better performance for small-scale and large-scale voting. Our measurements show even better results than our model approximation, because the proof-of-work consensus generated new blocks faster than expected. We would nevertheless expect higher durations with the same general load, i.e., residual capacity, as in Bitcoin.

4.4 RELEVANCE OF DECISION-MAKING DAPPS

We have shown that decision-making DApps have become a relevant use case in Ethereum, most often to collectively manage funds. To this end, we presented our blockchain analysis toolchain, that we used to identify and analyze voting smart contracts with respect to their popularity, complexity, and funds. On the one hand, our benchmark of transactions to voting smart contracts and their respective fundings confirm a high relevance. On the other hand, we observed a trend of centralization due to the popularity of DAO contracts.

We further used these insights to assess the feasibility of future large-scale voting on blockchains. Therefore, we also evaluated other well-established blockchains, i.e., Bitcoin and Dash. While small-scale voting scenarios seem feasible on all analyzed blockchains, large-scale voting suffers from severe scalability issues. Although our model-based calculations indicate that large-scale votings can theoretically be conducted in reasonable time under perfect conditions, our measurements on well-established public blockchains show that minimum durations increase significantly due to the limited transaction throughput.

Despite all the shortcomings of blockchain-based voting, we have shown that on-chain voting is especially relevant for decision-making use cases.

Part II

DECISION-MAKING FOR URBAN PARTICIPATION

5

BBBLOCKCHAIN

This chapter is based on joint work published in [Mut+19] by Robert Muth, Kerstin Eisenhut, Jochen Rabe, and Florian Tschorsch. "BBBlockchain: Blockchain-Based Participation in Urban Development." In: International Conference on eScience. *IEEE, 2019, pp. 321–330, and in [Mut+22b] by Robert Muth, Beatrice Ietto, Kerstin Eisenhut, Jochen Rabe, and Florian Tschorsch. "Lessons Learned: Transparency in Urban Participation Utilizing Blockchains." In:* Eurasian Studies in Business and Economics. *In publication. Springer, 2022*

In this chapter, we present *BBBlockchain*, a decision-making DApp for urban participation. The aim of this DApp is to improve transparency, trust, and participation in urban planning processes. To this end, we leverage blockchain technologies for urban participation projects to prevent monopolistic control over information, and public decision-making. We therefore identify and develop several blockchain-based use cases to complement current participation processes. The use cases include secure timestamping and document management, open discussion and social media integration, surveys and voting, and the integration of blockchain tokens. With these use cases, we address the different levels of participation [Arn69; @IAP14], ranging from information and consultation to decision-making and empowerment of citizens.

We therefore build BBBlockchain as an Ethereum DApp. We opt for a permissionless design approach to make decision-making processes transparent, accountable, and trusted. The deployed smart contracts instantiate our use cases and allow users to interact directly with their functions, e.g., to submit a vote. For user convenience and overall better usability, we implement a web-based frontend so that a wide range of non-technical experienced to blockchain experts can use the DApp. In particular, we are looking for solutions to make the DApp available on as many devices as possible, such as desktop computers and mobile devices.

During the design phase of the BBBlockchain use cases, we identify suitable blockchain features to improve participation processes. As the complexity of blockchain features grows, so does the complexity of the implementation. Hence, we are identifying potential technical barriers that cause difficulties for non-blockchain experts to use the BBBlockchain DApp as well. On the one hand, we therefore develop visualizations and user interface concepts so that the DApp can be used by as many people as possible. On the other hand, we take measures to minimize the technical requirements for the users. For example, we cannot expect BBBlockchain users to host their own wallet to participate. However, to preserve the principles and fundamentals of blockchains, it must remain possible to participate with a personal wallet, if one already posesses one.

In this thesis, we envision BBBlockchain as a DApp to explore the current state of the art, gain experience deploying the DApp in a non-technical users' environment, and gain insights at a fundamental level. As part of our research project, BBBlockchain is being deployed in two

real-world pilot projects in Berlin, Germany. We therefore implement the use cases for informing and consulting citizens. With the deployment, we analyze the DApp development phases and actual usage to identify missing blockchain capabilities related to on-chain abilities and privacy. In particular, establishing secure communication channels to enable communication between participants, anonymous voting, and anonymous authentication for reliable voting.

We first introduce the basics of civic participation processes and the role of civic technology in urban participation in Section 5.1. Next, in Section 5.2, we explore the potential of a blockchain-based participation and propose various blockchain use cases. In Section 5.3, we introduce the initial BBBlockchain DApp architecture and infrastructure for the pilot deployment. Finally, in Section 5.4, we summarize our findings for the BBBlockchain deployment and identify technical limits to improve decision-making DApps in general.

5.1 CITIZEN PARTICIPATION PROCESSES

Motivation

In the context of urban planning, citizen participation processes have always been considered an important element to improve democracy, and in several countries, including Germany, they are enforced by legislation [Amn06]. Today, citizen participation has become institutionalized and implemented through structured methods as part of the organizational logic of government [MK12]. In practice, however, planning processes are mostly influenced by corporate and political actors whose interests differ greatly from the democratic values of participatory planning [Arn69]. Mistrust and the image of corruption have therefore become the default perception of citizens of urban development projects [WTC19].

The lack of trust among stakeholders has led to the need for more transparency [PA19; GW12]. Historically, transparency has been difficult to achieve because of the indecisiveness of public officials, the lack of clear mechanisms for establishing transparency, and the costs involved [BC11; GM13]. When government procedures, policies, and plans are made transparent, citizens can more easily detect improper behavior and government officials can be held accountable for their actions [WD13]. Historically, public participation in urban planning has taken many forms, including dialogue meetings, opinion surveys, panels, consultations, open labs, and so on [Whi96]. While these methods can be fruitful for gathering public opinion, previous research has shown that they suffer from limited transparency, as they tend to limit the expression of conflicting opinions, hide power imbalances, and maintain the status quo [Whi96; AH12].

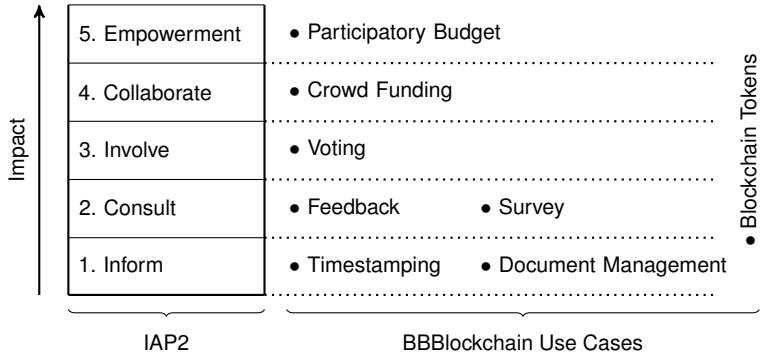


Figure 15. Levels of participation according to the IAP2 spectrum of public participation compared to our use cases.

Digitalization alone offers new opportunities to transform urban participation processes. There is a growing number of so-called *civic tech* platforms that provide online participation tools and attempt to improve cooperation between citizens and government institutions. However, previous studies have shown that current online tools have processes that are similar to more established participation processes [KL14; Ran+19]. Therefore, accountable outcomes are still lacking.

Urban planning therefore requires citizen participation and trustworthy decision-making processes, especially as cities undergo rapid change, such as accelerating urbanization, demographic and climate change, and the increasing digitalization of cities. Collaboration on a trusted decision-making platform is essential to ensure participation and inclusivity. Yet existing formal participation processes, while valuable, often lack trust. We therefore envision a DApp for civic participation processes that can mediate between stakeholders and overcome the underlying problems, leveraging blockchain technologies. To this end, we draw on well-established research in the field of civic participation, such as the ladder of participation [Arn69].

Levels of Participation

Civic participation is important in urban development, but the level and quality of participation varies. Nevertheless, civic participation has gained significant importance in urban development over the past decade. In 1969, Arnstein introduced the ladder of participation [Arn69], arguing for increased involvement of the civil society in decision-making, so that participation is meaningful and produces real impact beyond mere pro-forma processes controlled by planners. Since then, the ladder of participation [Arn69] has remained a valid benchmark for meaningful civil society involvement in decision-making. In particular, participation is key to the successful implementation of development plans, as this depends on the degree to which citizens ac-

cept the plans [Ryd11]. This means that the successful implementation of development plans depends on the acceptance of the plans by the citizens.

In addition, the IAP2 spectrum [@IAP14], an international standard for participation processes, outlines similar levels of public participation in decision-making. It starts with one-way communication, then moves to public consultation, stakeholder dialogue, government-citizen collaboration, and finally public empowerment. Our research also builds on this model.

Transparency and Accountability

Transparency and accountability are essential for successful urban participation. Citizens need to know and trust the information provided about changes, underlying causes, and future developments of ongoing construction projects. In Germany, the right to information is a democratic right [Bun17], but transparency and accountability depend on relationships between stakeholders. The success of these relationships depends on how well these two concepts are achieved. Many cities currently aim to capture and channel these relationships in participation guidelines [Hab15; @Deg+17].

Decision-Making and Empowerment

The basic concept of participation is the balance and distribution of power. Public participation in urban development is essential for citizens to shape their living environment, as well. Currently, power is often delegated to elected officials, but some cities are promoting direct democracy through innovations like participatory budgeting [Cab17]. Digital technologies, like blockchain, offer new alternatives for decision-making and promote transparency and accountability. In particular, and one of the key functions of a blockchain is to act as a neutral intermediary. Thus, the full potential of this technology lies in providing new alternatives for decision-making.

Civic Technology

The concept of using digital tools for local participatory processes has been around for some time and is commonly referred to as *civic technology*. Recent literature supports the idea of civic technology as a movement, with citizens driving its development for technology-based participation [Rum15; @Hen+16]. It combines a startup mindset with civic participation, leveraging the wave of technology startups disrupting various industries. However, civic technologies have a focus on social impact that goes beyond profit-driven motivation. Their ultimate goal is to change the way our society interacts, ultimately reinventing

current methods of governance. More specifically, civic technology softens boundaries by enabling "government from the outside" [DMJ18].

Researchers have emphasized the importance of active citizenship and involvement in decision-making processes through participatory democracy and discourse [Bar03; Dry02]. Civic technology studies have shown that digital planning has the potential to remove barriers and create accessible and ongoing engagement processes [WTC19]. For example, there are related platforms, most notably Adhocracy [@Liq23], that map participatory processes in an online format.

Most digital tools for urban planning participation follow traditional (centralized) formats, ignoring the potential of technology to simplify the process. The authors of [Sal+19] on civic technology reviewed 35 case studies and discovered that communication was primarily one-way. Utilizing digital tools, yet, does not always result in accountable decision-making. The use of digital tools does not always lead to accountable decision-making. The transformative power of civic technology is inherent because it is primarily bottom-up, however, government commitment is still crucial for successful digital participation [Sal+19]. Furthermore, citizens are more motivated to participate when they feel their input will have an impact. Unfortunately, current online participation platforms are centrally controlled, leading to trust and transparency issues.

Blockchain-Based Civic Technology

More recently, a growing body of research has argued that blockchain can provide new opportunities to make government more transparent, especially in situations where government decisions are likely to be lengthy and controversial [Zhe+18; Swa18]. Blockchains ensure data integrity and make stored data immutable, resulting in the ability to track changes, which in turn supports initiatives to prevent (secret) data manipulation. The permanence of data records also allows citizens to better understand and monitor government decisions [Cen+21].

One platform employing blockchain technology is Social Coin [@The20], which has been implemented in Barcelona, Spain. The project explores ways to use blockchain tokens to incentivize citizens' participation by handing out tokens that can be used for payment in local shops. Another example is MiVote [@MiV18], a US-based voting platform which is also based on blockchain technology. Instead of voting for a party or representatives, MiVote aims for voting on individual issues. While our concept can be compared to other civic technology applications, the use of blockchain (and its principles) is the key differentiator. Similar to the Social Coin project, tokens are explored as an incentive, but the scope of BBBlockchain goes further and includes functionalities like voting and surveys, as well as providing a newsfeed for increased trust and transparency. Unlike

MiVote, BBBlockchain aims to test the impact on direct democracy by exploring how decisions can be distributed between authorities and citizens to achieve more representative levels of participation, both in terms of quantity and diversity of participants.

5.2 BBBLOCKCHAIN USE CASES FOR URBAN PARTICIPATION

The decentralized and immutable nature of a blockchain addresses three issues for our decision-making use case: First, blockchain technology makes it possible to prevent monopolistic control of information, thereby increasing trust and transparency. Second, one of the key functionalities of the blockchain is the ability to act as an trusted intermediary. Lastly, advanced blockchain technologies enable private and anonymous decision-making processes that all stakeholders can trust, i.e., through their verifiability. Blockchain technologies, thus, have the potential to enhance participatory processes that rely on privacy and anonymity for their decision-making processes.

BBBlockchain aims to unite three key stakeholder groups involved in participatory processes: the private sector, the public sector and civil society. That is, in our case, housing associations, governmental institutions, and citizens, respectively. Therefore, our research examines BBBlockchain's ability to maintain ongoing stakeholder engagement and consultation throughout the process. We additionally explore how BBBlockchain can facilitate genuine citizen empowerment through blockchain-based decision-making. Based on the IAP2 spectrum of public participation, we develop a set of *use cases* for the deployment of BBBlockchain in urban development decision-making processes. Figure 15 therefore provides an overview and assigns the use cases to the respective level of participation.

Therefore, we first explore the potential to enhance *information sharing* throughout the often complex and protracted urban development processes. Next, we also explore potential use cases from *consultation* to *empowerment*. In addition, we seek to increase and diversify participation by introducing blockchain tokens that promise to enhance all IAP2-based BBBlockchain use cases.

Timestamping

USE CASE Our first use case takes advantage of a feature inherent to blockchains: it utilizes the immutability combined with the frequent creation of blocks to realize a so-called *timestamping service* [HS91; Nak08].

The use case addresses the first level and most fundamental layer of participation to generate transparency, i.e., to *inform* [@IAP14]. The provision of verified information in long-term and often controversial processes can thus be established with BBBlockchain, which is a major

step towards greater transparency in most urban development projects. Public participation rules may also require the provision of specific information by stakeholders at certain points in the planning process. Citizens and nearby living residents, can verify whether this information was provided on time and whether it has since been manipulated or deleted. This allows all stakeholders to prove or assess whether information was provided in a timely manner and was followed during the planning, permitting and construction process.

Therefore, with BBBlockchain, we develop a platform to continuously track ongoing urban development projects and document the planning and approval process. We do this by managing, archiving, and securing documents such as land use plans, urban development contracts, and general construction information and specifications. To this end, each new publication will be timestamped on the blockchain. This allows all users to reliably track the time of publication, author information, and content of the publication. To do this, BBBlockchain calculates the hash value of the publication and publishes it to all users. With a similar concept as already established in 1991 by Haber and Stornetta [HS91], anyone can now verify if the hash value matches and when this hash value was published. BBBlockchain therefore not only covers the first layer of IAP2 [@IAP14] by informing the public, but also ensures that all announcements are secured with blockchain technologies.

IMPLEMENTATION To reduce on-chain storage costs, instead of storing files (e.g., large PDF files or images) directly on the blockchain, BBBlockchains stores the files on an external storage and secures their cryptographic hash values on the blockchain. BBBlockchain therefore leverages that cryptographic hash values represent arbitrary data as fixed-length numbers. Thus, users can verify the integrity of large documents and files by downloading them from the external storage, calculating the hash values locally, and comparing them to the hash values in the blockchain. If an external file has been subsequently changed or missing, the application will display an error to all BBBlockchain users. For security reasons, the content and the corresponding hash verification are performed locally on the users' devices.

Additionally, we use transaction verification as part of our visualizations to convey a secure transfer of information in an understandable way, as we will show in Section 5.4. Once the participation content has been downloaded, the DApp frontend calculates the corresponding hash value locally and displays it at the top of each article.

Social Networks Integration

USE CASE While it is essential to provide comprehensive and reliable information, it is equally important to *consult* and *involve* citizens

in urban development projects (cf. IAP2 Level 2 and 3) [@IAP14]. To this end, we propose a feedback mechanism. Citizens can submit their feedback, e.g., comment on information provided, participation workshops, or statutory planning information. We record this event as hashed representation in the blockchain but refrain from recording plaintext for costs and legal reasons. This way, we also reduce the opportunity for abuse of BBBlockchain as described in [GMF22; Mat+18].

IMPLEMENTATION We explore the integration of social networks as feedback channels. For this purpose, we use a crawler that searches for hashtags related to BBBlockchain projects. Messages found, e.g., on Twitter, will be hashed and the hash values will be stored on the blockchain. Another approach is to generate unique hashtags on demand for specific topics managed by one of our smart contracts. App users can then open documents and start a discussion about a topic with others on social media, and the crawler can associate the messages with specific topics. Again, to reduce abuse and save on-chain storage costs, we do not record the plaintext. At the same time, we outsource the detection of abuse to the respective social media platforms. Since we are unable to detect botnet-driven manipulation, we rely on the social networks' security mechanisms. Once a social network publishes content to our service, it is timestamped and we can prove that the content was published. The timestamp allows us to ensure that social networks do not censor after the fact. However, intentionally deleted messages (by users) can also be detected by comparing the stored timestamps. In general, a DApp could display a notification when it detects a deleted message. BBBlockchain thus establishes a multi-channel consultation and targeted participation tool. Because urban development projects are very diverse and cannot be easily standardized, BBBlockchain must provide the flexibility to respond to specific communication channels agreed upon by stakeholders from the beginning of a project.

Voting and Surveys

USE CASE For further options, to consult and involve the public, we distinguish between *voting*, which we consider as a form of co-decision-making, and *surveys*, which are a consultation feature. Depending on the impact and the institutionalization of BBBlockchain voting, we advance into the fourth and fifth of the IAP2 levels of participation, *collaborate* and *empowerment*.

For example, housing associations commit to public participation on different development or design options [@Deg+17], including a binding vote for the preferred option. In current practice, these infrequent votings are usually neither legally binding nor sufficiently transparent or private. Even with fully committed stakeholders the effort to conduct reliable votings often proves to be too onerous and difficult. It

often boils down to participation meetings and paper-based voting. As a consequence, the current situation suffers from a lack of inclusiveness and trust issues. With BBBlockchain, we aim to overcome these issues in order to enable a more prolific use of voting in urban participation processes.

IMPLEMENTATION While we envisage different ways to instantiate votings and surveys, the easiest way is to save electoral processes in an array, where each element represents a voting choice and with a public function for increasing the respective counter. However, this approach is vulnerable to fraud as voting multiple times by repeatedly calling that function becomes possible. We therefore need a mechanism to validate and authenticate voting rights, e.g., the voter's identity or a digital certificate of eligibility to vote. To prevent fraud and realize reliable votings, BBBlockchain issues registered voting tokens for each eligible citizen. The right to vote can only be exercised when a valid token is passed with the function call.

In general, though, blockchain-based voting inherently suffers from certain disadvantages [Sim04]. Most notably, the necessary prerequisites for voting, e.g., identity checks, are more difficult to establish and maintain online than in the offline world. For example, it is more difficult to check identities and voting rights electronically than in person at a voting office with personal IDs. This raises a number of challenges for blockchain-based voting and boils down to a tradeoff: On the one hand, blockchain offers unconditional voting transparency and auditability. On the other hand, achieving voter anonymity is challenging, because all votes in a blockchain are fully traceable. As a first step, we use pseudonyms as a first way of achieving privacy. However, contrary to public elections votes, in urban development projects the local reach might allow or demand the public identification of the voters. In [@Fou22a] for instance, video chat or video proofs are used for identification. To be clear, BBBlockchain is not intending to replace offline voting. Aligned to the project's overall motivation to foster the diversity and inclusiveness of public participation in urban development projects, we investigate the pros and cons of blockchain-based voting for participation use cases.

Tokenization

USE CASE One reason for the current popularity of blockchains is the token system. We see tokenization as a cross-cutting feature that can enhance most of the use cases discussed above in one way or another. For example, we use tokens to manage voting rights and to incentivize participation. While there are many possible uses, tokenization is likely to have the greatest impact on the upper levels of participation, since tokens are by definition an instrument of exchange.

We therefore distinguish between *coins* and *tokens* to highlight technical differences and different use cases: Cryptocurrencies issue coins as their inherent trading currency, which typically has monetary value, and use a blockchain to record transfers and balances. In Bitcoin, for example, coins serve as a means of payment, but also to incentivize miners to keep the system running [TS16]. Ethereum [@But23b] and other blockchain-based systems also offer the ability to create and issue custom tokens on top of the infrastructure. These tokens are issued and managed by smart contracts, so they are not built into the protocol of the blockchain. They can follow their own specific rules, but still provide basic functionality such as checking balances or making transfers. For example, Ethereum provides standards for implementing custom tokens [@VB15]. In BBBlockchain, we utilize both coins and tokens: Since coins have monetary value, we use them to manage a participatory budget, for example, an increasingly popular instrument for co-decision-making, which we will discuss in the following Section 5.2. Since tokens, on the other hand, can be decoupled from monetary value, we use them as voting tokens and in various targeted forms to incentivize participation. In the following, we will elaborate on the latter use case.

VOTING AUTHENTICATION We use tokens to authenticate whether a user is eligible to participate and vote. The amount of tokens can indicate the priority of the user, i.e., more tokens give more weight to their vote, but this depends on the voting scheme. In this way, we could give more voting tokens to people who live near a project than to people who live far away. Following the concept of liquid democracy, users could delegate their voting rights to someone else. This functionality allows both higher participation rates (the representative can vote even if the user is not available) and the accumulation of influence with individually legitimized representatives with specific knowledge or user status (e.g., mobility specialist or official tenant representative). While it would also be possible to issue tokens in exchange for real fiat money or coins, we distance ourselves from this approach because incentivizing through monetary value is democratically questionable.

As a side effect, tokens can be used to achieve anonymity during voting: For example, tokens can be issued as QR codes, shuffled, and sent to residents. The shuffling of QR codes results in untraceable pseudonyms. While this seems like a reasonable approach to achieve some level of anonymity in practice, we prefer cryptographically secured techniques over analog ones.

INCENTIVIZATION Tokens offer a wide variety of ways to provide incentives for participation. For example, the use of the BBBlockchain DApp can be rewarded with tokens that can be collected and used for discounts. As a reward system, we provide an interface for shop

owners to offer services or discounts in exchange for tokens. For example, cafes near the building project location could offer a fixed price discount to attract new customers. More importantly, the targeted provision of tokens can attract citizens to support analogous on-site participation formats such as permanent exhibitions or workshops in neighborhood cafes. To further increase interest or site traffic we can also distribute tokens throughout the neighborhood or at institutions of particular of particular interest to the project, e.g., via printed QR codes (municipalities, developers, showrooms, etc.). They could also be embedded in relevant documents so that interested parties are rewarded for participating in the process.

Participatory Budget and Crowdfunding

USE CASE DApps can be used to implement crowdfunding. Crowdfunding is a type of online fundraising. Projects can collect small donations from a large number of people. In an urban context, crowdfunding can serve as an alternative way to (co-)finance smaller projects for the neighborhood. The provision of funds by many can be seen as a public vote to realize certain projects, and similar to BBBlockchain's functionality to enable participatory budgeting, the smart contract can enforce execution once the funding goal is reached. From a technical perspective, many variations are possible. An alternative funding method would be crowdsourced participatory budgets. We can also implement the concept of matching funds, where a certain amount of crowdsourced coins is matched by other sources, such as a housing authority.

The transfer of decision-making power in urban development from the institutions that have been elected to represent the public good directly to the citizens obviously challenges the current balance between direct and representative democracy. BBBlockchain enables functionalities previously unavailable or infeasible in an urban development process, and this project will need to explore and redefine the boundary conditions that support both citizen empowerment and the common good.

IMPLEMENTATION BBBlockchain can conduct votes using smart contracts with binding results. It can also enforce the contractually agreed results. Participatory budgeting is a powerful form of participation that benefits from this functionality. Current analog formats reserve a certain portion of the public budget that is centrally managed. The use of this part of the budget within predefined spending corridors is decided directly by citizens through voting. In our use case, we are exploring a participatory budget in coins, bound in a smart contract, outside the control of a central authority. A simple example is the selection of public art in housing projects. Developers reserve the

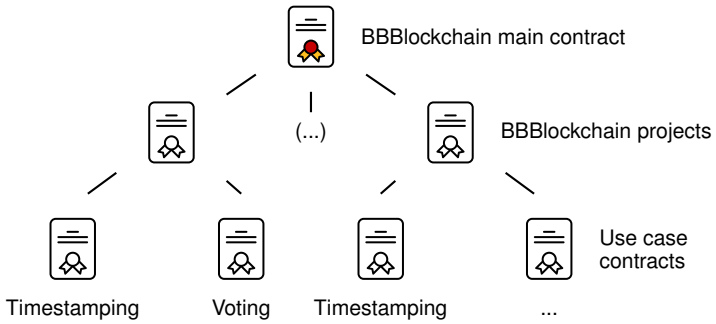


Figure 16. Smart contract architecture of BBBlockchain.

mandatory budget for artwork (in some countries, such as Germany, construction projects are legally required to fund art with a certain amount of the total construction cost) and let residents, neighbors, or a broader group of citizens vote for their preferences. After the vote, the winning artist is contracted and the budget is transferred. A participatory budget, secured by a smart contract, is thus a true transfer of decision-making power to the eligible group of participants. Since the voting results are directly implemented, the authorities or developers can no longer interfere with the citizens' decision.

5.3 BBBLOCKCHAIN ARCHITECTURE

We implement the BBBlockchain DApp with smart contracts in Solidity. We decided to build an Ethereum DApp because it is a public, permissionless, and established blockchain that provides enough flexibility to implement our platform. In the following, we will introduce the BBBlockchain architecture and infrastructure. In addition, we explain our design choices.

BBBlockchain Smart Contracts

The main BBBlockchain smart contract manages multiple other smart contracts for all building projects and handles all permissions. Once deployed, it provides all data for the app and manages multiple building projects and their use cases. As shown in Figure 16, it serves as a central entry point. At this point, we have explicitly decided against a proxy smart contract for future upgrade options to improve trust in this case. Although we are committed to a permissionless platform, not everyone is allowed to freely manage our projects. We distinguish between read-only calls and write transactions in the smart contracts. All data is stored openly for read-only access, but management of the smart contract is restricted to a closed user group representing key stakeholders.

An urban development project may have multiple use cases that are implemented as separate smart contracts. The main smart contract therefore maintains a list of all use case contracts. Currently, we have implemented smart contract solutions for the following use cases: timestamping, surveys, voting, and our own token. As each urban development project has its own participation strategy we have taken a modular approach; each use case can be activated individually, repeatedly or simultaneously. And because these strategies often change over time, new functionality can be developed and added at any time.

Infrastructure

BLOCKCHAIN ACCESS Processing the Ethereum blockchain requires high CPU resources and storage (about 1 TB for syncing Ethereum Mainnet with Geth at the time of writing). Therefore, it is not yet practical to run fully synchronized blockchain clients on mobile devices for normal usage. For the pilot phase, we therefore installed our own Ethereum node and provided access for BBBlockchain users via a public server with an API. However, smart phones and recent blockchain technologies enable direct access to the blockchain with light client implementations, which do not process all transactions but rely on public or self-hosted full synchronized clients. We intend to implement different blockchain access capabilities to eventually replace the BBBlockchain API server. However, the smart contracts support direct access, so it also possible to interact with BBBlockchain without the API.

The participating housing associations, local authorities, and residents' representatives have been given access for publishing contents on BBBlockchain with a fine-grained access control mechanisms. While BBBlockchain is designed to give stakeholders direct access to the smart contracts, we additionally provide a server-based web admin interface for easier usage. We therefore run a content management system, so authors can compose new posts in an intuitive editor interface. New contents will then be added to the blockchain by an external monitoring tool, basically a blockchain oracle.

DAPP ACCESS BBBlockchain relies on the Ethereum blockchain infrastructure and does not necessarily rely on self-hosted infrastructure. However, we do not expect our users to be familiar with maintaining an Ethereum wallet and interacting with a smart contract. Therefore, we have implemented a mobile app for iPhone and Android devices and also provide a website that offers most of the functionalities. We also do not expect users to host a full node, so we provide an API that basically mirrors all smart contract functionality and manages blockchain interactions. The API provides access to our fully synchronized Ethereum node and executes all smart contract calls. The app not only

visualizes the API data, but also verifies hash values locally to check integrity (see Section 5.2). However, users can always execute the smart contract functions themselves without the API, as all BBBlockchain smart contracts are stored on a public blockchain that are open source. Finally, we do not expect users to pay for their transactions, so we cover transaction fees within BBBlockchain. While users can choose to run their own node and verify transactions for full transparency, they then have to pay the transaction fees themselves.

INITIAL ARCHITECTURE Figure 17 gives an overview of the initial technical infrastructure and all implemented technologies for the first pilot projects. Stakeholders at the top can access BBBlockchain with a mobile app and web browser. We therefore host a centralized web server for the graphical frontend and storage, and our own Ethereum node for API access. However, the node and the storage can both be mirrored by anybody for enhancing transparency. Again, our infrastructure is only provided for keeping participation barriers low, especially by not operating full Ethereum nodes on mobile devices. Our hosted Ethereum node can be replaced by other trustworthy parties as the source codes are open-source.

For independence from Ethereum networks the smart contracts are also compatible with Quorum [@Con18], which is a permissioned version of Ethereum for enterprise usage, and other EVM-compatible blockchains. We also want to emphasize our focus on accessibility and inclusivity in our application. It is very important for us to make the documents as accessible as possible. Additionally, we provide a content management system (CMS) for conveniently creating new contents.

With the ongoing development of available tools for Ethereum, new options have emerged to improve the architecture of the BBBlockchain while the pilots were still in progress.

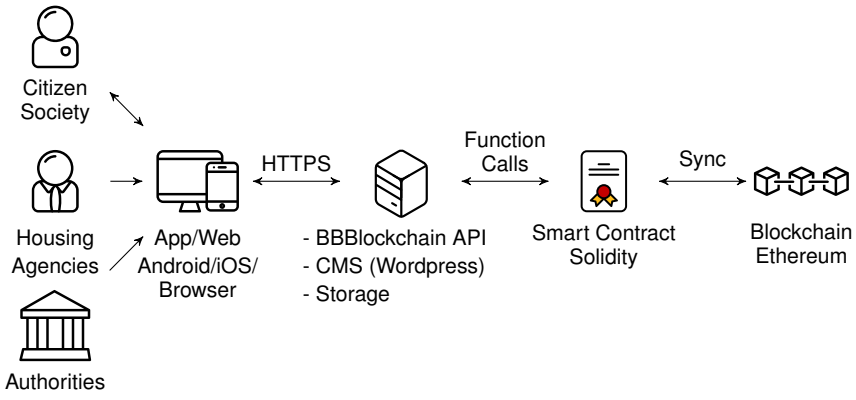


Figure 17. Initial BBBlockchain architecture and technologies with API access.

Public Permissionless Blockchains

As the name already implies, public blockchains can be accessed publicly (whereas private blockchains keep all data confidential) and therefore are the only option for our intentions. When it comes to so-called *permissioned* and *permissionless* blockchains the decision seems not as clear. The pros and cons of both are not obvious and have a direct impact on the transparency and integrity of the BBBlockchain. Basically, the two approaches follow different strategies on who can participate in the consensus algorithm, i.e., who can become a miner (or validator).

Permissionless blockchains generally allow anyone to join the network, become a miner, and help to verify the blockchain. Thus, they require a global consensus between miners and nodes. It is not necessary to assume that nodes trust each other, but that the majority is benign and uses the same consensus protocol. In contrast, permissioned blockchains allow only a selected group of nodes, i.e., so-called validators, to verify and advance the blockchain. It implies an authority that decides on who is allowed to join, and who not. Permissioned blockchains should not be confused with permissions in smart contracts, though. Although a smart contract can limit who interacts with it, e.g., who can participate in a survey, it does not matter whether the blockchain itself is permissioned or permissionless.

In general, the consortium of miners/validators can also mutually agree to change data stored in the blockchain. For BBBlockchain, we opt for a permissionless blockchain, in our case Ethereum, to provide full transparency of all stored data. Since we do not have influence on the consensus algorithm, once published data cannot be manipulated unnoticed; neither by us, housing associations, or governmental authorities, as long as not more than half of the network approves it.

Test Network

For the first pilot deployment, BBBlockchain uses the Ethereum test network Rinkeby, a centralized file storage, a BBBlockchain API server, and an admin interface. We chose to use a test network because it has no monetary exchange value for its cryptocurrency, unlike the Ethereum Mainnet. This allows us to experiment during the initial development phase without financial pressure. We also decided on launching the first pilot phase on the Rinkeby test network, because we cannot control or manipulate the blockchain ourselves, i.e., Rinkeby can be considered permissionless, as anyone can join the network and submit new transactions, except the miners, who are restricted to a closed group. Therefore, it has comparable trust properties to the Mainnet for our use cases. This means that our transactions cannot be manipulated unnoticed by us or any other BBBlockchain stakeholder. While on Rinkeby, we monitor the network for unexpected behavior that would violate our assumptions.

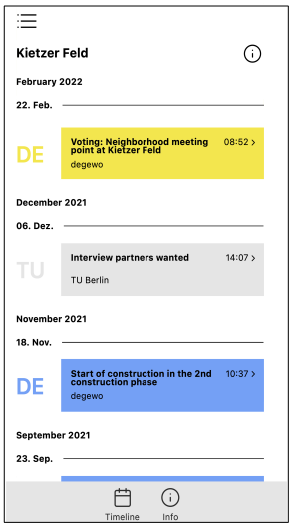


Figure 18. Initial timeline view without any blockchain details.

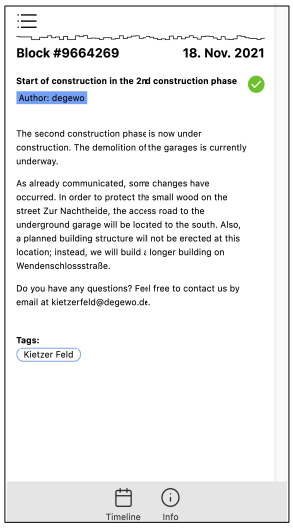


Figure 19. News entry view with blockchain visualizations at the top.

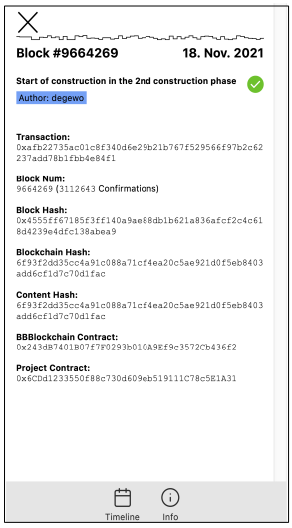


Figure 20. Further blockchain details for the entry.

5.4 PILOT DEPLOYMENT

BBBlockchain launched in 2019 with two pilot projects. Together with the housing associations *degewo* and *Gewobag*, we deployed BBBlockchain as an additional participation component in two real building projects in Berlin, Germany. The pilot therefore focused on the residents living in or near the building projects. In the following we will present our DApp frontend, the use case implementations and our experiences with BBBlockchain.

DApp Frontend

The BBBlockchain DApp frontend provides information in a strictly chronological order to allow users to keep up with the latest updates in an urban development project. The app interface therefore evolves around a so-called timeline view, as shown in Figure 18. Its design was developed in close collaboration with an experienced user interface design studio [@Nol+19]. It is intended to remind users of a calendar app that provides a chronological, color-coded overview of all published entries by participating stakeholders (e.g., news articles or official announcements). Users are initially exposed to as little blockchain details as possible, but can explore all the details if they wish.

With recent posts at the top, the app also follows the concept of a social media feed that lists titles and shortened contents. While the timeline view is the starting point, opening a timeline post shows the corresponding content, including rich media entries (e.g., with images and file attachments), as shown in Figure 19. Users can also inspect blockchain details for each entry's transaction, as shown in Figure 20,

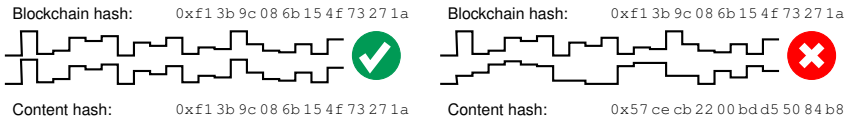


Figure 21. Hash value visualization and comparison.

allowing technically savvy users to verify the content without the BBBlockchain app. For example, the transaction hash can be used to verify blockchain details with an external blockchain explorer.

For a consistent look-and-feel of a mobile app, the navigation between the timeline, posts, and further details, follows the well-established concept of a navigation stack that allows users to go back step-by-step by swiping on their mobile phones (or using the back button). Additionally, the app header always displays navigation information at the same positions and shows the author's name and specific color in further navigation levels as a recognizable orientation marker.

Timestamping

To visualize blockchain functionality, we implement a visualization technique for the key concept of integrity verification, as shown in Figure 21. To do this, we visualize cryptographic hash values for timestamped publications and documents. Such cryptographic hashes represent arbitrary data as numbers. They are mathematically designed as one-way functions, so they are easy to compute, but computationally very hard to reverse, i.e., to find the corresponding data for a given hash value. Blockchains use this concept to verify the integrity of all stored data. Therefore, the top plot line in Figure 21 represents the immutable hash value of the original content on the blockchain when the content was uploaded. The bottom line shows the hash value of the corresponding content downloaded to the user's device. Users can now visually verify whether or not the lines match. In addition, a check mark indicates whether the hash values match or not. A green checkmark confirms the integrity of all data; a red cross appears if the content was manipulated or deleted after it was published. The aim is for users to be immediately aware of the integrity, whether or not they are familiar with the underlying cryptographic concepts.

While this verification could be done hidden in the background of the application's backend, our visualization approach aims to help non-technical users understand the cryptographic concepts of hash verification. Thus, the visualization is not necessary to improve technical security, but helps users gain confidence in the verification process.

Costs Evaluation

Ethereum provides the blockchain technology and an existing broad peer-to-peer network, but using its infrastructure is not completely free of charge. While joining the network, validating transactions, and querying data is free for everyone, persistent interactions will cost flexible fees. Depending on the consumed Gas of a transaction and how many transactions were proposed during the same time span, the fee rises [@But+19]. The fee then has to be paid by the transaction sender before it is executed, stored, and broadcast to other nodes. Other blockchains can handle transactions fees differently, of course. In addition, the Gas limit prevents the code from getting stuck in an infinite loop or from consuming a disproportionate amount of computing resources. If a transaction sender undercharges Gas, the miners will ignore the transaction, or the transaction will fail and the Gas will be wasted.

GAS COSTS We evaluate the Gas costs for various contract types, including timestamping and voting. We also evaluate the total expected costs of deploying BBBlockchain. To this end, we use the median Gas price and Ether exchange price to USD of 2022. I.e., we used the median base fee per Gas of $35.05 \cdot 10^9$ GWei and 1 679 USD per Ether for estimating costs, as before in Chapter 4. We provide a public Git repository[15] with our test contracts that we used to estimate the Gas costs. Its readme file explains how to run the Gas costs evaluation manually. The repository contains three *truffle*-framework projects, each for timestamping, basic voting schemes and voting with ZoKrates. Migrating the contracts with the framework outputs the estimated Gas costs.

TIMESTAMPING COSTS For us and other BBBlockchain hosts it is especially interesting how much providing an API costs. For example, predicting how expensive it is to offer a timestamping service is important for operating the platform: The deployment of the timestamping smart contract costs approximately 2 428 484 Gas, which is about 142.93 USD. Timestamping a data item and adding it to the blockchain, i.e., registering a hash value with our smart contract, costs approximately 214 650 Gas, which is about 12.63 USD per timestamp.

VOTING COSTS As explained in Chapter 2, the transaction fees depend on the complexity of the instructions set and the current blockchain load. In order to get a first overview, we distinguish between different voting techniques. We therefore evaluated the Gas costs of different voting schemes implemented in Solidity, as shown in Figure 22: The minimal implementation allows unlimited voting without any control mechanisms. The address check implementation saves the

15 https://github.com/robmuth/bbblockchain-gaseval

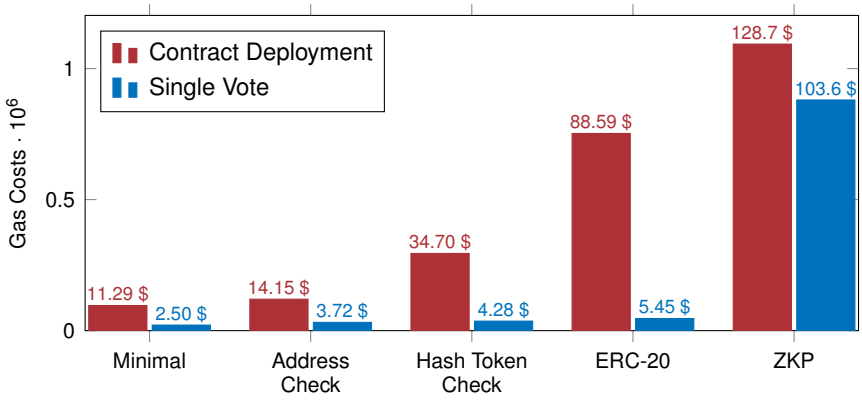


Figure 22. Gas costs for deployment and submitting a single vote using different voting schemes.

sender's address after voting and validates before voting, so one can only vote once per sender address. That leads to slightly higher smart contract deployment and voting fees. The hash token check requires a secret token for voting, so deployment is more expensive because of the hashing function. With ERC-20, we allow to set a weight to a vote and transfer voting rights to other persons, leading to a more complex smart contract. However, voting can become cheaper, because one can vote multiple times for an option with a single transaction for a higher vote weight. At the end we evaluated voting with a zero-knowledge-proof for validating voting tokens and ensuring anonymity. We used ZoKrates [EH18] for generating a validator smart contract and proofs for the voting tokens. Validating voting tokens with complex mathematically proofs become much more expensive.

DEPLOYMENT COSTS Deploying cost for a BBBlockchain project can be estimated as well. The Gas cost is achieved by deploying the main contract (1 281 973 Gas), utility contracts (758 106 Gas), a building project (2 383 215 Gas), and its use cases. The Gas cost depends on the meta data and therefore can only be approximated. Using the same Gas price and Ethereum exchange rate as in Chapter 4, we assume deploying BBBlockchain cost at least 260.34 USD right now, plus the individual Gas costs for the use cases. We conclude that the Gas costs for timestamping and voting as well as the deployment costs of BBBlockchain are reasonable. When compared to traditional participation processes, which for example include providing public participation offices, they even can be considered negligible.

Deployment

For the pilots, we deployed BBBlockchain on the Ethereum Rinkeby test network. Unfortunately, the Rinkeby test network turned out to be not as reliable as we wished for our pilots operation. Since the smart

contract-based framework was developed and tested, we could have deployed BBBlockchain smart contracts to the Mainnet. This however would imply to also reset timestamps. Current users would notice inconsistencies by inspecting transaction timestamps, if we move BBBlockchain from one network to the other. Instead, we decided to fork the Rinkeby test network. Hence, we operate our own blockchain network without losing past transactions, and keeping the timestamps intact. To maintain our trust assumptions, we installed three miners in our own proof-of-authority blockchain, each located at another stakeholder, i.e., university and two building agencies. While we believe that for our research-driven pilot phase, this is an acceptable trade off, we still envision that BBBlockchain operates in a permissionless setting.

Security

We successfully implemented and conducted several votings with BBBlockchain using hash tokens. Technically, the hash tokens are not implemented as ERC-20 [@VB15] tokens. Instead, they are implemented using a revealment scheme that uses random secret keys. Therefore, we generated as many random keys as there are citizens eligible to vote. We cryptographically hashed these random keys, stored the hashes in the voting smart contract, and gave the plaintext keys to the housing associations. Next, the housing authorities gave the list of tokens and addresses of eligible voters to a postal printing service. The printing service randomly assigned the tokens and mailed them to the voters. Finally, the voters could prove that they were eligible to vote by revealing the cleartext to the smart contract. Now, neither we nor the housing authorities know who received which token. So we implement anonymous voting with hash tokens by making it impossible to track who voted and how, unless stakeholders conspire and cooperate.

While this approach worked well during the BBBlockchain pilot deployment, we discovered a serious technical problem with it. Blockchains do not guarantee that a submitted transaction will be appended to the next block. Therefore, new transactions can remain in the so-called transactions *mempool* until a miner appends them to the next block. Miners and other blockchain nodes can technically analyze these pending transactions and thus learn the cleartext of the hash tokens. Although this approach may not always be effective, if an attacker generates a new transaction with higher transaction fees, it could potentially lead to something similar to a front-running attack [EMC20] on the transaction.

We have not seen such an attack while running BBBlockchain, however, we are looking for more secure techniques for anonymous voting. In particular, for completely trustless solutions, where even stakeholders could collaborate but anonymity is not compromised.

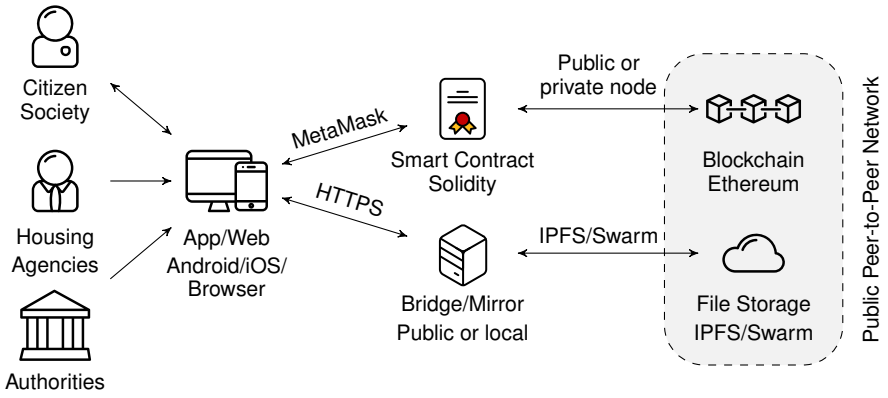


Figure 23. Future BBBlockchain architecture based on development insights after the pilot phase.

Future Architecture

After deploying BBBlockchain, we identified several potential technical improvements and technological advancements that led us to propose an improved architecture. With three years of operation, we propose several adjustments to the architecture, as shown in Figure 23. First and foremost, we experienced difficulties with our self-hosted infrastructures multiple times. Since we decided to use a CMS to create new contents (see infrastructures in Chapter 5.3), its backend software required several updates during the pilot phase. These updates caused several problems by changing the way new and existing content was rendered by the web browsers. BBBlockchain then detected (supposedly) manipulations in the contents where no changes had been made. We also experienced outages in our infrastructures, e.g., due to increased load caused by media attention and attempted attacks.

To that end, we decided to get rid of the centralized CMS. Therefore, all participants still access the BBBlockchain through a mobile app or a web browser. But in the same way as most DApps currently interact, the frontend receives data and controls the smart contracts via a local wallet. We therefore propose that BBBlockchain uses either a light client or a local wallet with an external node. For example, on Apple iOS, an app can directly access Ethereum using the *web3swift* library [@Vla23] with a local light-client or Infura. In the future, we plan to offer API services only as a backup option. Additionally, for publishing new contents, we propose to implement a browser-based user interface that connects to Ethereum via Metamask and a local node. This will prevent (unintentional) corruption by updates or infrastructure failures in the future. We also suggest using IPFS or Swarm file storage [DT22] for hosting files and images. This allows files to be replicated across all stakeholders and other interested parties. Thus, everyone has access to all files and can mirror these files on their own infrastructure. So, unlike using a centralized storage provider, the files then become available

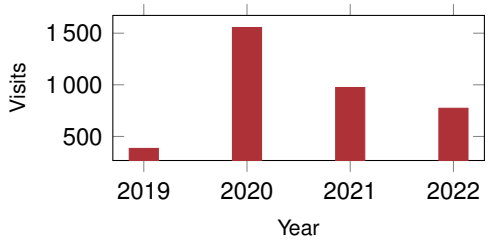


Figure 24. Visitors per year (between Nov 2019 and December 2022).

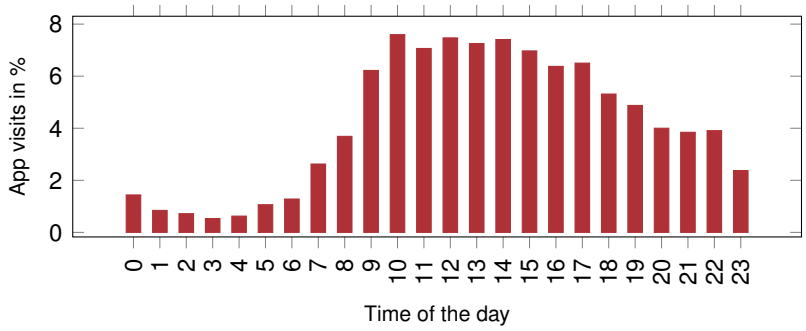


Figure 25. Visits per hour in percent (between Nov 2019 and December 2022).

from multiple servers; if one server becomes unavailable, the files are still available on other servers. As a result, network outages and maintenance issues would no longer affect the verification of BBBlockchain entries.

However, we are not considering to get rid of all centralized infrastructures in the near future. We still believe that BBBlockchain will benefit from native mobile applications and their typical system environment's features. For example, features like push notifications on mobile devices would allow us to provide instant updates, even if such an approach requires centralized infrastructures. Nevertheless, data management should be handled entirely by the smart contracts, without any centralized databases or server infrastructures.

Outreach

BBBlockchain users were asked for their consent for user statistics when they access the app for the first time. As shown in Figure 24, between 2019–2022 more than 3.7k visitors used BBBlockchain. This is particularly remarkable since BBBlockchain is not aimed for the general public, but focused on nearby living residents at the pilot projects' locations. Please note that the BBBlockchain pilots started at the end of 2019.

Figure 25 shows the traffic on the BBBlockchain app per year. The highest usage of the app was during average working hours. Interestingly, the majority of users accessed the app via smart devices (69 percent), rather than via desktop web browsers (cf. Table 4). We hence

User Device Type	Visits
Smartphone & Tablet	66.40 %
Desktop Browser	31.10 %
Others / Unknown	2.50 %

Table 4. Used devices (between November 2019 and December 2022).

conclude that introducing the BBBlockchain app offers the potential to reach a broader audience in participation processes than analog participation formats, which is a key ambition of all urban participation projects. However, we would like to stress that the BBBlockchain is not designed to replace analogue participation methods, rather to offer a hybrid option.

Key Findings

BBBlockchain provides a blockchain-based platform for participation during urban development processes. From an operational perspective, we showed how complex and long-term decision-making processes for urban participation can offer more transparency with blockchain technologies. Therefore, we have implemented real-word participation use cases in smart contracts. We do not claim that blockchains solve all trust issues for participation processes, but we consider the trade-offs between its risks and disadvantages on the one hand, and its potential and novel possibilities on the other hand.

TRANSPARENCY The BBBlockchain pilot projects have successfully confirmed that our platform can improve transparency and make information more accessible to citizens. From September 2019 to May 2023 a total of 48 entries were published on BBBlockchain by three main stakeholders: two housing associations (44 entries), the involved municipalities (2 entries) and the tenant council (2 entries).

However, the immutability of blockchain could increase accountability pressures on stakeholders. So, BBBlockchain has disrupted the communication procedures of institutional stakeholders, prompting them to rethink their approach. Additionally, using blockchain technologies had a big impact on stakeholders like housing associations and local authorities. They had to share more information than usual, which changed their analog communication practices. This was challenging because collaborative and frequent communication is not the natural approach of these stakeholder groups. Most importantly, blockchain's immutability raised concerns about accountability and exposure. Initial communication difficulties, however, were overcome by implementing new standards for transparent and inclusive sharing of information.

CONFLICT MANAGEMENT In order for conflicts to be properly managed, the involved parties must be aware of the existing accountability mechanisms and how to use the available information. Currently, it remains unclear to what extent citizens are legally empowered to monitor urban planning project integrity [EFS21]. Therefore, it is important for policymakers to clearly articulate how each party can be held accountable, by whom and how.

To this end, BBBlockchain is implemented to operate on public permissionless blockchains. Due to the permissionless design and an intuitive visual representation of hash values, the app is conceptually resistant to data manipulations and allows users, including non-expert users, to check the integrity of the data. This enables all involved stakeholders, citizens in particular, to openly monitor how urban planning projects unfold and to what extent they deviate to what was previously decided and agreed upon. Therefore, in case of conflicts among the interested parties, the immutable historical record stored on the blockchain, could represent a reliable and transparent basis to manage the conflict. Nonetheless, during the pilot phase of BBBlockchain, we did not observe any significant conflicts. Therefore, at this stage, we cannot report any experiences.

USER SURVEY We analyzed the impact of transparency with BBBlockchain through a user survey. Our user survey consists of $n = 25$ participants, conducted during the end of the first phase of the pilot project in September 2020. The scope of the survey specifically focused on how the platform might have improved citizens accessibility to information and if they truly felt that the platform could have their opinion taken into account in decision-making processes. In summary, our analysis suggests that the problem of improving citizen participation cannot only be solved through better technology. Technology is only one dimension to effectively design and implement a citizen participation platform. Effective participation needs to be conceptualized considering the broader socio-cultural context of applications.

Building on BBBlockchain, there are also other studies that go deeper into the results from an operational and participatory point of view [Rab+21; Iet+22; Iet+23].

Design Challenges

From a technical perspective, the deployment of BBBlockchain has highlighted several aspects that need improvement in terms of privacy and on-chain functionality. We identified specific areas for improvement, such as private communication, anonymous voting, and reliable voter authentication. For example, we have shown that there are significant transaction fees for publishing contents on the blockchain. If we were to apply the same magnitude of cost to a DApp-based open

communication platform with secure messaging on BBBlockchain, the costs would be high. Therefore, we need to find a solution to make communication cost-effective for participants. In addition, BBBlockchain revealed the need for reliable and anonymous blockchain voting. Since our pilots revealed serious flaws in the current implementation, we need to find better technical solutions. In the following, we use these insights and findings to propose privacy improvements and mechanisms for DApps in general.

Part III

PRIVACY MECHANISMS

6

SMARTDHX

This chapter is based on joint work published in [MT20] by Robert Muth and Florian Tschorsch. "SmartDHX: Diffie-Hellman Key Exchange with Smart Contracts." In: International Conference on Decentralized Applications and Infrastructures (DAPPS). *IEEE, 2020, pp. 164–168*

In this part, we present three privacy mechanisms for decision-making DApps with a focus on on-chain verifiability. Through our experience with developing and operating BBBlockchain, we discovered several technical limitations regarding privacy and scalability for reliable decision-making. One problem we have found is that DApp participants, who need to communicate with each other for decision-making, cannot handle all communication on the blockchain due to privacy issues and transaction costs. However, it would be critical if communication was hindered simply because it was too expensive, especially for our BBBlockchain use case. To address these issues, we developed an on-chain, multi-party Diffie–Hellman key exchange protocol that allows DApps to establish a secure off-chain communication channel between participants and can help to keep transaction costs low.

6.1 MOTIVATION

The core concept of decision-making DApps is that all actions take place on the blockchain, rather than on centralized off-chain servers. Therefore, we are looking for a solution to bring all communication on-chain while preserving privacy and blockchain properties. We propose to utilize Diffie–Hellman key exchange (DHKE), which is a building block for many cryptographic algorithms, to establish a shared secret over an open channel [DH76]. For example, to facilitate secure web browsing, the client first negotiates a secret key with a web server to encrypt the subsequent communication. Even a (passive) eavesdropper cannot reconstruct the secret key when DHKE is employed. To this end, we present *SmartDHX*, a blockchain-based DHKE scheme with multi-party capabilities. In SmartDHX, *all* cryptographic operations are implemented in a smart contract, without any client-side modifications or any additional libraries. This enables DApps to load the code directly from the blockchain and to execute a DHKE verifiably on-chain. With our scheme, DApp clients load runtime code from the SmartDHX smart contract, and execute it locally with a given random seed in a JavaScript environment. Therefore, clients do not have to know how the key exchange internally works, because the runtime code handles the interaction with the smart contract. For decision-making DApps, such an approach enables DApp participants to establish secure communication channels between their clients. For example, participants can start messaging with each other or exchange files.

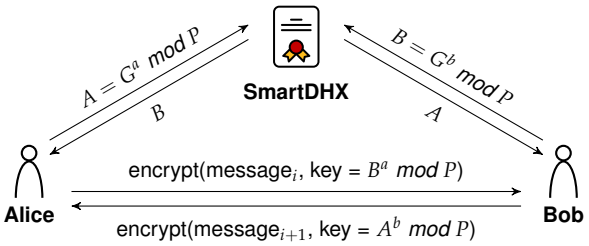


Figure 26. Overview of the clear-text communication between two participants and the SmartDHX smart contract for secure key exchange, and the resulting encrypted communication channel between the participants using the securely exchanged key.

Most DAOs still use the web for communication, e.g., participants use a bulletin board for discussion or emails. However, this has drawbacks for many decision-making processes, such as the lack of confidentiality or unreliable verifiability. To this end, we propose SmartDHX to establish a secure communication channel between DApp participants. As shown in Figure 26, participants use SmartDHX for secure key exchange (details on the exchanged values follow in Section 6.2). As a result, the securely exchanged key can be used to establish an encrypted connection for direct communication, e.g., via TCP or UDP sockets. For that, SmartDHX can run asynchronously because exchanged messages are stored on-chain, enabling DHKE for participants who cannot be online at the same time. Alternatively, for messaging, messages can be encrypted and securely exchanged via the permissionless blockchain using it as a mailbox. Since DHKE is designed to be secure in public networks, it will also remain secure in a blockchain setting. In addition, message integrity and authenticity are provided by the blockchain, therefore effectively mitigating Man-in-the-Middle (MitM) attacks. It is widely known, though, that "plain" DHKE is vulnerable to active MitM attacks. In order to protect DHKE from these attacks, clients cryptographically sign their messages, however, that requires a way to verify signatures. With SmartDHX the messages are signed and verified as blockchain transactions, and therefore do not need any extra public key infrastructure (PKI).

We show that all logic for the DHKE can be implemented in a smart contract without any modifications to the blockchain client, i.e., Ethereum or Geth, respectively. For this, we provide an implementation of SmartDHX as proof-of-concept. To this end, we implemented the cryptographic logic in Solidity with the Truffle framework and evaluate the approach using an Ethereum test network. With our proof-of-concept, we provide unit tests, which verify that multiple participants can exchange a secret key without storing it on-chain. The implementation can be tested locally and with a variable number of participants. As expected, the key exchange requires additional time due to the blockchain overhead. In order to show that our proposed scheme is

also capable of handling more complex cryptographic logic, we also implemented multi-party capabilities for SmartDHX.

Similar to our approach, McCorry et al. utilize Bitcoin for authenticated DHKE [McC+15]. To this end, the authors modified the Bitcoin Core client and implemented the DHKE logic as remote procedure commands, which are stored and executed off-chain. However, our approach contributes to a larger vision of truly DApps which store their logic on-chain without separating between a client and blockchain side. Right now, DApps are usually split into frontend client-side code, and the smart contract in the blockchain. Thinking one step further, our approach enables new cryptographical use cases, e.g., DApps which are completely stored on-chain, but can communicate with each other encrypted.

6.2 PROOF OF CONCEPT

In the following, we introduce the concepts of SmartDHX, its multi-party capabilities, and present our proof-of-concept implementation in a smart contract.

Two-Party SmartDHX

Two-party Diffie–Hellman key exchange allows to exchange secret keys between two participants for subsequent encryption without revealing it on the communication channel [DH76]. The security is not compromised when someone is passively listening. In order to exchange a secret key, the participants have to agree on a prime number P and a generator number G. Certain properties on the numbers have a direct effect on the security of the whole cryptographical computations [Boe88], which we do not discuss further for the sake of simplicity. There are also different cryptographic techniques for making DHKE more secure, e.g., by using elliptic curve cryptography [Mil85]. After agreeing on P and G, the participants choose private keys a, b randomly, and publicly exchange their results of $A = G^a \ mod \ P$ and $B = G^b \ mod \ P$, which we call public keys. Finally, the participants calculate the secret key $s = (G^B)^a \ mod \ P = (G^A)^b \ mod \ P$ independently, however retrieving the same result.

Listing 1 shows how to generate a and A for one party in Solidity, how to send the public key A to another smart contract as a transaction, and how to retrieve the final secret key s. For the sake of simplicity, we assume that the seed is given. Later, we will provide a solution to generate a random seed locally and securely without any additional requirements. Under this assumption, a passive adversary is unable to compute s, because she has neither learned a nor b from any of the transactions. Only active MitM adversaries or weak cryptographic parameters can weaken the security. Hence, as long as the discrete

```
// Globals
uint public G, P, B; // set during initialization

// Call for retrieving private a and public A based on a secret seed
function getA(uint seed) public view returns (uint a, uint A) {
  a = uint(keccak256(abi.encodePacked(seed, block.timestamp)));
  A = G.bigMod(a, P); // RPC modulo
}

// Send transaction for sending A to other SmartDHX contract (as B)
function sendA(DHX other, uint A) public { other.setB(A); }

// Call for retrieving the secret s
function getS(uint a) public view returns (uint s) {
  s = B.bigMod(a, P);
}
```

Listing 1. Two-party DHKE implementation in Solidity.

logarithm problem is considered difficult [Boe88], DHKE—and therefore also SmartDHX—can be used via untrusted channels such as blockchains.

For a deeper understanding of SmartDHX, it is important to notice the difference between calling a smart contract function locally and sending a transaction with a function call. All functions which use the private key a are executed *locally* (cf. `contract.method.call(...)`). By calling a function locally, no transaction will be broadcasted, and thus nobody can see that the function has been called, nor will the parameters be disclosed. Likewise, any changes on the blockchain's storage variables will be discarded without persistent change. The function can, however, return a value based on the blockchain's current storage. We use this feature to generate the public key A (without revealing the private key a) and to generate the secret key s. That way, we can store protocol logic in the blockchain without revealing any processed data. In contrast, "transmitting" the public A will be executed as a transaction (cf. `contract.method(...)`) and therefore permanently written into the blockchain. Please note, both parties involved in the DHKE have the exact same view on the deployed smart contract, using the exact same function for making their public key available to the other party's contract.

Multi-Party SmartDHX

DHKE can also be used for exchanging a single secret key between more than two parties. Of course, all parties could exchange keys bilaterally and then derive a common secret. Alternatively, all parties could use multi-party DHKE to agree on a shared secret key. In the following, we present multi-party SmartDHX, which generalizes the two-party approach.

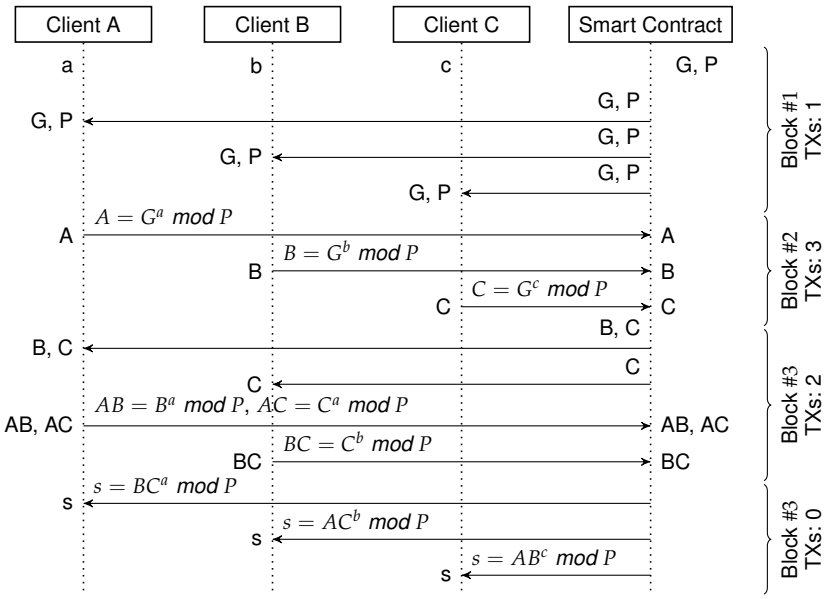


Figure 27. Multi-party SmartDHX for $n = 3$. Lower-case values remain secret, upper-case values are publicly stored in the blockchain.

For multi-party SmartDHX, we follow the same philosophy and implement the complete logic to perform the key exchange in a smart contract. Specifically, the logic to provide prime P and generator G, and the logic for calculating the random private keys $a, b, c, \ldots$ (with a random seed), the public keys $A, B, C, \ldots$, and the secret key $s = G^{(A \cdot B \cdot C \cdot \ldots)} \bmod P$ are implemented in a smart contract. Since we need to coordinate the message exchange between all parties, we implement an extra "control" smart contract.

Figure 27 shows an example for a three-party SmartDHX, including all calculations and smart contract interactions required to compute the secret key s. The first block contains the smart contract deployment, and thus the runtime code as well as the common parameters P and G. The arrows leading from the smart contract to the clients (right to left) indicate a local execution. Inversely, arrows pointing towards the smart contract (left to right) represent persisting transactions. The annotations on the right summarize the number of required transactions. Please note that a block can hold multiple transactions issued by different parties.

Even though the control smart contract could act as a MitM, it is unable to obtain (or derive) the secret key. While the smart contract could actually manipulate the exchanged messages between parties, all transactions are publicly available in the blockchain. Each party can therefore verify the correct execution of the smart contract. Thus, as long as all transactions are executed correctly and the program code of the smart contract is not malicious, a MitM attack is not possible.

```
// Generate local secret seed (32 byte == uint256)
let secretSeed = [...Array(32)].map(() => parseInt(Math.random() *
    256));

// Generate private a and public A (local call)
let dhxKeys = await dhx.generateA.call(secretSeed);

// Send public A to other participant via blockchain (transaction)
await dhx.sendA(dhxPartner.address, dhxKeys.A);

// Calculate secret key (local call)
return await dhx.calcS.call(dhxKeys.a);
```

Listing 2. Two-party key exchange implemented in JavaScript. The script is loaded from the SmartDHX smart contract, handles the blockchain communication, and returns the exchanged secret key.

Seeding SmartDHX

In order to generate a private key for DHKE, a client has to generate a secret random number, which is not trivial to achieve when accepting smart contract inherent code only. For one, the fundamental requirement of the Ethereum Virtual Machine (EVM) is a deterministic execution of all commands. Consequentially, Solidity does not offer a pseudo random number generator (PRNG). Smart contract developers instead retrieve a random number usually by using hash values of previous blocks or implement a commit-reveal scheme [Dam98]. Such a random number, however, would not be secret anymore and therefore is not usable for DHKE.

Our solution to the problem is to use JavaScript's PRNG. In particular, we deliver a JavaScript snippet with the smart contract that generates a 256-bit random number (as shown in Listing 2). Since this snippet is executed *locally*, it will not disclose the random number to the blockchain. For improved security, it would also be possible to import an NPM library with a cryptographically secure PRNG, because the JavaScript is executed in a Web3/Node.js environment.

Proof-of-Concept

We implemented SmartDHX in Solidity 5.8 with the Truffle framework 5.0.22 and make the code publicly available on GitHub.[16] The implementation's purpose is to showcase and analyze SmartDHX's feasibility only and does not implement any additional security measures against hijacking the smart contracts. In order to initiate the key exchange, each party deploys SmartDHX and executes JavaScript code locally, which is stored in and retrieved from the smart contract as shown in Listing 2. This script is executed in a Web3 environment

16 https://github.com/robmuth/smart-dhx

```
// SmartDHX deployment
let dhx = await deployer.deploy(SmartDiffieHellman);
let dhxPartner = await SmartDiffieHellman.at(0x...);

// Generate secret and exchange public A with dhxPartner
let jsGenerateExchangeKeys = await dhx.jsInitTransmit();
let dhxKeys = await eval("(async (dhx, dhxPartner) => {" +
    jsGenerateExchangeKeys + "})(dhx, dhxPartner)");

// Calculate exchanged key
let jsCalcSecret = await dhx.jsCalcSecret();
let secret = await eval("(async (dhx, dhxKeys) => {" + jsCalcSecret +
     "})(dhx, dhxKeys)");
```

Listing 3. Two-party SmartDHX Truffle migration script.

	Two-party SmartDHX	**Multi-party SmartDHX**
Secret keys	$\binom{n}{2}$	1
Transactions	$n(n-1)$	$\sum_{k=1}^{n} k$
Blocks	3	$1+n$
PoC runtime, $n=2$	75 s	165 s
PoC runtime, $n=9$	1 275 s	375 s
PoC fees, $n=2$	2 813 350 Gas	7 184 970 Gas
PoC fees, $n=9$	11 443 649 Gas	125 366 493 Gas

Table 5. Minimum number of secret keys, transactions, and blocks for two-party SmartDHX compared to multi-party SmartDHX, and the key-exchange time and Gas costs of our proof-of-concept (PoC) implementation in the Ethereum Rinkeby Testnet with n participants.

during a Truffle migration and invoked by JavaScript's runtime evaluation command `eval(...)` as shown in Listing 3. After all parties executed the script, it returns the exchanged secret key. The main advantage of storing the key exchange script in a smart contract is that no third-party is needed, e.g., a web server, that provides the script. This makes the program code verifiable and has the potential to improve user experience, because no additional client software is required.

Performance

In the following, we analyze blockchain specific metrics instead of network metrics like latencies or bandwidths. Accordingly, we do not compare the execution time of off-chain DHKE with SmartDHX, as mining can be expected to induce significant delays.

In Table 5, the number of exchanged secret keys, issed transactions, and blocks are compared between two-party and multi-party SmartDHX. In order to exchange bilateral keys between n clients, two-party SmartDHX needs at least $\binom{n}{2}$ secret keys and twice as many transactions, i.e., $n\,(n-1)$. Since all key exchanges can run independently from each other, two-party SmartDHX can be completed in a minimum

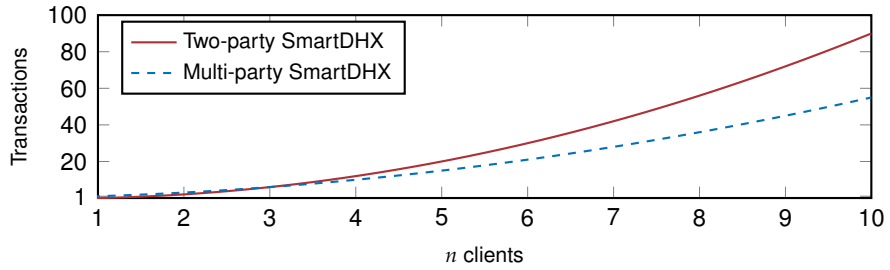


Figure 28. Number of transactions for two-party and multi-party SmartDHX.

of three blocks (one block for the deployment and two blocks for the two-way handshake between clients).

For multi-party SmartDHX, let us revisit Figure 27. The deployment of the smart contract and providing P and G requires one block. In the following blocks, the key exchange will be step-by-step completed by each client, always adding another public key. For example as shown in the figure, the public keys A, B, C are exchanged in Block #2. Next, clients can calculate the public keys AB, AC, BC, and publish them in Block #3. In each round, another client terminates, because of the redundancy in the public keys, which eventually leads to a decreasing number of transactions every block. For example in Block #3, Client C could calculate AC and BC, but they are also calculated by Client A and B. As a result, multi-party SmartDHX needs a minimum of $n + 1$ blocks and $\sum_{k=1}^{n} k$ transactions.

In a best-case scenario, two-party SmartDHX can exchange secret keys faster than multi-party SmartDHX for more than two clients, if all transactions are mined in the minimum number of blocks. Multi-party SmartDHX, however, needs less transactions, but the number of blocks increases by 1 per participant.

The overall time for a key exchange with SmartDHX depends on the average block generation rate. That is, for a number of blocks β and the average block rate λ_β, the execution time t is given by $t = \beta \cdot \lambda_\beta$. In terms of economic performance, however, the number of transactions might be more interesting than blocks: With an increasing number of clients, two-party SmartDHX requires more transactions than multi-party SmartDHX, which can be seen in Figure 28. In case of Ethereum, more transactions do not necessarily lead to higher total costs, because transaction fees depend on the computational complexity. Even though the number of transactions for two-party SmartDHX surpasses multi-party SmartDHX, the overall gas price for our PoC multi-party SmartDHX is higher (cf. Table 5). The reason are the many on-chain key distributions (i.e., write operations) in the smart contract. In the end, there are two axis that influence the decision: First, one has to decide whether parties should exchange separate keys or a single shared key. Second, we need to tradeoff speed (i.e., number of required blocks) and costs (i.e., number of required transactions).

Security

Besides the security of "plain" DHKE, as described in [DH76; Boe88; Mil85], the blockchain's security also directly influences the secrecy of the final exchanged key. We already pointed out that DHKE is safe as long as no MitM can actively and secretly manipulate the communication between the participants. But permissionless blockchains with longest-chain consensus rules can be attacked with the so-called 51%-attack [KDF13], which allows changes in the blockchain retrospectively. That way, an attacker could actively change blockchain transactions, which is the worst-case scenario for DHKE. Fortunately, an attacker cannot manipulate transactions without also tampering the sender's identity or transactions signature. With the identity management of blockchains, transactions can be authenticated as shown by McCorry et al. [McC+15]. They analyzed the security of DHKE via Bitcoin, that can also be applied to Ethereum's transaction authentication. For that, they sketched proofs in their security analysis for the private key security and session key security, which also applies to our approach. So, as long as all participants can trust and verify each other's transaction signatures a 51%-attack does not threaten the SmartDHX security.

Discussion and Application Areas

The costs to perform DHKE on-chain become an issue. In particular the costs for multi-party SmartDHX seem high. However, depending on the use case, the additional costs can be negligible, e.g., exchanging a shared key to enable an encrypted broadcasting. For instance, in situations where many users receive encrypted data as broadcast messages, it could be beneficial to have a single shared key. With a shared key and using the blockchain as a broadcast medium, only a single transaction is needed for broadcasting an encrypted message to many recipients. Therefore, transactions fees can be reduced and might compensate (break even) for the expensive key exchange.

However, we generally observe that SmartDHX is executable in a reasonable amount of time and offers some very interesting properties, including asynchronicity as well as message integrity and authenticity. As a result, SmartDHX can be used to enhance the privacy of DApps users with communication channels between the DApp users, e.g., for exchanging private messages with encryption instead of using a centralized web forum. In addition, off-chain communication channels can be used to reduce transaction costs and avoid scalability issues.

Another potential application, which emphasizes a feature of our approach, might be plausible deniability as in Off-the-Record Messaging [BGB04]. In addition to encrypted on chain communication, the blockchain can be used to disclose a user's MAC keys to a wider audience.

Looking outside the box, our protocol is also suitable for other use cases not related to decision-making. As described in [McC+15], this can be used to provide end-to-end encrypted communication for post-payment scenarios.

6.3 CONCLUSION

We have shown that it is possible to fully implement DHKE in an Ethereum smart contract, allowing participants to establish a secure communication channel over blockchains. To this end, we implemented not only the cryptographic logic in Solidity, but also the client-side logic for interacting with the smart contracts. In our proposed scheme, clients retrieve their program logic directly from the smart contract and execute it locally in a Javascript environment. Thus, SmartDHX can be implemented in decision-making DApps to enhance privacy and avoid scalability issues. In addition, we provide a building block that contributes to the vision of storing DApps entirely in smart contracts without splitting them into blockchain-side and client-side code.

7 ANONYMOUS CREDENTIALS VERIFIER

This chapter is based on joint work published in [Mut+22a] by Robert Muth, Tarek Galal, Jonathan Heiss, and Florian Tschorsch. "Towards Smart Contract-based Verification of Anonymous Credentials." In: Financial Cryptography Workshop on Trusted Smart Contracts. Vol. 13412. Lecture Notes in Computer Science. Springer, 2022, pp. 481–498

During the development and operation of BBBlockchain, we identified the problem of *reliable* privacy-preserving voting. We can either leave votings completely open and risk the reliability of the results, or we can restrict them so that users cannot vote arbitrarily. To ensure reliable voting and decision-making, DApps typically need to verify identity-related information of their users. For example, identifying all users before they can vote to prevent double-voting, or limiting voting to a pre-defined group of users. However, personal identity-related information should be kept private and should not be processed on a public blockchain. It is therefore common practice to use centralized infrastructures to verify personal identity-related information off-chain instead of using a smart contract-based DApp. To this end, we propose using *anonymous credentials* for DApps [CL01] to realize privacy-preserving, reliable user verification. Unfortunately, anonymous credentials cannot yet be verified with smart contracts in the EVM. Currently, they are implemented for Hyperledger Indy [@Hyp18] blockchains and can be managed with the Hyperledger Ursa cryptography library. In this chapter, we therefore examine how anonymous credentials work and how they can be verified with EVM-based smart contracts.

7.1 MOTIVATION

Decision-making DApps benefit from the inherent properties of blockchains, namely transparency and censorship resistance; In particular, autonomous organizations [@Jen16], decentralized finance [Qin+21] DApps, and participation apps (as presented in Chapter 5). However, interacting with many DApps often requires their participants to demonstrate personal identity-related information. For example, in decision-making DApps, participants must first prove that they are eligible to vote by providing personal information confirming that they meet certain criteria, such as being a certain age or living within a certain address range.

Unfortunately, in current practices, verification is often realized off-chain using centralized services and infrastructures. For example, in the Open Voting Network implementation for Ethereum [MSH17], the voting initiator approves voters' eligibility off-chain and publishes an acceptance list of eligible voters to the smart contract.

This is also a major issue for other DApp use cases, such as Defi related DApps. For example, DApps for crypto token offerings, where an initial amount of free tokens is released to attract new community

members, identity-related information is used to prevent malicious users from exploiting these *airdrops* through Sybil attacks [Dou02], i.e., repeated requests for free tokens using different blockchain accounts. During the Stellar airdrop [@Dal19], for example, users were required to prove that they have a valid GitHub account with a past registration date to prevent attackers from creating multiple GitHub accounts as their Sybils. The account validity has not been verified on-chain but through an allegedly trusted off-chain verifier. In both examples, a dishonest verifier can cheat without being noticed, e.g., by unjustifiably denying access to the voting or tokens.

To maintain the decentralization and censorship resistance of decision-making DApps, verification must be performed on-chain. However, this introduces two new challenges: runtime isolation of DApps and privacy. Verifying identity-related information typically requires a trusted third party to vouch for the accuracy of the information. In the airdrop example, it is not enough for the user to claim ownership of a GitHub account; instead, GitHub itself must certify it. Similarly, in the voting example, the required identity information could be attested to by a public institution. However, smart contracts run in an isolated execution environment, so they can only access information that exists in the same runtime. To access off-chain information, they require an oracle, which implies trust in trusted third parties [HET19]. In addition, verifying identity-related information typically reveals personal information to the verifier. This is already a problem for off-chain use. However, when verification happens on-chain, sensitive information becomes accessible to unauthorized blockchain nodes and immutably anchored on-chain for an unknown period of time. As a result, identity information cannot be naively verified on-chain to protect users' privacy rights and comply with applicable privacy regulations.

One approach towards privacy-preserving verification of identity information are *anonymous credentials*. They can be implemented by using zero-knowledge proofs to enable credential verification without revealing sensitive identity attributes to the verifier. While anonymous credentials have been around for a long time [CL01], they have increasingly gained attention. For example, they are part of Hyperledger Indy [@Hyp18] a decentralized credential management system where they are realized based on Camenisch-Lysyanskaya (CL) signatures [CL01].

Anonymous credentials are a promising solution to overcome the privacy limitations of the blockchain and enable non-revealing verification of identity-related information. However, the verification of anonymous credentials takes place as part of an interactive process between the identity issuer, the identity holder, and the verifier. This conflicts with the limited ability of a smart contract to communicate with off-chain infrastructures. Furthermore, smart contracts are constrained by technical properties of the underlying execution environ-

ment, e.g., the Ethereum Virtual Machine [Woo22], which must be taken into account when implementing anonymous credentials verification. To date, there is no smart contract-based implementation for verifying CL signature-based anonymous credentials [CL01].

Therefore, we design and implement a mechanism for smart contract-based verification of anonymous credentials issued as part of Hyperledger Indy's credential management routine. First, we propose a procedure for integrating smart contract-based credential verification into Hyperledger Indy. The procedure connects two previously disconnected worlds, allowing blockchain-enabled DApps to verify anonymous credentials issued by Hyperledger Indy-based systems. Second, we present a technical specification that explains the verification of CL signature-based anonymous credentials [CL01] in a developer-friendly way. The specification is based on both formal descriptions [CG12] and insights gained from an analysis of the Hyperledger Indy SDK code repository [@Fou22b]. Finally, we provide a proof-of-concept implementation for verifying CL signature-based anonymous credentials on Ethereum in Solidity, and we document the technical challenges we encountered during implementation.

7.2 ANONYMOUS CREDENTIALS

In the followoing, we first introduce concepts and roles related to anonymous credentials in Hyperledger Indy. Based on this overview, we propose a procedure for integrating smart contract-based verification of anonymous credentials into Hyperledger Indy's standard credential management routine [@Sov21].

Anonymous Credentials in Hyperledger Indy

Credentials can be understood as a set of claims about the holder's identity [Müh+18], i.e., statements about specific identity attributes such as the holder's name, address, or date of birth. Hyperledger Indy [@Hyp18] provides a decentralized system for managing such credentials, based on the principles of the self-sovereign identity (SSI) paradigm [@Chr16], in which the holder is in control of its identity claims, rather than allowing third parties to control them. Additional evidence is required to independently verify a credential. Evidence is typically attached to the credential by a trusted third-party that can attest to the credential holder's attributes, e.g., with a cryptographic signature. This will make them *verifiable credentials* [@Wor22].

However, naive verification of signatures constructed on identity attributes could reveal potentially sensitive identity attributes to the verifier, potentially violating the holder's privacy rights. Furthermore, revealing credentials to a decision-making DApp may violate our requirements for independent processes, e.g., for anonymous voting. Anonymous credentials [CG12], to that end, address these problems

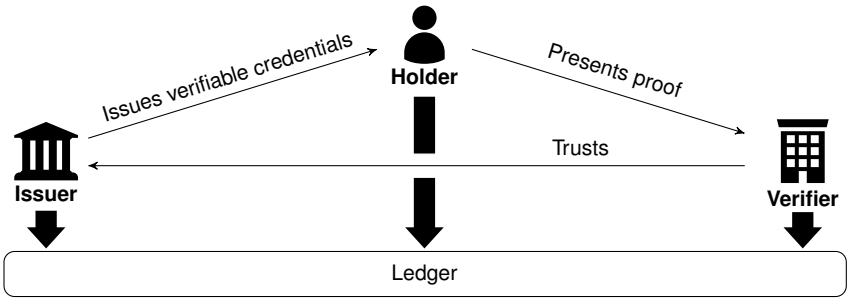


Figure 29. SSI Model adopted from [Müh+18].

by enabling selective disclosure for credential verification, i.e., verifying predicates on sensitive identity attributes without revealing them to the verifier. Therefore, Hyperledger Indy implements Camenisch-Lysyanskaya signatures (CL) [CL02], a signature scheme with zero-knowledge properties.

As shown in Figure 29, credential verification requires three roles: The identity *holder* who owns an identity, the *issuer*, who issues and attests to identity attributes through verifiable identity claims, and the *verifier* who verifies the identity claims. Hyperledger Indy also uses a public, permissioned blockchain, the *ledger*, as a public and decentralized storage system to make public artifacts accessible to all stakeholders. Private credentials of holders are stored in personal wallets, protecting them from unauthorized access.

Smart Contract-Based Integration of Anonymous Credentials Verification

We now show how anonymous credentials of Hyperledger Indy can be used for smart contract-based verification, i.e., Ethereum smart contracts. In order to establish compatibility between Hyperledger Indy and the EVM we define the following requirements:

- No infrastructure modifications: Smart contract developers should be able to build on existing Hyperledger Indy systems for credential verification. Therefore, on-chain verification should be integrated without any changes to the native Hyperledger Indy system.
- No further trust assumptions: In Hyperledger Indy, issuers are trusted by the verifier to truthfully attest to a holder's credentials, and no further trust assumptions should be introduced. More specifically, integration should work without trusted oracles [HET19].

The integration procedure builds on the native Hyperledger Indy credential verification flow, as described in [@Sov21]. As an extension, we propose a new verifier role, i.e., the *verifier contract*, and introduce the

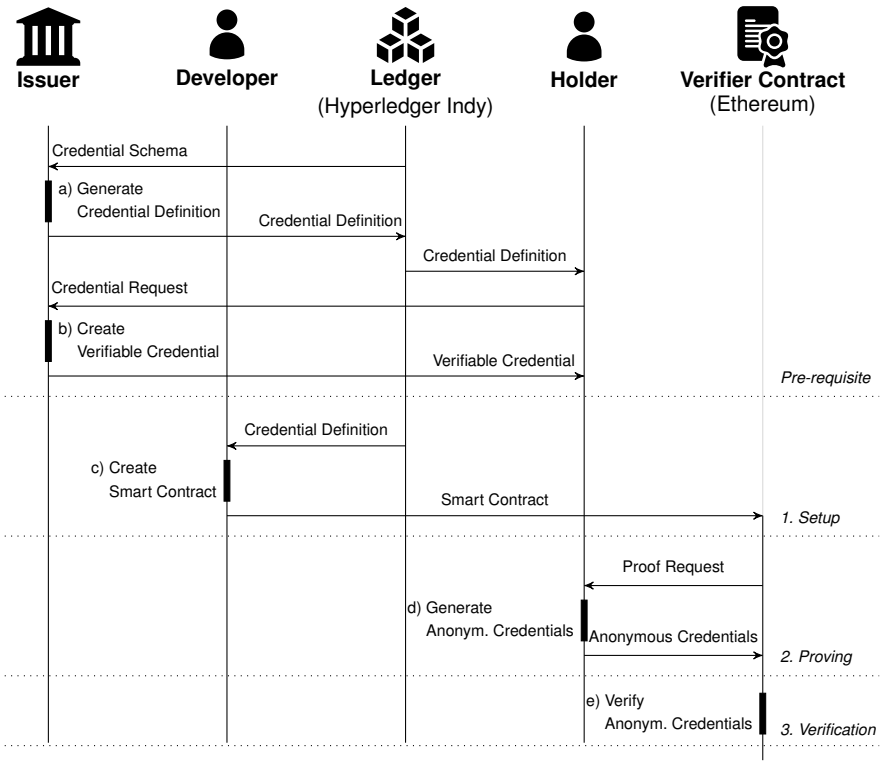


Figure 30. Adopted proof verification flow of the Hyperledger Indy proof verification procedures with a smart contract verifier.

developer, who is responsible for implementing and deploying the smart contract. As shown in Figure 30, the entire proof verification process can be divided into three phases: setup, proving, and verification.

Pre-requisite: As a pre-requisite, we assume that a *credential schema* exists that defines a set of identity attributes. From that, the issuer has generated a *credential definition* and registered it on the Hyperledger Indy ledger (Step a). The credential definition is a public artifact that is typically specified by the issuer and accessible on the ledger. Among others, it contains the set of attributes to be verified, the issuer's public key, and a reference to the signature algorithm which, in our case, is the CL signature scheme.

We also assume that the holder is already in possession of the verifiable credential that is required by the smart contract. In Hyperledger Indy, this is done in the form of a *credential request* [@Sov21] submitted from the holder to the issuer. On receiving the request, the issuer creates the evidence, here the CL signature, and returns the verifiable credential to the holder where it is stored in her wallet (Step b).

1. **Setup**: During the initial one-time setup, the developer determines the set of attributes and predicates to be verified on-chain and the attesting issuer in the form of a *proof request*, which in Hyperledger Indy is typically created by the verifier. Accordingly, the developer selects a credential definition from the ledger. For

simplicity, we assume that the determined proof request matches a single credential definition. Based on the issuer's public key, attributes, predicates, and a reference to the credential definition, the developer creates the smart contract (Step c). In Hyperledger Indy, this information is contained in the credential definition and the proof request. The developer then deploys the smart contract to the blockchain.

2. **Proving**: The holder obtains the proof request from the smart contract which also includes a reference to the credential definition on the ledger. Based on the credential definition and schema obtained from the ledger as public information, and the verifiable credential taken from the wallet as private information, the holder constructs a zero-knowledge proof based on her CL signature (Step d) to obtain anonymized credentials. Finally, the holder submits the resulting anonymous credentials to the smart contract.

3. **Verification**: On reception, the smart contract (i.e., DApp) verifies the proof using the on-chain credential definition (Step e).

Successfully verified credentials can then be used as part of the DApp's logic. As shown by the motivating examples, the credential verification can represent a pre-condition for using specific functionality provided by the DApp, e.g., a voting.

7.3 PROOF VERIFICATION

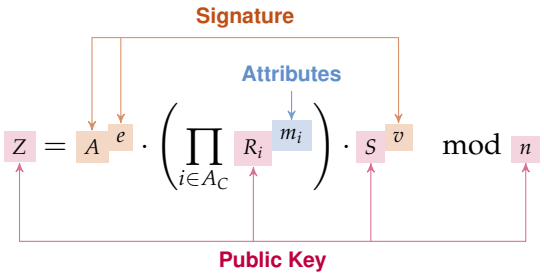
Given the high level description of integrating smart contract-based anonymous credential verification into the native Hyperledger Indy credential management routine, we now focus on the details of proof verification. Therefore, we first introduce the basics of CL signature verification, which is used in Hyperledger Indy to construct zero-knowledge proofs. Based on this, we describe the actual anonymous credentials proofs, i.e., a primary proof that builds on equality proofs and inequality predicate proofs.

Signature Proof of Knowledge

Conceptually, ZKPs in anonymous credentials prove knowledge of some discrete logarithms modulo a composite [CG12], and are referred to as *signature proofs of knowledge*. They allow a credential holder to prove possession of a CL signature over certain attribute values without revealing other attributes, as well as to prove that an attribute value lies within a certain range without revealing it. To understand how this is accomplished in anonymous credentials, we give an overview of what a CL signature looks like and show how a signature proof of

knowledge is generated for it. We do this by referring to corresponding steps in the procedure presented in Section 7.2.

As part of the pre-requisites, the credential issuer creates the credential definition (Step a). Therefore, she generates a CL signature key pair based on a credential schema which contains a set of predefined attributes (e.g., attributes of a driver's license). The CL signature scheme [CL02] defines the public key in this key pair as the quadruple $(Z, S, \{R_i\}_{i \in A_C}, n)$ where A_C is the set of indices of attributes in the credential schema, n is a special RSA modulus [CL02], and $Z, S, \{R_i\}$ are random quadratic residues modulo n. Then, on receiving a credential request from the holder, the issuer attests to the attribute values $\{m_i\}$, and, for creating a verifiable credential (Step b), generates a CL signature as explained in [CL02], such that the following holds:



$$Z = A^{e} \cdot \left(\prod_{i \in A_C} R_i^{m_i} \right) \cdot S^{v} \mod n \tag{1}$$

While the public key information $Z, \{R_i\}, S$ and n are publicly available on the ledger, the signature (A, e, v) is proven under the strong RSA assumption to be computable only by the issuer [CL02], who owns the private key, in order to keep the whole equation true. Together with the values of the attributes $\{m_i\}_{i \in A_C}$, the holder is now able to prove possession of the credential. Therefore, she generates a ZKP (Step d) which proves the knowledge of the exponents $e, \{m_i\}_{i \in A_C}, v$ but keeps them secret when presenting the proof to a verifier. Thus, the verifier can still verify that the prover knows those exponents, and thereby, is in possession of a valid signature (Step e).

In addition, a holder can prove that a credential contains an attribute with a certain value that it reveals. For this we say $A_C = A_r \cup A_{\bar{r}}$ such that A_r contains revealed attributes and $A_{\bar{r}}$ contains unrevealed attributes that are kept secret. Since A_r and $A_{\bar{r}}$ are mutually exclusive, we can adjust the Equation 1 as follows:

$$Z = A^{e} \left(\prod_{i \in A_r} R_i^{m_i} \right) \left(\prod_{i \in A_{\bar{r}}} R_i^{m_i} \right) S^{v} \mod n \tag{2}$$

$$\frac{Z}{\left(\prod_{i \in A_r} R_i^{m_i} \right)} = A^{e} \left(\prod_{i \in A_{\bar{r}}} R_i^{m_i} \right) S^{v} \mod n \tag{3}$$

This allows us to prove that a credential contains a set of attributes with the revealed values $\{m_i\}_{i \in A_r}$, by proving knowledge of the exponents $(e, \{m_i\}_{i \in A_{\bar{r}}}, v)$ in Equation 3. We use these insights as a basis for the following proof descriptions.

Primary Proof Verification

Anonymous credentials, as explained in [@KL18], are based on the previously introduced CL signature proofs of knowledge. These proofs are similar to the Schnorr protocol [Sch91], and are made non-interactive (i.e., only one round instead of commitment, challenge, and response) by implementing the Fiat-Shamir heuristic [FS86]. Anonymous credentials structure a proof as a combination of different sub-proofs belonging to a *primary proof* that is used to verify them as a whole. For this purpose, a primary proof consists of a set of equality sub-proofs $\{\mathrm{Pr}_C\}$ and a set of inequality predicate sub-proofs $\{\mathrm{Pr}_P\}$:

- An *Equality proof* proves that a credential contains expected values
- An *Inequality predicate proof* proves that a credential contains a value that lies within a certain range (e.g., zip code between a given range)

Additionally, a primary proof also contains a set $\mathcal{C}$ with necessary information for the non-interactive ZKP execution, and a ZKP challenge c to verify the correct execution. Along that, the verifier requires the proof generator (i.e., the holder) to include a given nonce η, so that the verifier can make sure that the proof corresponds to a particular proof request. During verification, sub-proofs are processed one after the other, and their results are appended to a set $\widehat{\mathcal{T}}$ which is shared across all the sub-proofs. At the end, the ZKP responses in each sub-proof are individually processed and the results are aggregated into $\widehat{\mathcal{T}}$. Once $\widehat{\mathcal{T}}$ is complete, the verifier hashes the final result of $\widehat{\mathcal{T}}$, $\mathcal{C}$, and the nonce η. The proof verification succeeds if c equals $H(\widehat{\mathcal{T}} \mid\mid \mathcal{C} \mid\mid \eta)$.

Equality Proof Verification

In the following, we introduce equality proof verification. We therefore present the cryptographical procedure, as we did for the signature proof of knowledge introduction in Section 7.3. In fact, equality proof verification is based on the same technique, but is implemented over a randomized version of the CL signature [CG12]. However, we do not intend to recap all proof details, as they are explained in [CG12].

In a nutshell, an equality proof attests the possession of a CL signature (i.e., credential) over a set of expected attributes, but without revealing any other information. Still, A_r contains the indices of revealed attributes and $A_{\bar{r}}$ contains the unrevealed attributes. The difference is, a verifier receives $\mathrm{Pr}_C = (\widehat{e}, \widehat{v}, \{\widehat{m_j}\}_{j \in \bar{r}}, A')$ and the revealed attribute values $\{m_j\}_{j \in A_r}$ from a prover, but the original values A, v (cf. Equation 1) are not directly used. Instead, A' belongs to a randomized CL Signature (A', e, v') [CG12] where e, v' are exponents in that signature and $\widehat{e}, \widehat{v}$ belong to ZKPs of knowledge of e and v' respectively. The prover

generates this randomized signature based on the original CL signature she possesses by following the procedure described in [CG12] to guarantee unlinkability across different proofs.

For completeness, we present the equality proof equation which calculates the sub-proof results $\widehat{T}$. For the sake of simplicity and easy recognition of relevant components, we highlighted the parts in the equality proof equation that belong together:

Attributes **Signature** **Public Key** **Zero-Knowledge Proof**

$$\widehat{T} \leftarrow \left(\frac{Z}{\left(\prod_{j \in A_r} R_j^{m_j} \right) (A')^{2^{596}}} \right)^{-c} \cdot (A')^{\widehat{e}} \cdot \left(\prod_{j \in A_{\bar{r}}} R_j^{\widehat{m_j}} \right) \cdot (S^{\widehat{v}}) \pmod{n} \tag{4}$$

In the end, computing $\widehat{T}$ is a pre-verification step for the given ZKP parameters, and is completed as part of the primary proof verification. At this point, we want to underline that the same c is used for exponentiation as for the final hash comparison in the primary proof.

Inequality Predicate Proof Verification

An inequality predicate consists of an attribute, one of $>, \geq, <, \leq$ as a comparison operator, and a constant value to compare to. Again, the credential holder proves that a specified inequality is satisfied without revealing the actual value of the attribute. In order to do that, the prover constructs a zero knowledge proof that the inequality predicate is satisfied. Additionally, the prover attests that the attribute indeed belongs to her, by constructing another zero knowledge proof. This zero knowledge proof is in fact just an equality proof that does not reveal the attribute's value.

Verifying an inequality proof requires processing the associated equality proof first where the index of the attribute in the predicate belongs to $A_{\bar{r}}$ (unrevealed attributes). Afterwards, predicate-specific zero knowledge proofs are processed, each extending $\widehat{T}$ in the primary proof verification. Since the computations involved in processing an inequality predicate proof are very similar to those involved in processing an equality proof, we omit them from this section and refer to the anonymous credentials specification [@KL18].

7.4 IMPLEMENTATION

In this section, we present our proof-of-concept implementation for on-chain anonymous credential verification. We therefore introduce our smart contract implementation and provide an overview of its technical design. With reference to Section 7.2, we show how the complex proof verification can be implemented and executed with limited and isolated EVM resources. Finally, we evaluate the transaction cost of proof verification using our implementation.

Proof-of-Concept

As part of our implementation for anonymous credentials verification, we provide a Truffle project with a single smart contract implementation for the proof verification. In addition to that, we include Mocha (Node.js) test cases for single proof verifications and full proofs with combined equality and predicate proofs. We consider our implementation a proof-of-concept, since we limit it to verification only and focus on the technical feasibility. We also stress that our implementation should not be considered production-ready yet. The source codes is available on GitHub.[17]

We use the Hyperledger Ursa cryptography library[18] as a reference implementation and replicate relevant parts of its test cases to ensure that our implementation produces the same results. We also provide additional test cases which are used for our evaluation.

For the smart contract development, we face several challenges regarding the CL signatures' key size. Firstly, proof verification requires arithmetic operations $(+, -, \cdot)$, exponentiation, and multiplicative inversion modulo a large number. As explained in Section 7.3, n is defined by the public key as the modulus. In our case, anonymous credentials and the implemented test cases use n of size 2 050 bits[19], which exceeds the size of the EVM's largest data type for numbers (max. 256 bits for unsigned integer). To this end, we integrate a big number library[20] for on-chain computations which ports parts of the OpenSSL big number implementation to Solidity and YAL (inline assembly). The library takes numeric input as byte arrays and performs computations in EVM memory. With EIP-198 [@But17a] the EVM offers a precompiled contract for computing $a^b \mod n$ where a, b and n can be larger than 256-bit unsigned integers. It is noteworthy that the big number library takes full advantage of this precompiled contract, as it also uses it for efficient multiplication.

17 https://github.com/robmuth/eth-ac-verifier
18 https://github.com/hyperledger/ursa/blob/34ef392/libursa/src/cl/prover.rs
19 n is 3 074 bits in anonymous credentials [@KL18], but HyperLedger Ursa sets n to 2 050 bits.
20 https://github.com/firoorg/solidity-BigNumber

Additionally, the big number library and the precompiled contract do not support computing modular multiplicative inverses. However, given a, b, n, it is possible to check if a is the inverse of b modulo n by utilizing the available exponentiation. Our implementation therefore computes the required modular multiplicative inverses off-chain and passes them together with the proof. For this reason, we use another big number library for off-chain computations in JavaScript[21] which pre-computes intermediary results (i.e., modular multiplicative inverses) for a proof verification. As explained above, our implementation verifies that the passed values are correct and aborts the execution otherwise. Thus the values are calculated off-chain, but all passed values are checked on-chain and rejected if necessary.

Finally, the number of variables required for proof calculations leads to capacity shortages for the Solidity compiler. The EVM is designed as a stack machine [Woo22] with a maximum size of 1 024 words (256 bits each), which is just enough to pass a complete proof in one transaction. Unfortunately, the limit of 16 parameters and local variables per function call makes all proof computations difficult, since our required parameters and computations (mainly big integer computations) cause stack overflow exceptions during compilation. Therefore, and for better code readability, we use predefined *struct* data structures to pack multiple parameters, and split computations into multiple functions that allocate and release local variables from the stack for intermediate computations.

The Truffle project provides basic migration scripts for the verification contract and the linked big number library. The test cases provide unit tests for the verification procedures, as well as, exemplary credentials. Once passed, they return the corresponding transaction costs for each verification in Gas, which we evaluate in the following.

Evaluation

We evaluate the costs for the deployment of the smart contracts and for transactions created by test cases with exemplary credentials. Therefore, we compile the smart contracts with Solidity 0.5.16 (enabled optimizer with 200 runs), and analyze the transactions with Ganache 2.5.4. Additionally, we implement the Hyperledger Ursa cryptography library unit tests into our proof-of-concept test cases and additionally generate our own exemplary anonymous credentials, issued with Hyperledger Indy.

Table 6 shows Gas costs for the deployment of the smart contracts and function calls of our test cases. While the deployment of a verification contract only puts the compiled byte code on the blockchain storage, executing a proof verification requires passing a full proof to the smart contract and executing the anonymous credentials verifica-

21 https://github.com/indutny/bn.js

Transaction	Gas
Verifier contract deployment	6 711k
Big number library deployment	77k
1. Test credential: Primary proof verification with equality sub-proof	32 001k
2. Test credential: Primary proof verification with inequality predicate sub-proof	84 826k
3. Test credential: Primary proof verification with combined sub-proofs	84 033k

Table 6. Transaction costs in Gas for smart contract deployments and different proof verification test cases in Ethereum.

tion process as explained in Section 7.3. In our first test, the credential contains a set of 14 identity-related attributes (e.g., name, date of birth, address, etc.) and reveals the first name with a primary proof and an equality sub-proof. Our second test contains the same attributes and an inequality predicate (i.e., date of birth has to be before a specified date), which is realized as a combination of an equality sub-proof and an inequality predicate sub-proof in the primary proof. By looking at the difference between the Gas cost for the first and second test credentials, one can see that the Gas costs increase with the number of sub-proofs, especially with inequality predicate proofs. This explains why the Gas costs of the third test, which reveals the first name and verifies the date of birth together, is not significantly different from the second test case.

Since Gas represents the resource consumption per EVM command [Woo22], proof verification becomes expensive for three reasons: First, the passed arguments (i.e., the proofs) contain large byte arrays, e.g., for each attribute and the corresponding zero-knowledge proof parameters. Second, the complex data structures for handling big numbers require allocation of EVM memory space, which consumes extra Gas [Woo22]. Third, calling the precompiled contract for exponentiation and modulo operations [@But17a] allows cheaper computations than an on-chain implementation; but due to the involvement of a large number of these operations in a proof verification, the Gas cost add up quickly. We also point out, that the number of attributes influences the proof size and therefore has an impact on the transaction costs, as well.

In the end, our evaluations show that the Gas costs are very high, so the resulting transaction fees (i.e., Gas multiplied by the demand-regulated Gas price) render it difficult for most DApp use cases. Considering a current exemplary Gas price of 100 Gwei, the full proof verification of our test case would cost approx. 8.4 Ether. However, we stress that our proof-of-concept implementation is not optimized for reduced Gas costs and there is certainly great potential for optimization (e.g., more efficient memory allocations). Furthermore, Gas costs cannot be translated directly to transaction fees in Ethereum, since a transaction sender specifies how much Ether she is willing to pay per Gas. This means, the actual transaction costs depend on the blockchain's current transaction load.

7.5 DISCUSSION

In this section, we discuss remaining open issues that should be considered for practical usages of our proposed solution. To this end, we put the evaluated transaction costs into context and present further options to implement our solution. For completeness, we also take a look at revocability of credentials and explain how it could be integrated into our smart contract implementation. At the end, we briefly discuss our considerations regarding Sybil resistance and unlinkability.

Transaction Costs

As shown in the evaluation in Section 7.4, our proof-of-concept demonstrates the general ability to verify anonymous credentials in a smart contract on Ethereum, and therefore enables compatibility with Hyperledger Indy-based identity platforms. However, the expected transaction costs are too high for current DApp implementations.

Nevertheless, blockchain technologies continue adopting new techniques to address rising transaction costs, scalability limits, and performance issues of execution engines, respectively [Pop19; Rou20; BET21]; Especially since hype-driven blockchain applications (e.g., Cryptokitties or ICOs) caused fees to skyrocket [SFG19] multiple times, in the past. However, since our implementation is EVM-based but not limited to the Ethereum Mainnet, we envisage other blockchains that are compatible with our Solidity implementation (e.g., Polygon/Matic[22] or Polkadot[23]).

In the meantime, we consider two possible ways to implement anonymous credentials verification on-chain: First, the computationally expensive big number operations could be implemented as precompiled contracts into the EVM, i.e., implementing big number data types. Precompiled contracts are natively implemented in the EVM client and can be addressed as external contracts by other smart contracts, but they define their Gas costs independently. Doing so, the overall transaction costs of our implementation could significantly decrease in the same way, as EIP-198 [@But17a] decreases costs for big integer modulo operations. Second, in the same manner, the whole verification procedure could be implemented as a precompiled contract. The latter, we at least consider reasonable for instantiating new EVM-based blockchains with anonymous credentials capabilities, since implementing precompiled contracts is common practice for new private networks. Hence, for example, anonymous credentials verification can be implemented as precompiled contract in the official Ethereum client *Geth* which also supports instantiating private and permissioned networks.

22 https://polygon.technology
23 https://polkadot.network

Non-Revocation Proof

Our solution works for credentials that are generally valid, that is, credentials over attributes that neither have an expiry date nor are revocable by their issuer (e.g., date of birth). In contrast, an issuer can create *revocable credentials* for attributes that are subject to change (e.g., address). In this case, it is possible that a revocable credential could have already been revoked by its issuer, by the time a verifier receives a proof based on it. Therefore, a verifier must be able determine a credential's revocation status in order to accordingly decide whether to accept or reject the given primary proof.

Anonymous credentials in [@KL18] allow a holder to prove that the credentials used during the primary proof construction have not been revoked. For this purpose, tracking the revocation status uses CKS accumulators [CKS09], where a so-called *accumulator value* is periodically published by the credential issuer, containing information about all non-revoked credentials. When a credential is issued, the issuer also provides the holder with a *witness*, which allows the holder to prove the validity of the credential with respect to the accumulator value. The process of checking the revocation status takes place as part of the credential verification process. In addition to the primary proof, a credential holder also sends a non-revocation proof, which is a zero-knowledge proof constructed over the accumulator parameters.

CKS accumulators make use of weak Boneh and Boyen signatures [BB08; BBS04], BN-254 curves [BN05] and type-3 pairing which are all different cryptographic primitives than the ones used by credential attestation proofs. Therefore, verifying those proofs is a subject for future work.

Sybil Resistance and Unlinkability

In our system, Sybils are resubmissions of already verified anonymous credentials by different holders. Since Sybil attacks can be effectively defeated by a trusted party that guarantees uniqueness [Dou02], we can instrument the only trusted party for this purpose: the issuer. If the issuer provides a unique identifier as an identity attribute, it can be verified on-chain as part of the proof. This establishes a verifiable one-to-one mapping between proof and holder. The unique identifier could be represented as a revealed attribute that is part of the proof construction and verifiable on-chain.

An alternative solution that does not require cooperation from issuers is described in [@KL18]. A verifier can require provers to turn off unlinkability in their proofs. We have shown in Section 7.3 that this is possible if a holder does not generate a randomized CL signature and instead, constructs proofs based on her original CL signature. Thus, the smart contract keeps track of all proof identifiers and, hence, can identify Sybils.

However, while storing unique identifiers of proof-holder mappings helps to defeat Sybil attacks, it also introduces linkability. Linkability is a privacy-related characteristic and applies in this context if the linkage of anonymous credentials allows unauthorized third parties to derive private information of the holder. In smart contract-based applications, linkability is critical since the history of blockchain transactions is available to all blockchain nodes.

While holders can protect against linkability by using different blockchain accounts each time they interact with the blockchain, this protection becomes ineffective if unique identifiers of holders are stored on-chain. They should, consequently, not naively be used for Sybil resistance but require further considerations to keep holders protected from linkability.

7.6 SELF-SOVEREIGN IDENTITIES

Hyperledger Indy has become one of the most popular SSI management technologies [SNA21] and is adopted in multiple SSI management projects. IDunion[24], for example, is a German consortium of public and private organizations that uses Hyperledger Indy to implement a SSI management system. Similarly, the Verifiable Organizations Network[25] is an initiative that leverages Hyperledger Indy to realize SSI and enable digitization of identities in a secure, user-centric manner. With our proposed solution, smart contracts can now verify credentials of any issuer that is part of these projects.

Beyond Hyperledger Indy, other blockchain-based credential management systems exist. In the context of SSI, uPort [NJ20] and Jolocom [@JOL19] are two approaches that, instead of building upon a public, *permissioned* blockchain, leverage Ethereum as a public, *permissionless* blockchain. These approaches, however, currently only store non-revealing identity information on-chain but do not provide anonymous credential verification. Given the privacy requirement, this makes them not applicable for smart contract-based credential verification. However, recent improvement proposals (e.g., EIP-725 [@VY20] and EIP-735 [@Vog19]) show that there is an interest in establishing Ethereum-based SSI-Systems, which has even yielded the formation of a self-proclaimed SSI alliance.[26]

7.7 CONCLUSION

Motivated by the privacy limitations of DApps, we have taken a first step towards verifying identity information in DApps in a privacy-preserving and trustless manner. Using the example of Hyperledger Indy as an established credential management system, we have shown

24 https://idunion.org
25 https://vonx.io
26 https://erc725alliance.org

how CL signature-based anonymous credentials can be verified by Ethereum-based smart contracts without introducing further trust assumptions.

With our approach, DApps can verify identity-related information completely on-chain, preserving transparency and censorship resistance as desirable properties of DApps. For decision-making DApps, a public authority could issue eligibility claims as anonymous credentials that are verifiable on-chain, helping decision-making DApps yield more reliable results. As our approach builds on concept of SSI, we envision an environment for decision-making DApps where sufficient trusted authorities are available.

However, we also revealed aspects that stand between our prototype and a production-ready system. As seen in our technical evaluation, Gas costs are currently impractically high due to the high verification costs of CL signature-based zero-knowledge proofs. However, the active research and ongoing developments in the area of verifiable credentials and smart contracts lead us to expect technological advances that can benefit smart contract-based credential verification. For example, a new anonymous credential design called Anonymous Credentials 2.0 has been proposed in [@Mic19], which may be adopted in Hyperledger Indy in the future.

In the end, we have shown that CL signature-based anonymous credentials can be verified in smart contracts, while paving the way for the integration of other identity infrastructures and technologies.

8

TORNADO VOTE

This chapter is based on joint work published in [MT23] by Robert Muth and Florian Tschorsch. "Tornado Vote: Anonymous Blockchain-Based Voting." In: International Conference on Blockchain and Cryptocurrency (ICBC). *IEEE, 2023, pp. 1–9*

The most important instrument for decision-making DApps is voting. To this end, in Chapter 4, we showed the relevance of blockchain-based voting for decision-making DApps, i.e., by evaluating funds and interactions. In Chapter 5, we showed the need for *anonymous* voting for reliable decision-making in our urban participation DApp. Following the participation levels of the IAP2 public participation model [@IAP14], BBBlockchain implements blockchain-based voting for the consultation use cases. For consultation, blockchain-based voting is implemented to ask participants for their opinions and to be consulted on decisions regarding the construction process. Similar to democratic elections, it is critical that all participants have confidence in the reliability, fairness, and security of the voting process and can submit their votes privately. While blockchain-based voting has been effective in establishing transparency and security, it is difficult to ensure anonymity and privacy, so that no one can see how others have voted. We therefore design and implement a novel privacy mechanism for anonymous voting focusing on decision-making DApps.

8.1 MOTIVATION

Many DApps hold considerable financial assets and are collectively managed by their stakeholders without delegating decision-making power to centralized bodies (cf. Chapter 4.1). Instead, a blockchain-based voting process is used to coordinate, where the smart contract logic collects votes and ensures that decisions are executed accordingly. That is, anyone can make a public proposal, which can then be accepted or rejected by others. Blockchain-based voting has therefore become an integral part for the governance of DApps.

The inherent transparency of blockchains, however, threatens voters' privacy as it can be considered pseudonymous at best [Bér+21]. Anonymous voting is often necessary for democratic decision-making processes, though. While the issue is known for some time and inherent to many blockchains, including Ethereum [@But21], existing solutions struggle with scalability [MSH17], require a central, trusted off-chain party [SGY20], or cannot directly be used for DApps as they require their own blockchain [Kil+22].

We therefore present *Tornado Vote*, a new blockchain-based voting protocol for Ethereum that yields anonymous, fair, and practical on-chain voting. To this end, we build upon the mixer protocol Tornado Cash [@Ale19], which enables an anonymous coin transfer service.

While Tornado Cash alone can be used for anonymous voting (as we will argue), we adapt the protocol to realize *fair* voting [DKR10] by keeping individual votes secret until all voters have submitted their votes. Additionally, Tornado Vote features (optional) properties such as delegation of voting rights and plural voting. In order to maintain security and anonymity despite our adaption, we use security analysis tools for the smart contract implementations [@Mue18; FGG19] and perform formal code verification, namely VeriSol [Wan+19]. In addition, we assess the feasibility of Tornado Vote. To this end, we develop different Gas cost models to quantify theoretical limits and real-world performance bottlenecks. Since the introduction of EIP-1559 [@But+19], block capacities are variable and transaction fees depend on past transaction loads. As we show, this mechanism can lead to unaffordable costs when populating blocks to the maximum. We therefore model the fee calculation of EIP-1559 and develop Gas cost models that optimize for a constant fee level. Our evaluation clearly reveals a feasibility trade-off between the number of blocks (or time) it takes to cast a number of votes and the required financial resources. The developed models can help to adjust this trade-off and find reasonable parameters.

To that end, in the following, we first introduce the basic concepts of the Tornado Cash protocol, which we later leverage to develop Tornado Vote. Additionally, we discuss our ethical considerations regarding the recent U.S. ban on Tornado Cash. Second, we develop Tornado Vote, which adapts the Tornado Cash protocol for fair and anonymous voting. Third, we evaluate our approach for scalability and cost. Finally, we briefly review related work and conclude this chapter with a summary of our findings on anonymous voting for decision-making DApps.

8.2 BACKGROUND

Tornado Cash

Tornado Cash [@Ale19] is a smart contract-based, non-custodial mixer for Ethereum's native cryptocurrency *Ether*, standardized tokens (e.g., ERC-20 compatible [@VB15]), and other blockchain assets. For the sake of simplicity, we will refer to all of these assets as *coins* in the following. In a nutshell, multiple accounts deposit coins of the same amount into a shared wallet and withdraw them with a new account in a way that cannot be linked. Anonymity is therefore achieved by hiding in a set of transactions that are indistinguishable, i.e., the so-called anonymity set. Tornado Cash provides such a wallet, i.e., a smart contract, and ensures by using a cryptographic proof that the connection between accounts is not disclosed. At the same time, the proof also ensures that users can only withdraw as many coins as deposited. In order to use Tornado Cash, users do not need to register beforehand and can immediately deposit coins with the first transaction. Users also determine the point in time when to withdraw the coins and therefore can wait for an

individually preferred anonymity set size by monitoring the number of deposits.

From a technical perspective, Tornado Cash provides a public smart contract instance, the so-called *vault* and external *relayers*. First, a user deposits a coin from their account to the Tornado Cash vault together with the hash of a personal secret r and a nullifier k. Additionally, the user deposits a fee coverage for future transactions to redeem a relayer in the following steps. Second, the user withdraws the deposited coin to a new account. However, since a completely new account for the withdrawal does not have any balance yet, it cannot cover the transaction fees to request the deposit. The user therefore contacts a Tornado Cash relayer via an anonymized communication channel (e.g., with Tor [DMS04]) and requests the deposit. For that, the user provides a zkSNARK zero-knowledge proof [Gro16] to prove the knowledge and the hash value of k, without revealing r or k in clear-text. The relayer then submits a new blockchain transaction with the ZKP and the hash to the vault to transfer the deposit to the new account. The vault remembers the nullifier and rejects any future request with the same. The vault also rewards the relayer with the pre-paid fee from the first step. Eventually, one cannot reliably trace the depositor back to the withdrawer.

Ethical Considerations

The U.S. Department of the Treasury's Office of Foreign Assets Control (OFAC) released a press statement on August 8, 2022, imposing sanctions on Tornado Cash. OFAC's motivation is to prevent money laundering and illegal financing. The sanctions had far-reaching implications, such as banning the use of Tornado Cash, banning trades with specific Ethereum wallets, and the temporary removal of all source code repositories from GitHub. Since Tornado Cash, however, is already deployed on the Ethereum Mainnet, it cannot easily be removed anymore. Operations will therefore continue, particularly since the deployed smart contracts have no assigned owner anymore and do not implement an emergency stop or similar. While our work is technically based on the same technology, the use is intended for anonymous voting—and not for any illegal purposes, which we distance ourselves from. In fact, we consider Tornado Cash to be a neutral technology on whose usage or exploitation we have neither influence nor take a position; rather, we consider its protocol only as building block for blockchain-based voting. However, another suitable mixer or blockchain can be used if necessary, e.g., CoinShuffle [RMK14] or MicroMix [@WG19].

8.3 TORNADO VOTE

In the following, we present Tornado Vote, a novel privacy mechanism for anonymous, fair, and feasible voting. It is specifically designed

for autonomous organizations, the governance of DApps, e.g., *The DAO* [@Jen16], and decision-making DApps in general. While we utilize blockchain transparency to ensure a secure voting process, we also require on-chain privacy to break the link between voters and their vote. We therefore use the Tornado Cash mixing protocol, which has proven effective in the past [@VK19; @KV19b; @KV19a].

In a naive approach, Tornado Cash (or another suitable mixer for that matter) can be used to realize a very simple voting. For instance, voters can use unowned accounts each representing a proposal's option and whose balances represent the votes. Accordingly, voters transfer coins anonymously to these addresses by using Tornado Cash (as explained in the previous section) and eventually compare the balances to come to a decision, e.g., the account/option with the higher balance wins. As illustrated in Figure 31, this approach has the great advantage that other (regular) Tornado Cash users enlarge the anonymity set. This naive approach, however, has some serious, inherent implications: it clearly allows plural voting, where one entity can vote multiple times. While this can be considered as weighted voting, there is no mechanism to easily limit the number of votes per entity. Moreover, the unowned accounts reveal the progress of the voting process and intermediate voting results, which violates the concept of fair voting [DKR10]. Lastly, the approach increases voting costs as it not only requires transaction fees but also transfers coins to a voting account.

In order to make these implications optional and not inherent to the voting process, we developed Tornado Vote. In particular, we extend the voting process with a commit-and-reveal mechanism, which we use to collect votes without revealing the result until the end of the voting, i.e., realizing fair voting. Since we additionally issue our own voting token without inherent monetary value, we can control the number of votes. Yet, plural or weighted voting can still be realized with a voting token by issuing the token according to the respective weights. In the following, we define design requirements, properties, and assumptions of our approach. Next, we present the protocol of Tornado Vote and its different voting phases. Finally, we present the technical infrastructure and security considerations.

Design Requirements and Properties

Tornado Vote inherits properties such as correct protocol execution and public verifiability from the underlying blockchain. Nevertheless, there are additional design requirements and properties of Tornado Vote, which we define and elaborate as follows.

ELIGIBILITY Tornado Vote requires a voting token to prove eligibility. More specifically, Tornado Vote requires its own custom ERC-20 token per voting and an initial voting administrator to create and transfer all tokens to eligible voters. Hence, the number of the number of

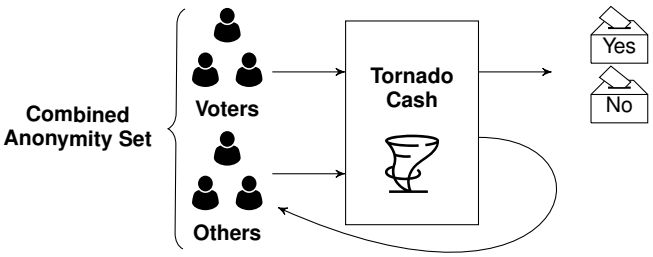


Figure 31. Utilizing Tornado Cash directly for anonymous voting yields an enlarged anonymity set comprising voters and other users: a voter deposits a coin and anonymously transfers it to one of multiple unowned accounts, whose balances eventually represent the voters' decision.

votes is limited by tokens and managed as standard blockchain tokens. The administrator is responsible for the correct handling at the beginning. After the distribution, the administrator should not own any tokens anymore and should be unable to mint new tokens, which both can be verified on-chain. A voter uses her token to initiate the voting process by depositing it to Tornado Vote. Please note that the voting token does not have any inherent monetary value and therefore does not unnecessarily drive the voting costs as in the naive approach.

TRANSFERABILITY In some cases, depending on the voting type, it makes sense to allow the delegation of voting rights, e.g., for representative or liquid democracy. Using an ERC-20 compatible token [@VB15] as voting token clearly allows and, to some extent, can even be used to encourage the transfer of voting rights. In fact, preventing transferability is a technical challenge because even if a token cannot be transferred to another account, the whole account could be transferred to someone else by sharing its private key. To this end, if a voting must reliably prevent transferability, the right to vote could be tied to the identity of a voter. For example, by using *anonymous credentials* to prove that a voter is personally eligible to vote regardless of the blockchain account, while keeping all personal information secret [Hei+22]. We therefore consider transferability an optional feature that Tornado Vote can enable or restrict.

PLURAL VOTING While democratic voting typically strives for equal voting weights, shareholders of capital stock, e.g., as in the DAO [@Jen16], gain voting weights according to their stake. We can realize weighted voting by issuing voting tokens to voters according to their stake, for example. This eventually leads to plural voting, an instance of weighted voting. As with transferability, we consider plural voting an optional feature for Tornado Vote.

FAIRNESS In the context of voting, fairness implies that preliminary counts do not influence voters while voting is in progress [DKR10]. Therefore, the voting system must be built in such a way that votes are not published before the final tallying. That obviously presents a technical challenge for public, permissionless blockchains. Tornado Vote therefore requires a commit-and-reveal mechanism. With this mechanism, voters first commit to their vote and only when all voters have cast their vote, the result will be disclosed as clear-text vote.

ANONYMITY Voter privacy is achieved by the unlinkability of voters' accounts and their votes. To this end, Tornado Vote is based on the Tornado Cash protocol. While the public token balances might reveal the eligible voter accounts, the mixing protocol ensures that voters can anonymize their voting choice. In order to achieve a high level of anonymity, a high number of other participating voters is necessary. As we will describe later in detail, a relayer service is also necessary to cast a vote. The interactions with relayers must be carefully timed to prevent de-anonymization attacks by time correlations, which we assume are the voters' responsibility. We also emphasize that, as with any other public voting process, voter privacy is only guaranteed if the final result is not unanimous. Additionally, network communications between the voters and the relayers must not reveal any metadata that could be correlated. Therefore, voters should use an anonymous channel to communicate with relayers, e.g., with Tor [DMS04].

TRUST Following the principles of a permissionless blockchain, trusted third parties and centralized components must be avoided. While Tornado Cash, and therefore Tornado Vote, require a separate relayer infrastructure, the relayers do not gain any further permissions or trust on the protocol side. Their only task is to forward transactions and cover fees, which will eventually unlink the sender and receiver accounts on chain. Additionally, a relayer cannot manipulate transactions due to their cryptographic signatures and ZKP verification, so that a violation would be noticed and rejected. We assume that sufficient relayers are available to choose from.

FEASIBILITY Blockchain-based voting systems are limited primarily by the blockchain transaction throughput and smart contract capacities. This limits the number of deposits and thus theoretically affects anonymity. According to our empirical analysis of blockchain-based voting in Chapter 4, we consider $n = 10\,\text{k}$ votes as a reasonable number for use cases such as a DAO. Therefore, Tornado Cash must be built to handle that many votes with respect to external blockchain constraints.

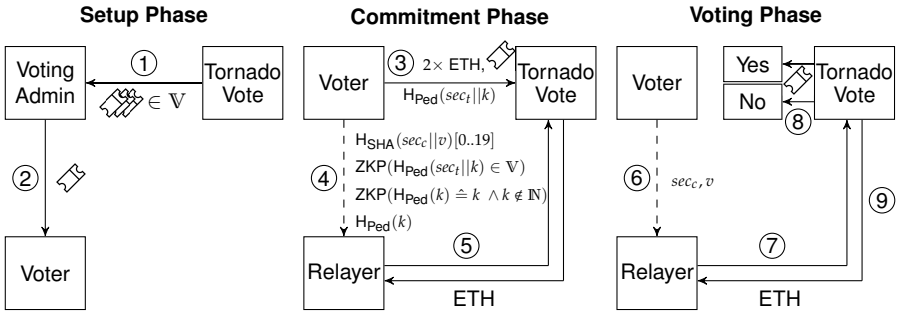


Figure 32. Setup: The administrator mints a limited amount of voting tokens $\mathbb{V}$ and transfers them to eligible voters. Depending on the type of voting, delegation of voting rights and/or plural voting are possible by transferring tokens accordingly. Commitment: Voters deposit their voting token, the fees in Ether (ETH) for the relayers, and a hash. Next, voters send ZKPs for proving ownership of the token and a commitment including their vote v to the relayer. The relayer forwards everything and receives a service fee. Voting: The voters reveal their commitment and vote to the relayer, who transfers the voting token to the corresponding vote address. The relayer receives a second service fee.

Protocol

Tornado Vote distinguishes between three roles: an administrator, voters, and relayers. Each voting is split into three phases: a setup, a commitment, and a voting phase. In the following, we will take a closer look at each of these phases. Therefore, Figure 32 shows the three phases and each interaction.

SETUP PHASE First, the smart contracts are deployed to the blockchain by the administrator, which comprises the Tornado Vote anonymity provider, ERC-20 compatible voting token, and the voting smart contract. Second, the administrator publishes the voting proposals and sets voting parameters, e.g., the voting period and decision quotas. Additionally, the administrator mints a limited amount of voting tokens exclusively for this voting. Each token allows the submission of exactly one vote and represents a single vote in the final phase. After this setup, the administrator owns all voting tokens $\mathbb{V}$ (Step 1). The administrator transfers these voting tokens irrevocably to the accounts of eligible voters (Step 2). Once the administrator has transferred all voting tokens, the voting process continues with the commitment phase.

COMMITMENT PHASE At the beginning of the commitment phase, voters generate their own individual random token secret sec_t and a random nullifier k locally. Each voter then transfers the voting token to the Tornado Vote vault (Step 3), similar to a deposit in Tornado Cash. This includes two times the transaction fee (required for the

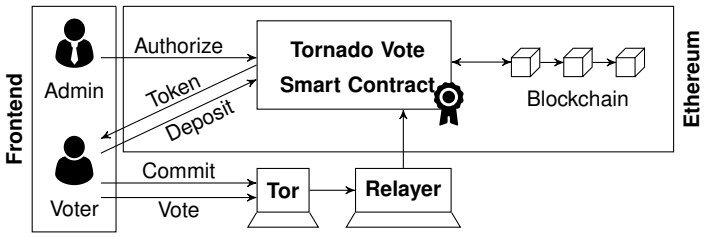


Figure 33. Tornado Vote's architecture and interactions between its components.

relayer) and the Pederson hash value $H_{\text{Ped}}(sec_t||k)$ [@Ide19]. We use the Pedersen hash, as in the Tornado Cash protocol, for efficient ZKP computations. Otherwise, we use SHA-3 where possible to save Gas. After an appropriate waiting time, the voters generate another commitment secret sec_c to commit to their vote v before revealing it. The voters therefore send the first 20 bytes of the hash value $H_{\text{SHA}}(sec_c||v)$, the hash of the nullifier $H_{\text{Ped}}(k)$, and a ZKP for sec_t to a relayer (Step 4). The relayer then forwards everything on-chain (Step 5) to Tornado Vote. Tornado Vote accepts the commitment if the ZKP contains a valid sec_t (without revealing it) and verifies if k was not used before. Therefore, Tornado Vote does not yet experience the voting choice but remembers the corresponding computed hash value and the hash of k in $\mathbb{N}$. Lastly, Tornado Vote rewards the relayer with Ether to compensate for the transaction costs. Once all voters have committed to their vote, we transition from the commitment to the voting phase.

VOTING PHASE In the last phase, all voters reveal their individual voting choices via a relayer (Step 6). To this end, the voters reveal their vote in clear-text and the commitment secret sec_c. This way, Tornado Vote can cross-check that the vote is eligible (Step 7). If so, Tornado Vote transfers one of the ERC-20-tokens to an unowned account, which represents the corresponding voting choice (Step 8). The final balances of these addresses eventually represent the final voting results. After each eligible vote is processed, Tornado Vote again rewards the relayer to compensate for the transaction costs (Step 9).

Implementation

Tornado Vote's smart contracts are implemented in Solidity for Ethereum and EVM-compatible blockchains. While Tornado Vote's source code is primarily based on Tornado Cash, it runs independently of it and is deployed entirely on its own. By default, Tornado Vote consists of four components: smart contracts, frontend, relayers, and anonymous communication channel (i.e., Tor [DMS04]). As shown in Figure 33, all interactions are managed by a central Tornado Vote instance. For the commitment and vote submission, voters use a Tornado

Vote relayer to prevent third parties from linking the two interactions. Furthermore, to protect connection metadata from the relayers, the voter communicates via Tor with the relayers. Internally, Tornado Vote is divided into several smart contracts: the vault (also called *anonymity provider*), Merkle Tree management, ERC-20 token, ZKP verifier, and an additional voting contract. The latter takes care of following the voting phases and counting the votes.

The voters' frontend cannot be used without further ado, as voters require software libraries to generate ZKPs and may download a web-based graphical user interface to interact with Tornado Vote. These software components must first be downloaded, although access to central infrastructures is not a problem here, because file signatures and on-chain proof verification can ensure correctness. Please note that we did not develop a graphical user interface but provide automated test cases which simulate user interactions. The tests cover a sample voting process and edge cases. To this end, they can be executed in a Truffle project environment on the local machine or an existing EVM network. Furthermore, since the Tornado Cash relayer infrastructure is not compatible with Tornado Vote, we provide a proof-of-concept relayer implementation for test purposes. All implementations are available on GitHub.[27]

Security

The security of Tornado Vote relies on various components. First of all, the Tornado Vote smart contracts are based on the original Tornado Cash smart contracts. Commissioned audits confirm the security of Tornado Cash's smart contracts, cryptography, and circuit system [@VK19; @KV19b; @KV19a]. For the security of Tornado Vote, we refer to these audits and therefore focus on our customizations for the voting process. To this end, we use security analysis tools and well-established security libraries. Additionally, we verify our voting process with a formal verification proof framework. In the following, we analyze and improve Tornado Vote's security and explain the technical measures in more detail.

Vulnerability Analysis Tools

To ensure that our smart contract customizations do not introduce security vulnerabilities, we use tools for automated code security analysis. Durieux, Ferreira, Abreu, and Cruz [Dur+20] analyzed different tools and concluded that *Mythril* [@Mue18] and *Slither* [FGG19] offer the best vulnerability detection abilities. We use both tools to detect known vulnerabilities; however, we point out that there may still be yet unknown and undetectable vulnerabilities. Additionally, smart contracts

27 https://github.com/robmuth/tornado-vote

are often so highly complex that the analysis of conditional branches and potentially many possible states cannot be performed in sufficient time. We therefore consider these tools as very useful and to some extent essential for secure smart contract programming. As a result, both tools confirm that our changes do not introduce new security vulnerabilities.

Formal Verification

Formal verification is a technique to prove that given rules and conditions are true for a given program, i.e., a smart contract in our case. For example, a condition that proves that a token smart contract cannot issue more tokens than specified at initialization. Thus, if the condition can be fulfilled, it is formally proven that no vulnerability exists to add new tokens arbitrarily. To do so, a formal verification tool must inspect every possible branch in the program logic and verify its states. However, such an excessive state inspection and verification can take a very long time, but if so, it eventually guarantees the condition.

For Tornado Vote, for example, we define conditions that there must be the correct number of votes after each submission. That is, by requiring that a voting counter always increases by one after submitting a vote, we prevent a vote from being ignored, counted twice, or leading to an integer overflow that would reset the counter to zero. To this end, we use formal verification similar to our unit-test cases but universal and without requiring implementation details.

Several formal verification tools and frameworks for smart contracts exist. We use Microsoft Research's VeriSol [Wan+19] since it is open-source, compatible with Solidity, and can be executed locally. VeriSol provides a library with additional assertion functions, and verifies all possible outcomes. In Listing 4, the first formal verification rule ensures that the number of possible votes always equals the number of minted voting tokens. To this end, this rule sums up all token balances and compares the sum with the total voting token supply. Additionally, we

```
// #Votes == #Tokens
VeriSol.ContractInvariant(VeriSol.SumMapping(_balances) == _
    totalSupply);

// Optional: Voters can only deposit tokens to Tornado Vote
// in function _beforeTokenTransfer(_from, _to, _id)
if (currentBlock >= commitPhaseBlock && currentBlock <=
    votingPhaseBlock) {
  assert(balanceOf(admin) == 0);
  assert(_to == tornadoVoteAddress);
}
```

Listing 4. Formal verification rule for VeriSol to verify that the number of submitted votes equals the total number of tokens, and an optional check to prevent token transfers to other accounts.

		Gas Costs			
Transaction	**Paid by**	**Min**	**Max**	**Mean**	**Stddev**
Deployments	Admin	8 192k	–	–	–
Token transfer	Admin	43k	58k	58k	15k
Approve	Voter	44k	44k	44k	0
Deposit	Voter	979k	1 000k	997k	21k
Commit	Voter	337k	337k	337k	0
Vote	Voter	35k	66k	50k	15k

Table 7. Gas costs for deployments and voting transactions.

developed several other assertions for all voting phases directly into Tornado Vote's smart contracts. While most rules can be successfully verified right away, others require constraining parameters or minor smart contract modifications due to code complexity. In summary, we developed and tested 13 conditions with 45 assumptions to verify, from which 11 conditions can be fully verified and 2 require minor smart contract changes. All rules and their verification results are available in our GitHub repository.

8.4 EVALUATION

Performance and costs are essential factors for voting systems, which in the case of blockchain-based voting lead to a feasibility trade-off. In the following, we analyze the feasibility of Tornado Vote for ideal best-case scenarios with our *Gas costs model* and then compare it with real-world conditions using our *residual capacities model*. To this end, we assume that all available block capacities are available for voting transactions and use Ethereum Mainnet data to evaluate realistic capacities. Since our residual capacities model disregards transaction costs, we develop an *economic model* to find a feasible trade-off between maximum performance and minimum costs, which does not cause the transaction costs to skyrocket but still keeps the overall duration reasonable. Importantly, all models are optimistic and should be considered a lower bound for performance and costs.

For clarification, we use the term *Gas costs* to refer to the technical transaction costs in Gas, i.e., the fixed costs due to the transaction's computations and storage usage, and *transaction costs* to refer to the final costs that the transaction sender must pay in Ether or U.S. Dollars (USD), respectively.

Gas Costs Model

We evaluate the Gas costs of each Tornado Vote transaction for multiple votes. We, therefore, deploy the smart contracts locally on Ganache and

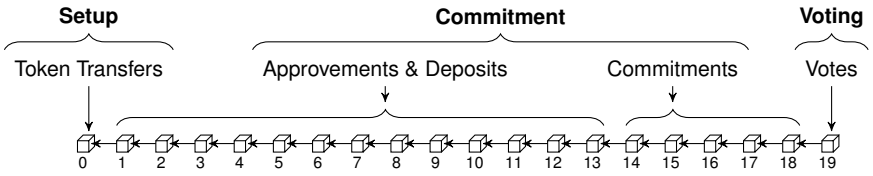


Figure 34. For $n = 180$ votes, Tornado Vote requires under idealistic conditions (Gas costs model) 20 blocks in Ethereum for all phases.

run exemplary voting test cases with up to $n = 250$ votes, as a greater n has no significant effect on the local Gas costs. As shown in Table 7, Gas costs for the one-time deployment remain constant for different n, while the transactions for registering eligible voters and submitting votes fluctuate during the following phases (cf. minimums, maximums, means, and the corresponding standard deviation). The Gas costs at the deposit are the most expensive voting transactions, because Tornado Cash internally updates a Merkle tree data structure, and, therefore, causes complex calculations and storage operations. Storage operations are also the main reason for fluctuating Gas costs because, for example, initial write operations to storage variables require more Gas than subsequent operations [Woo22].

Using the measured Gas costs, we can estimate the minimum number of blocks for submitting n votes with a static block size limit β (measured in Gas) as follows: $minBlocks(n, \beta) =$

$$\left\lceil \frac{n \cdot 58\text{k}}{\beta} \right\rceil + \left\lceil \frac{n \cdot (44\text{k} + 1\text{M})}{\beta} \right\rceil + \left\lceil \frac{n \cdot 337\text{k}}{\beta} \right\rceil + \left\lceil \frac{n \cdot 50\text{k}}{\beta} \right\rceil$$

The function basically adds the number of blocks in the various phases, or more precisely, the individual steps. To this end, we multiply the number of votes with the Gas costs from Table 7 and divide them by a static block size limit. We do this for each step in a voting process, i.e., transfer, approve, deposit, commitment, and vote. The approve and deposit steps can be combined and do not require to be finished in separate blocks. Since all other steps must be completed in separate blocks, we yield a total of four terms, which we have to round up to the next integer.

For example, a voting with $n = 180$ votes and a typical block size limit of $\beta = 15\text{M}$ Gas requires at least 20 blocks. In Figure 34, we also enumerate the number of required blocks for each of the phases and steps, respectively.

Residual Capacities Model

In this thesis, we specifically focus on the Ethereum blockchain due to its popularity and its usage of the Ethereum Virtual Machine (EVM), which is also compatible with many other blockchains, though. Since

each transaction requires computational resources and storage, transaction fees are required. The computational complexity to execute a transaction and the involved storage consumption is measured in the pseudo-unit *Gas* [Woo22]. In the past, transaction fees used an auction to determine Gas prices, which led to competition for available block capacity and therefore high fees. Recently, Ethereum introduced a new pricing mechanism to calculate Gas prices dynamically, namely EIP-1559 [@But+19]. It uses a congestion control mechanism that regulates a base fee based on available block capacities, which will be burned. Specifically, transaction fees for future blocks increase if the latest block contains "too many" transactions and decrease, respectively, if there is "enough" space up. As a reference for "too many" and "enough", EIP-1559 uses a target value of 15M Gas (as before), but now also supports temporary breaches up to 30M Gas to detect congestion.

Tornado Vote's practical feasibility is not only bound to the blockchain's maximum capacities but also to the blockchain's current workload. That is, third-party transactions lead to a fluctuating load, which changes from block to block. Accordingly, the amount of Gas available for the number of votes fluctuates as well. In order to capture this effect, we use a *residual capacities model* for the Ethereum Mainnet as in Chapter 4 to estimate the minimum number of blocks for n votes. Residual capacities are block capacities in Gas that remain unused after a block has been mined. That is, the transactions included in a block did not consume as much Gas as the actual block size limit allowed. We use residual capacities to quantify the number of voting transactions that we could add to blocks without exceeding Ethereum's capacities. Our model therefore sums up all residual capacities from a given point in time towards the past, and calculates how many transactions from Tornado Vote could have been processed despite existing background load.

Economic Model

As explained in Chapter 2, with the London Fork on the Ethereum Mainnet on the 5th of August, 2021, the EIP-1559 specification doubled the block size limit from $\approx$ 15M up to 30M Gas. The EIP-1559 protocol also introduced the so-called *base fee per Gas* to calculate how much Ether per Gas a transaction sender must pay as a minimum for each block. Basically, this base fee increases if the last block's size exceeded 15M Gas or decreases if it was less. However, EIP-1559 specifies that the base fee can only change by a maximum of $\pm 12.5\,\%$, even if the last block size has doubled or halved. This cap provides some confidence to transaction senders that their transactions will not be dropped from the mempool for the next few blocks as long as they provide sufficient Gas. If we, however, would fill up the residual Gas to the max ($\beta = 30$M Gas) with voting transactions as in our previous models, the base fee

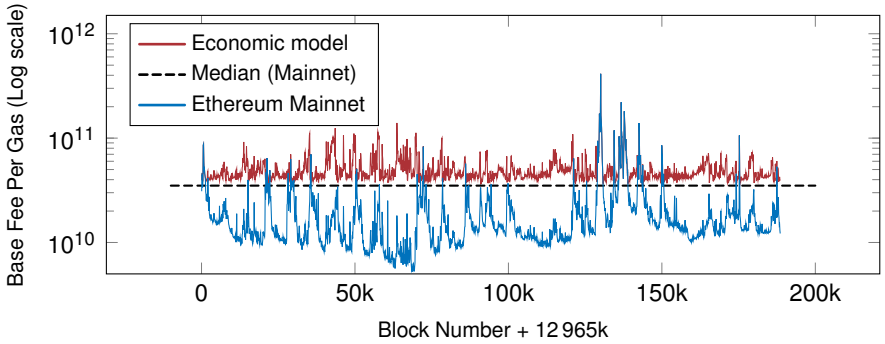


Figure 35. Base fee per Gas (y-axis) for blocks on the Ethereum Mainnet since EIP-1559 (x-axis) plotting historic Mainnet data and estimations according to the economic model.

would increase by 12.5 % each block. As a result, the base fee quickly reaches financially unfeasible amounts. Moreover, even if we utilize the residual Gas only up to $\beta = 15$M, the fees would never go down, but would increase every time third-party transactions exceed the threshold. This also leads to financially infeasible transaction fees, as a matter of course.

We therefore propose an *economic model* that only considers a block's residual capacities for voting transactions if the resulting base fee of the next block does not exceed a given threshold. To this end, we extend our analysis of the Ethereum Mainnet and analyze the past base fees since the introduction of EIP-1559. We show the base fee's development and median value in Figure 35. In the economic model, we consider residual capacities of a block only if its successor block's base fee does not exceed 35 $\frac{\text{Gwei}}{\text{Gas}}$. Otherwise, the block is omitted. By following our economic model, the base fee will not increase exponentially, but rather settles at the median.

Discussion

Our models allow us to estimate the number of blocks that Tornado Vote requires for n votes under different assumptions. To this end, we analyze Tornado Vote's performance with $n = 1$ to 10k votes, which we consider a reasonable number of votes for a voting DApp (cf. Chapter 4). Since the block Gas limit was doubled to 30M with EIP-1559, we evaluate our models with $\beta = 15$M and 30M Gas separately.

Figure 36 shows the expected trend that an increasing number of votes n (on the x-axis) leads to an increasing number of required blocks (on the y-axis). The Gas costs model constitutes a fundamental lower bound for Tornado Vote since it assumes that all capacities are exclusively available. It quantifies the limit of Tornado Vote in Ethereum, which is limited by Ethereum's transaction throughput. For instance, for $n = 10$k votes, an optimal coordinated voting with multiple votes per block requires at least 497 blocks with a block size limit of 30M Gas.

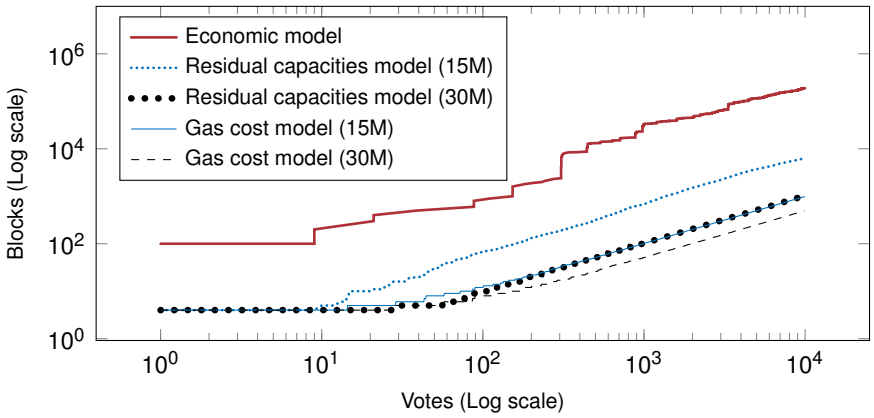


Figure 36. Minimum number of blocks (y-axis) for an increasing number of votes (x-axis) modelling different block Gas limits (historic data from the Ethereum Mainnet: 2022-11-16 to the past).

Multiplied by an average block generation time $\Delta = 15\,\text{s}$ per block, such a voting would require at least ≈ 2 hours. Accordingly, half the block size limit of 15M doubles the number of minimum blocks.

Since our Gas costs model ignores the blockchain's current workload, i.e., third-party transactions, we estimate the corresponding number of blocks with our residual capacities model, as well. For that, we measured the residual capacities of the Mainnet from 2022-11-16 until enough residual capacities for 10k votes accumulated. As shown in Figure 36, utilizing residual capacities therefore requires more blocks, as the blockchain capacities of the Ethereum Mainnet are no longer exclusively available. Interestingly, the residual capacities with 30M Gas (cf. bold dots) converge to the Gas costs model with 15M Gas. The reason for this is that EIP-1559 theoretically allows a block size limit of up to 30M Gas, but the protocol always aims for a block size of 15M through monetary incentives and penalties; hence, on average, there are always $\approx$ 15M Gas residual capacities that were unused. In the end, the residual capacities model requires at least $\approx$ 6k blocks for $n = 10$k votes with $\beta = 15$M and $\approx$ 1k blocks with $\beta = 30$M. In practice, however, these residual capacities cannot be used entirely because the base fee would skyrocket.

Considering the impact of third-party transactions on the base fee per Gas of EIP-1559, our economic model aims to keep the base fee at a financially reasonable threshold. As shown in Figure 35, we therefore analyzed the base fee for past transactions on the Ethereum Mainnet. Using the median base fee as threshold in our economic model, we will approximately maintain this base fee despite our voting transactions. Due to third-party transactions, however, the base fee can temporarily also exceed the median. Following the economic model further increases the number of minimum blocks, which can be seen in Figure 36 (cf. solid bold line). For instance, $n = 10$k votes require at least

189k blocks, which equals approximately 33 days with the same block generation rate Δ as before.

In order to get a sense of the final costs, we estimate the corresponding transaction costs in USD. Therefore, we take the median exchange rate of 2022, as before, which was 1 679 $\frac{\text{USD}}{\text{Ether}}$ [@Eth23]. Calculating the minimum Gas for a single vote with the Gas costs model, one vote costs approximately 2.43 USD. Taking the median base fee of 35 $\frac{\text{Gwei}}{\text{Gas}}$ into account as well, the costs for one vote increases to approximately 85.17 USD. This rather high price is, of course, only the case if the voting is completed in optimal time, but in any case, we can see that the costs are not increasing infinitely, as with the Gas costs model or residual capacities model.

Finally, we compare Tornado Vote's costs with the naive Tornado Cash approach (as explained in Section 8.3). To this end, we analyzed all past deposit and withdrawal transactions to Tornado Cash on the Ethereum Mainnet since EIP-1559 for the smallest denomination, i.e., 0.1 Ether. Considering the median of Gas costs for a deposit plus withdrawal transaction, a vote using Tornado Cash directly would cost approximately 1.26M Gas or 74.42 USD with the same Gas price and exchange rate as before. If we also consider the additional 0.1 Ether, which has to be sent to vote, the price per vote increases to approximately 242.32 USD. While the coin deposit covers the service fee for the relayer, we assume that for Tornado Vote the relayer service is provided for free, e.g., because the voting administrators have an intrinsic motivation for this.

Concluding, our model analyses revealed best-case capabilities of Tornado Vote. In practice, filling blocks with voting transactions en masse would have a significant and unpredictable impact on the whole blockchain, though. So, on the one hand, our models cannot predict accurate durations because unpredictable third-party transactions influence or would be influenced by such a large amount of voting transactions. On the other hand, however, we can use our models to assess lower-bound durations and costs.

8.5 CONCLUSION

We designed and implemented *Tornado Vote* as a novel privacy mechanism for blockchain-based voting to improve decision-making processes of DApps. Tornado Vote offers anonymous, fair, and practical voting with optional properties such as transferability and plural voting. As a building block, we adapted the well-established mixing protocol Tornado Cash to decouple voters' wallets from their votes. We used smart contract analyzing tools and formal verification to improve the reliability and security of our adaptations. In order to quantify the feasibility trade-off, we developed different evaluation models that yield Tornado Vote's limits and can help to adjust the trade-off for practical usages.

Part IV

CONCLUSIONS

9 DISCUSSION AND CONCLUSION

In this thesis, we explored the current technical state of decision-making DApps and showed that they manage significant financial funds. With BBBlockchain, we specifically explored the potential of blockchain use cases for urban participation. During the development and operation of BBBlockchain in two real-world building projects, we gained insights into reliable decision-making DApps with a special focus on the lack of privacy and transparency. We therefore investigated new privacy mechanisms specifically for decision-making DApps and contributed SmartDHX for establishing secure communication channels, an EVM-compatible CL-based anonymous credentials verifier, and Tornado Vote for anonymous voting. We see each of our contributions as a distinct component in advancing privacy for DApps. Taken as a whole, our contributions serve to provide answers to our initial research questions. In the following, we discuss the role of our contributions for decision-making DApps and recap our initial research questions in the conclusion of this thesis.

9.1 DISCUSSION

We have shown that decision-making has become a relevant blockchain use case for many popular applications, such as finance, DAO, and governance DApps. However, important privacy issues remain unaddressed or unresolved in real-world decision-making implementations. We therefore discuss our contributions to blockchain-based decision-making along with our privacy-related research focus.

PRIVACY As beneficial as transparency is for DApps, it potentially compromises user privacy or leads to dependencies on centralized, external infrastructures for hiding personal data from the blockchain. Today, many DApps rely on the personal data of their users, e.g., for KYC to prevent illegal financial transactions. To this end, at the beginning of this thesis, we asked how to implement appropriate privacy protection into decision-making DApps. Furthermore, along with our first research question, we are particularly interested in privacy-enhancing techniques and mechanisms to protect personal data that can be verified on-chain without exposing it to the public. Therefore, a DApp currently needs to externalize authentication processes, e.g., with verifiable off-chain computations using ZKPs, so that private data is not publicly exposed on the blockchain. Ultimately, it is important to find the right trade-offs between transparency, privacy, and reliability.

RELIABILITY Reliable decision-making requires trusted processes. This emphasizes the importance of reliable user authentication, for example, to prevent double voting or fraud. In particular, the pseudonymous identities of blockchains complicate the distinction between honest and fraudulent participants. While we provide a solution for privacy-preserving authentication with anonymous credentials, we cannot take the availability of a trusted identity provider for granted. We therefore envision our anonymous credentials verifier as a building block of a broader SSI paradigm. However, specific to blockchains, many dependencies still need to be resolved, such as technical dependencies on central infrastructures, e.g., for credential revocation.

BLOCKCHAIN-BASED VOTING Blockchains are particularly polarizing when it comes to voting. On the one hand, blockchains offer censorship resistance, transparency, and automated execution of the final result; on the other hand, blockchains require many compromises. We ruled out *real* elections for our decision-making use cases from the beginning, primarily because of severe scalability and security issues. Even for small-scale voting for decision-making, anonymity and reliable authentication cannot be implemented as easily as in centralized applications. In particular, privacy, transaction costs, and throughput render it technically difficult to implement. In the end, once again, it comes down to what trade-offs one is willing to make between transparency, privacy, and reliability.

DEMOCRATIC DECISION-MAKING Today's established DAOs do not suffer substantially from not fully identifying their users, and therefore do not face major privacy issues. Instead, they implement staking mechanisms, meaning that the more someone pays in, the more weight their vote has. However, there are decision-making use cases that need to be more democratic, for example, in urban building participation. This requires that such decision-making DApps reliably identify their users while protecting their privacy, e.g., during voting. Ultimately, we envision novel democratic use cases that could emerge in the future with a reliable solution for private user identification, e.g., democratically controlled DAOs.

COSTS Transaction costs are undeniably high when the blockchain is heavily loaded, or the transaction is more complex than a basic cryptocurrency transfer. The practicality of our contributions, especially on-chain anonymous credentials verification and voting with Tornado Vote, suffer from significantly high transaction costs. Specifically for decision-making DApps like BBBlockchain, we cannot assume that all users are willing to accept such high costs. While we have shown that blockchain developers are countering the intentional exploitation of high fees, better solutions are needed for real-world use. On the

one hand, one can expect that technological progress will eventually solve the problem as well, or consider alternative options such as SmartDHX for secure communication channels outside the blockchain. On the other hand, in the case of anonymous credentials, it should be considered whether the concept of CL-signatures verification is the way to follow, or whether other concepts are more appropriate. For example, Heiss et al. propose the concept of non-disclosing credentials on-chaining with ZoKrates [Hei+22], which also turns out to be more flexible for different authentication requirements.

9.2 CONCLUSION

In the following, we recap our initial research questions from the beginning of this thesis and discuss how our contributions have addressed them.

First, we asked how to implement decision-making DApps with on-chain privacy protection techniques. While transparency and trust are key properties of DApps that centralized solutions cannot achieve in the same way, the transparency also has its drawbacks, as it potentially compromises users' privacy. In fact, there are DApp use cases where user identification simply cannot be avoided, for example, to prevent double voting or money laundering with KYC procedures. Therefore, we contributed several solutions to address privacy-preserving authentication and anonymous voting for DApps. To this end, verifiable off-chain computation techniques emerged as a promising technique to achieve on-chain privacy. Thus, while private data remains off-chain, its correctness and validity can be verified using ZKP techniques and other privacy protection techniques. This allows decision-making DApps to keep their users private and deploy reliable processes. Overall, our contributions preserve the principles of a blockchain and its inherent transparency and trust.

The second research question of this thesis focused on how to keep as much cryptographic verification on-chain as possible while keeping private data off-chain. To this end, we argue that privacy can only be achieved by keeping private data off the blockchain and using verifiable off-chain computation to verify and process personal data. Indeed, there are several approaches to realize privacy in smart contracts, e.g., with homomorphic encryption [Ste+19; Ste+22] or trusted execution environments [Kar+21]. However, for our decision-making use case, which is primarily focused on privacy, transparency, and trust, we believe it is most important to eliminate TTPs and oracles to enable genuine DApps. Therefore, implementing a blockchain-based voting that is private for all participants except the administrator was not a suitable option to answer our research questions. Fortunately, SmartDHX and Tornado Vote offer privacy solutions that can include TTPs, but without compromising privacy or trust.

Additionally, we have shown with our BBBlockchain DApp that decision-making DApps face more than just technical challenges. While blockchain technologies promise more trust and transparency, they also increase complexity. Blockchains are based on complicated algorithms and mechanisms that are not yet understood by a broad audience, so we had to find solutions to make BBBlockchain as accessible as possible. We therefore had to find the right trade-offs between transparency, decentralization, and scalability. Thus, we had to find appropriate solutions for anonymous and reliable voting. Additionally, we had to keep scalability constraints in mind and minimize costs for the users. With SmartDHX, we hence contributed a solution for secure communication channels which can be established off-chain and therefore addresses common blockchain scalability issues, i.e., expensive transaction fees and limited throughput. In addition, with a novel anonymous credentials verifier for the EVM, we contributed a building block to reliably authenticate users without compromising their private, personal data. Finally, with Tornado Vote, we contributed an anonymous voting system with an established mixer service that anonymizes votes instead of cryptocurrency.

Overall, our contributions focused on minimizing dependencies on trusted third parties and implementing as much cryptography and verification routines in the smart contracts as possible. By processing private data only locally on users' devices with verifiable off-chain computation techniques, our contributions achieved on-chain privacy that does not depend on oracles. Ultimately, our contributions can also be used to improve privacy for all kinds of decision-making DApps and other permissionless blockchain applications as well.

PUBLICATIONS BY THE AUTHOR

[Bra+18] Samuel Brack, Robert Muth, Stefan Dietzel, and Björn Scheuermann. "Anonymous Datagrams over DNS Records." In: *Local Computer Networks Conference (LCN)*. IEEE, 2018, pp. 536–544.

[Bra+19] Samuel Brack, Robert Muth, Stefan Dietzel, and Björn Scheuermann. "Recommender Systems on Homomorphically Encrypted Databases for Enhanced User Privacy." In: *Local Computer Networks Conference (LCN) Symposium*. IEEE, 2019, pp. 74–82.

[GMF22] Marcel Gregoriadis, Robert Muth, and Martin Florian. "Analysis of Arbitrary Content on Blockchain-Based Systems using BigQuery." In: *Companion of The Web Conference*. WWW '22. ACM, 2022, pp. 478–487.

[Hei+22] Jonathan Heiss, Robert Muth, Frank Pallas, and Stefan Tai. "Non-disclosing Credential On-chaining for Blockchain-Based Decentralized Applications." In: *International Conference on Service-Oriented Computing (ICSOC)*. Vol. 13740. Lecture Notes in Computer Science. Springer, 2022, pp. 351–368.

[Iet+22] Beatrice Ietto, Kerstin Eisenhut, Robert Muth, Jochen Rabe, and Florian Tschorsch. "Transparency in Digital-Citizens Interfaces Through Blockchain Technology: BB-Blockchain for Participation Processes in Urban Planning." In: *European Technology and Engineering Management Summit (E-TEMS)*. IEEE, 2022, pp. 65–71.

[Iet+23] Beatrice Ietto, Jochen Rabe, Robert Muth, and Federica Pascucci. "Blockchain for citizens' participation in urban planning: the case of the city of Berlin. A value sensitive design approach." In: *Elsevier Cities Journal* 140 (2023), p. 104382.

[Mut+19] Robert Muth, Kerstin Eisenhut, Jochen Rabe, and Florian Tschorsch. "BBBlockchain: Blockchain-Based Participation in Urban Development." In: *International Conference on eScience*. IEEE, 2019, pp. 321–330.

[Mut+22a] Robert Muth, Tarek Galal, Jonathan Heiss, and Florian Tschorsch. "Towards Smart Contract-based Verification of Anonymous Credentials." In: *Financial Cryptography Workshop on Trusted Smart Contracts*. Vol. 13412. Lecture Notes in Computer Science. Springer, 2022, pp. 481–498.

[Mut+22b] Robert Muth, Beatrice Ietto, Kerstin Eisenhut, Jochen Rabe, and Florian Tschorsch. "Lessons Learned: Transparency in Urban Participation Utilizing Blockchains." In: *Eurasian Studies in Business and Economics*. In publication. Springer, 2022.

[MT20] Robert Muth and Florian Tschorsch. "SmartDHX: Diffie-Hellman Key Exchange with Smart Contracts." In: *International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE, 2020, pp. 164–168.

[MT21] Robert Muth and Florian Tschorsch. "Empirical Analysis of On-chain Voting with Smart Contracts." In: *Financial Cryptography Workshop on Trusted Smart Contracts*. Vol. 12676. Lecture Notes in Computer Science. Springer, 2021, pp. 397–412.

[MT23] Robert Muth and Florian Tschorsch. "Tornado Vote: Anonymous Blockchain-Based Voting." In: *International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2023, pp. 1–9.

[Rab+21] Jochen Rabe, Beatrice Ietto, Robert Muth, Kerstin Eisenhut, and Federica Pascucci. "Citizens' Engagement in Urban Development through Blockchain: a Human-centered Design Approach." In: *International Conference on Technology Management, Operations and Decisions (ICTMOD)*. IEEE, 2021, pp. 1–6.

BIBLIOGRAPHY

[AKW20] Yousif Abuidris, Rajesh Kumar, and Wang Wenyong. "A Survey of Blockchain Based on E-Voting Systems." In: *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*. ICBTA '19. Xi'an, China: Association for Computing Machinery, 2020, pp. 99–104.

[AM17] Maher Alharby and Aad van Moorsel. "Blockchain-based Smart Contracts: A Systematic Mapping Study." In: *CoRR* abs/1710.06372 (2017).

[AH12] Phil Allmendinger and Graham Haughton. "Post-political spatial planning in England: a crisis of consensus?" In: *Transactions of the Institute of British Geographers* (2012).

[Amn06] Erik Amn. "Playing with fire? Swedish mobilization for participatory democracy." In: *Journal of European Public Policy* (2006).

[Arm+15] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. "A Guide to Fully Homomorphic Encryption." In: *IACR Cryptol. ePrint Arch.* (2015), p. 1192.

[Arn69] Sherry R Arnstein. "A ladder of citizen participation." In: *Journal of the American Institute of planners* (1969).

[ABC17] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. "A Survey of Attacks on Ethereum Smart Contracts (SoK)." In: *POST*. Vol. 10204. Lecture Notes in Computer Science. Springer, 2017, pp. 164–186.

[BC11] Frank Bannister and Regina Connolly. "The trouble with transparency: a critical review of openness in e-government." In: *Policy & Internet* (2011).

[Bar03] Benjamin Barber. *Strong democracy: Participatory politics for a new age*. Univ of California Press, 2003.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. "Pairing-Friendly Elliptic Curves of Prime Order." In: *Selected Areas in Cryptography*. Vol. 3897. Lecture Notes in Computer Science. Springer, 2005, pp. 319–331.

[BCL21] Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. "SoK: Lending Pools in Decentralized Finance." In: *Financial Cryptography Workshops*. Vol. 12676. Lecture Notes in Computer Science. Springer, 2021, pp. 553–578.

[Bér+21] Ferenc Béres, István András Seres, András A. Benczúr, and Mikerah Quintyne-Collins. "Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users." In: *DAPPS*. IEEE, 2021, pp. 69–78.

[Bis+17] Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, and Francesco Santini. "An end-to-end voting-system based on bitcoin." In: *SAC*. ACM, 2017, pp. 1836–1841.

[Boe88] Bert den Boer. "Diffie-Hellman is as Strong as Discrete Log for Certain Primes." In: *CRYPTO*. Vol. 403. Lecture Notes in Computer Science. Springer, 1988, pp. 530–539.

[BV16] Wouter Bokslag and Manon de Vries. "Evaluating e-voting: theory and practice." In: *CoRR* abs/1602.02509 (2016).

[BB08] Dan Boneh and Xavier Boyen. "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups." In: *J. Cryptol.* 21.2 (2008), pp. 149–177.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures." In: *CRYPTO*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 41–55.

[BGB04] Nikita Borisov, Ian Goldberg, and Eric A. Brewer. "Off-the-record communication, or, why not to use PGP." In: *WPES*. ACM, 2004, pp. 77–84.

[Bun17] Bundesministerium der Justiz und für verbraucherschutz. *§ 3 BauGB - Beteiligung der Öffentlichkeit*. 2017.

[BET21] Anselm Busse, Jacob Eberhardt, and Stefan Tai. "EVM-Perf: High-Precision EVM Performance Analysis." In: *IEEE ICBC*. IEEE, 2021, pp. 1–8.

[Cab17] Yves Cabannes. "Participatory budgeting in Paris: Act, reflect, grow." In: *Another city is possible with participatory budgeting* 179 (2017), p. 203.

[CE21] Giulio Caldarelli and Joshua Ellul. "The Blockchain Oracle Problem in Decentralized Finance—A Multivocal Approach." In: *Applied Sciences* 11.16 (2021).

[CG12] Jan Camenisch and Thomas Groß. "Efficient Attributes for Anonymous Credentials." In: *ACM Trans. Inf. Syst. Secur.* 15.1 (2012), 4:1–4:30.

[CKS09] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. "An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials." In: *Public Key Cryptography*. Vol. 5443. Lecture Notes in Computer Science. Springer, 2009, pp. 481–500.

[CL01] Jan Camenisch and Anna Lysyanskaya. "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation." In: *EUROCRYPT*. Vol. 2045. Lecture Notes in Computer Science. Springer, 2001, pp. 93–118.

[CL02] Jan Camenisch and Anna Lysyanskaya. "A Signature Scheme with Efficient Protocols." In: *SCN*. Vol. 2576. Lecture Notes in Computer Science. Springer, 2002, pp. 268–289.

[Cen+21] Piera Centobelli, Roberto Cerchione, Emilio Esposito, and Eugenio Oropallo. "Surfing blockchain wave, or drowning? Shaping the future of distributed ledgers and decentralized technologies." In: *Technological Forecasting and Social Change* (2021).

[Cor+23] Mikel Cortes-Goicoechea, Tarun Mohandas-Daryanani, Jose Luis Muñoz-Tapia, and Leonardo Bautista-Gomez. "Autopsy of Ethereum's Post-Merge Reward System." In: *CoRR* abs/2303.09850 (2023).

[Cro+16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed E. Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. "On Scaling Decentralized Blockchains - (A Position Paper)." In: *Financial Cryptography Workshops*. Vol. 9604. Lecture Notes in Computer Science. Springer, 2016, pp. 106–125.

[Dam98] Ivan Damgrd. "Commitment Schemes and Zero-Knowledge Protocols." In: *Lectures on Data Security*. Vol. 1561. Lecture Notes in Computer Science. Springer, 1998, pp. 63–86.

[DT22] Erik Daniel and Florian Tschorsch. "IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks." In: *IEEE Commun. Surv. Tutorials* 24.1 (2022), pp. 31–52.

[DMJ18] Nina David, John G McNutt, and Jonathan B Justice. "Smart cities, transparency, civic technology and reinventing government." In: *Smart Technologies for Smart Governments*. Springer, 2018, pp. 19–34.

[DKR10] Stéphanie Delaune, Steve Kremer, and Mark Ryan. "Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster." In: *Towards Trustworthy Elections*. Vol. 6000. Lecture Notes in Computer Science. Springer, 2010, pp. 289–309.

[DH76] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography." In: *IEEE Trans. Inf. Theory* 22.6 (1976), pp. 644–654.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. "Tor: The Second-Generation Onion Router." In: *USENIX Security Symposium*. USENIX, 2004, pp. 303–320.

[Dou02] John R Douceur. "The sybil attack." In: *International workshop on peer-to-peer systems*. Springer. 2002, pp. 251–260.

[Dry02] John S Dryzek. *Deliberative democracy and beyond: Liberals, critics, contestations*. Oxford University Press on Demand, 2002.

[Dur+20] Thomas Durieux, Joao F. Ferreira, Rui Abreu, and Pedro Cruz. "Empirical review of automated analysis tools on 47, 587 Ethereum smart contracts." In: *ICSE*. ACM, 2020, pp. 530–541.

[EH18] Jacob Eberhardt and Jonathan Heiss. "Off-chaining Models and Approaches to Off-chain Computations." In: *SERIAL*. ACM, 2018, pp. 7–12.

[ET18] Jacob Eberhardt and Stefan Tai. "ZoKrates - Scalable Privacy-Preserving Off-Chain Computations." In: *iThings/GreenCom/CPSCom/SmartData*. IEEE, 2018, pp. 1084–1091.

[EAA22] Mounir El Khatib, Asma Al Mulla, and Wadha Al Ketbi. "The Role of Blockchain in E-Governance and Decision-Making in Project and Program Management." In: *Advances in Internet of Things* 12.3 (2022), pp. 88–109.

[ECW21] Shawn M Emery, C Edward Chow, and Richard White. "Penetration Testing a US Election Blockchain Prototype." In: *E-Vote-ID 2021* (2021), p. 82.

[EFS21] Erik Eriksson, Amira Fredriksson, and Josefina Syssner. "Opening the black box of participatory planning: a study of how planners handle citizens' input." In: *European Planning Studies* (2021).

[Erw+20] Andreas Erwig, Sebastian Faust, Siavash Riahi, and Tobias Stöckert. "CommiTEE: An Efficient and Secure Commit-Chain Protocol using TEEs." In: *IACR Cryptol. ePrint Arch.* (2020), p. 1486.

[EMC20] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. "Sok: Transparent dishonesty: front-running attacks on blockchain." In: *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC*. Springer. 2020, pp. 170–189.

[FGG19] Josselin Feist, Gustavo Grieco, and Alex Groce. "Slither: a static analysis framework for smart contracts." In: *WETSEB ICSE*. IEEE, 2019, pp. 8–15.

[FS86] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *CRYPTO*. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194.

[FFB19] Michael Fröwis, Andreas Fuchs, and Rainer Böhme. "Detecting Token Systems on Ethereum." In: *Financial Cryptography*. Vol. 11598. Lecture Notes in Computer Science. Springer, 2019, pp. 93–112.

[GAC19] David Gabay, Kemal Akkaya, and Mumin Cebe. "A Privacy Framework for Charging Connected Electric Vehicles Using Blockchain and Zero Knowledge Proofs." In: *LCN Symposium*. IEEE, 2019, pp. 66–73.

[GY19] Hisham S. Galal and Amr M. Youssef. "Trustee: Full Privacy Preserving Vickrey Auction on Top of Ethereum." In: *Financial Cryptography Workshops*. Vol. 11599. Lecture Notes in Computer Science. Springer, 2019, pp. 190–207.

[Gar+19] Kanika Garg, Pavi Saraswat, Sachin Bisht, Sahil Kr Aggarwal, Sai Krishna Kothuri, and Sahil Gupta. "A comparitive analysis on e-voting system using blockchain." In: *4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*. IEEE. 2019, pp. 1–4.

[GM13] John Gaventa and Rosemary McGee. "The impact of transparency and accountability initiatives." In: *Development Policy Review* (2013).

[GW12] Stephan G Grimmelikhuijsen and Eric W Welch. "Developing and testing a theoretical framework for computer-mediated transparency of local governments." In: *Public administration review* (2012).

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments." In: *EUROCRYPT (2)*. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 305–326.

[HS91] Stuart Haber and W Scott Stornetta. *How to time-stamp a digital document*. Springer, 1991.

[Hab15] UN Habitat. "International guidelines on urban and territorial planning." In: *United Nations Human Settlements Programme* (2015).

[Har+18] Freya Sheer Hardwick, Apostolos Gioulis, Raja Naeem Akram, and Konstantinos Markantonakis. "E-Voting With Blockchain: An E-Voting Protocol with Decentralisation and Voter Privacy." In: *iThings/GreenCom/CPSCom/SmartData*. IEEE, 2018, pp. 1561–1567.

[Hei+18] Sven Heiberg, Ivo Kubjas, Janno Siim, and Jan Willemson. "On Trade-offs of Applying Block Chains for Electronic Voting Bulletin Boards." In: *E-Vote-ID* (2018), p. 259.

[HET19] Jonathan Heiss, Jacob Eberhardt, and Stefan Tai. "From Oracles to Trustworthy Data On-Chaining Systems." In: *Blockchain*. IEEE, 2019, pp. 496–503.

[HHK18] Ryan Henry, Amir Herzberg, and Aniket Kate. "Blockchain Access Privacy: Challenges and Directions." In: *IEEE Secur. Priv.* 16.4 (2018), pp. 38–45.

[Hja+18] Friorik P. Hjalmarsson, Gunnlaugur K. Hreioarsson, Mohammad Hamdaqa, and Gísli Hjálmtýsson. "Blockchain-Based E-Voting System." In: *IEEE CLOUD*. IEEE Computer Society, 2018, pp. 983–986.

[Kar+21] Rabimba Karanjai, Weidong Shi, Lei Xu, Lin Chen, Fengwei Zhang, and Zhimin Gao. "Lessons Learned from Blockchain Applications of Trusted Execution Environments and Implications for Future Research." In: *HASP@MICRO*. ACM, 2021, 5:1–5:8.

[Kil+22] Christian Killer, Moritz Eck, Bruno Rodrigues, Jan von der Assen, Roger Staubli, and Burkhard Stiller. "ProvotuMN: Decentralized, Mix-Net-based, and Receipt-free Voting System." In: *ICBC*. IEEE, 2022, pp. 1–9.

[KL14] Natalia Kogan and Kyoung Jun Lee. "Exploratory research on the success factors and challenges of Smart City projects." In: *Asia pacific journal of information systems* (2014).

[KDF13] Joshua A Kroll, Ian C Davey, and Edward W Felten. "The economics of Bitcoin mining, or Bitcoin in the presence of adversaries." In: *Proceedings of WEIS*. 11. Washington, DC. 2013.

[KV18] Nir Kshetri and Jeffrey M. Voas. "Blockchain-Enabled E-Voting." In: *IEEE Softw.* 35.4 (2018), pp. 95–99.

[Len+22] Jiewu Leng, Man Zhou, J. Leon Zhao, Yongfeng Huang, and Yiyang Bian. "Blockchain Security: A Survey of Techniques and Research Directions." In: *IEEE Trans. Serv. Comput.* 15.4 (2022), pp. 2490–2510.

[MSS22] Diksha Malhotra, Poonam Saini, and Awadhesh Kumar Singh. "How Blockchain Can Automate KYC: Systematic Review." In: *Wirel. Pers. Commun.* 122.2 (2022), pp. 1987–2021.

[Mat+18] Roman Matzutt, Jens Hiller, Martin Henze, Jan Henrik Ziegeldorf, Dirk Müllmann, Oliver Hohlfeld, and Klaus Wehrle. "A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin." In: *Financial Cryptography*. Vol. 10957. Lecture Notes in Computer Science. Springer, 2018, pp. 420–438.

[McC+15] Patrick McCorry, Siamak F Shahandashti, Dylan Clarke, and Feng Hao. "Authenticated key exchange over bitcoin." In: *International Conference on Research in Security Standardisation*. Springer. 2015, pp. 3–20.

[MSH17] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. "A Smart Contract for Boardroom Voting with Maximum Voter Privacy." In: *Financial Cryptography*. Vol. 10322. Lecture Notes in Computer Science. Springer, 2017, pp. 357–375.

[MS19] Julia Meier and Benedikt Schuppli. "The DAO Hack and the Living Law of Blockchain." In: *Digitalisierung–Gesellschaft–Recht: Analysen und Perspektiven von Assistierenden des Rechtswissenschaftlichen Instituts der Universität Zürich* (2019), pp. 27–43.

[Met20] William Metcalfe. "Ethereum, Smart Contracts, DApps." In: *Blockchain and Crypt Currency* 77 (2020).

[Mil19] Andrew Miller. "Permissioned and permissionless blockchains." In: *Blockchain for distributed systems security* (2019), pp. 193–204.

[Mil85] Victor S. Miller. "Use of Elliptic Curves in Cryptography." In: *CRYPTO*. Vol. 218. Lecture Notes in Computer Science. Springer, 1985, pp. 417–426.

[MM18] Debajani Mohanty and Debajani Mohanty. "Ethereum use cases." In: *Ethereum for Architects and Developers: With Case Studies and Code Samples in Solidity* (2018), pp. 203–243.

[MK12] Valeria Monno and Abdul Khakee. "Tokenism or political activism? Some reflections on participatory planning." In: *International Planning Studies* (2012).

[Mös+18] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. "An Empirical Analysis of Traceability in the Monero Blockchain." In: *Proc. Priv. Enhancing Technol.* 2018.3 (2018), pp. 143–163.

[Müh+20] Roman Mühlberger, Stefan Bachhofner, Eduardo Castelló Ferrer, Claudio Di Ciccio, Ingo Weber, Maximilian Wöhrer, and Uwe Zdun. "Foundational Oracle Patterns: Connecting Blockchain to the Off-Chain World." In: *BPM (Blockchain and RPA Forum)*. Vol. 393. Lecture Notes in Business Information Processing. Springer, 2020, pp. 35–51.

[Müh+18] Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. "A survey on essential components of a self-sovereign identity." In: *Comput. Sci. Rev.* 30 (2018), pp. 80–86.

[NJ20] Nitin Naik and Paul Jenkins. "uPort Open-Source Identity Management System: An Assessment of Self-Sovereign Identity and User-Centric Data Platform Built on Blockchain." In: *ISSE*. IEEE, 2020, pp. 1–7.

[Nak08] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.

[Nat18] National Academies of Sciences, Engineering, and Medicine and others. "Securing the Vote: Protecting American Democracy." In: (2018), pp. 103–105.

[Par+21] Sunoo Park, Michael A. Specter, Neha Narula, and Ronald L. Rivest. "Going from bad to worse: from Internet voting to blockchain voting." In: *J. Cybersecur.* 7.1 (2021).

[PA19] Gabriel Piña and Claudia Avellaneda. "Central government strategies to promote local governments' transparency: Guidance or enforcement?" In: *Public Performance & Management Review* (2019).

[Pin+19] Andrea Pinna, Simona Ibba, Gavina Baralla, Roberto Tonelli, and Michele Marchesi. "A Massive Analysis of Ethereum Smart Contracts Empirical Study and Code Metrics." In: *IEEE Access* 7 (2019), pp. 78194–78213.

[Pop19] Serguei Popov. "IOTA: Feeless and Free." In: *Blockchain Technical Briefs* (2019).

[Qin+21] Kaihua Qin, Liyi Zhou, Yaroslav Afonin, Ludovico Lazzaretti, and Arthur Gervais. "CeFi vs. DeFi - Comparing Centralized to Decentralized Finance." In: *CoRR* abs/2106.08157 (2021).

[Ran+19] Nripendra P Rana, Sunil Luthra, Sachin Kumar Mangla, Rubina Islam, Sian Roderick, and Yogesh K Dwivedi. "Barriers to the development of smart cities in Indian context." In: *Information Systems Frontiers* (2019).

[RYM19] Pierre Reibel, Haaroon Yousaf, and Sarah Meiklejohn. "Short Paper: An Exploration of Code Diversity in the Cryptocurrency Landscape." In: *Financial Cryptography*. Vol. 11598. Lecture Notes in Computer Science. Springer, 2019, pp. 73–83.

[RH11] Fergal Reid and Martin Harrigan. "An Analysis of Anonymity in the Bitcoin System." In: *SocialCom/PASSAT*. IEEE Computer Society, 2011, pp. 1318–1326.

[Rou20] Tim Roughgarden. "Transaction Fee Mechanism Design for the Ethereum Blockchain: An Economic Analysis of EIP-1559." In: *CoRR* abs/2012.00854 (2020).

[RMK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. "CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin." In: *ESORICS*. LNCS. Springer, 2014.

[Rum15] Rebecca Rumbul. "Who benefits from civic technology." In: *Demographic and public attitudes research into the users of civic technologies* 7 (2015), p. 2018.

[Ryd11] Yvonne Rydin. *The purpose of planning: Creating sustainable towns and cities*. Policy Press, 2011.

[Sal+19] Jorge Saldivar, Cristhian Parra, Marcelo Alcaraz, Rebeca Arteta, and Luca Cernuzzi. "Civic Technology for Social Innovation." In: *Computer Supported Cooperative Work (CSCW)* (2019).

[Sch21] Fabian Schär. "Decentralized Finance: On Blockchain- and Smart Contract-based Financial Markets." In: *FRB of St. Louis Review* (2021).

[Sch91] Claus Schnorr. "Efficient signature generation by smart cards." In: *Journal of Cryptology* 4 (Jan. 1991), pp. 161–174.

[SGY20] Mohamed Seifelnasr, Hisham S. Galal, and Amr M. Youssef. "Scalable Open-Vote Network on Ethereum." In: *Financial Cryptography Workshops*. Vol. 12063. Lecture Notes in Computer Science. Springer, 2020, pp. 436–450.

[SSH21] Alesja Serada, Tanja Sihvonen, and J. Tuomas Harviainen. "*CryptoKitties* and the New Ludic Economy: How Blockchain Introduces Value, Ownership, and Scarcity in Digital Gaming." In: *Games Cult.* 16.4 (2021), pp. 457–480.

[SHS20] Bhavye Sharma, Raju Halder, and Jawar Singh. "Blockchain-based Interoperable Healthcare using Zero-Knowledge Proofs and Proxy Re-Encryption." In: *COMSNETS*. IEEE, 2020, pp. 1–6.

[Sim04] Barbara B. Simons. "Electronic voting systems: the good, the bad, and the stupid." In: *ACM Queue* 2.7 (2004), pp. 20–26.

[SNA21] Reza Soltani, Uyen Trang Nguyen, and Aijun An. "A Survey of Self-Sovereign Identity Ecosystem." In: *Secur. Commun. Networks* 2021 (2021), 8873429:1–8873429:26.

[SFG19] Michael Spain, Sean Foley, and Vincent Gramoli. "The Impact of Ethereum Throughput and Fees on Transaction Latency During ICOs." In: *Tokenomics*. OASIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[SKW20] Michael A Specter, James Koppel, and Daniel Weitzner. "The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in US. Federal Elections." In: *29th USENIX Security Symposium (USENIX Security 20)*. 2020, pp. 1535–1553.

[Ste+22] Samuel Steffen, Benjamin Bichsel, Roger Baumgartner, and Martin T. Vechev. "ZeeStar: Private Smart Contracts by Homomorphic Encryption and Zero-knowledge Proofs." In: *IEEE Symposium on Security and Privacy*. IEEE, 2022, pp. 179–197.

[Ste+19] Samuel Steffen, Benjamin Bichsel, Mario Gersbach, Noa Melchior, Petar Tsankov, and Martin T. Vechev. "zkay: Specifying and Enforcing Data Privacy in Smart Contracts." In: *CCS*. ACM, 2019, pp. 1759–1776.

[Swa18] Melanie Swan. "Blockchain for business: Next-generation enterprise artificial intelligence systems." In: (2018).

[Sza97] Nick Szabo. "Formalizing and Securing Relationships on Public Networks." In: *First Monday* 2.9 (1997).

[Ten+22] Huang Teng, Wayneyuan Tian, Haocheng Wang, and Zhiyuan Yang. "Applications of the Decentralized Finance (DeFi) on the Ethereum." In: *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*. 2022, pp. 573–578.

[TFH17] Haibo Tian, Liqing Fu, and Jiejie He. "A Simpler Bitcoin Voting Protocol." In: *Inscrypt*. Vol. 10726. Lecture Notes in Computer Science. Springer, 2017, pp. 81–98.

[TS16] Florian Tschorsch and Björn Scheuermann. "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies." In: *IEEE Commun. Surv. Tutorials* 18.3 (2016), pp. 2084–2123.

[Vic20] Friedhelm Victor. "Address Clustering Heuristics for Ethereum." In: *Financial Cryptography*. Vol. 12059. Lecture Notes in Computer Science. Springer, 2020, pp. 617–633.

[VL19] Friedhelm Victor and Bianca Katharina Lüders. "Measuring Ethereum-Based ERC20 Token Networks." In: *Financial Cryptography*. Vol. 11598. Lecture Notes in Computer Science. Springer, 2019, pp. 113–129.

[Wan+19] Yuepeng Wang, Shuvendu K. Lahiri, Shuo Chen, Rong Pan, Isil Dillig, Cody Born, Immad Naseer, and Kostas Ferles. "Formal Verification of Workflow Policies for Smart Contracts in Azure Blockchain." In: *VSTTE*. Vol. 12031. Lecture Notes in Computer Science. Springer, 2019, pp. 87–106.

[WD13] Joachim Wehner and Paolo De Renzio. "Citizens, legislators, and executive disclosure: The political determinants of fiscal transparency." In: *World Development* (2013).

[WMP22] Bryan White, Aniket Mahanti, and Kalpdrum Passi. "Characterizing the OpenSea NFT Marketplace." In: *WWW (Companion Volume)*. ACM, 2022, pp. 488–496.

[Whi96] Sarah C White. "Depoliticising development: the uses and abuses of participation." In: *Development in practice* (1996).

[WTC19] Alexander Wilson, Mark Tewdwr-Jones, and Rob Comber. "Urban planning, public participation and digital technology: App development as a method of generating citizen involvement in local planning processes." In: *Environment and Planning B: Urban Analytics and City Science* 46.2 (2019), pp. 286–302.

[Woo22] Gavin Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger, Berlin Revision BEACFBD*. 2022.

[Wu19] Kaidong Wu. "An Empirical Study of Blockchain-based Decentralized Applications." In: *CoRR* abs/1902.04969 (2019).

[Xia+21] Pengcheng Xia, Haoyu Wang, Zhou Yu, Xinyu Liu, Xiapu Luo, and Guoai Xu. "Ethereum Name Service: the Good, the Bad, and the Ugly." In: *CoRR* abs/2104.05185 (2021).

[Yan+19] Jinhong Yang, Md Mehedi Hassan Onik, Nam-Yong Lee, Mohiuddin Ahmed, and Chul-Soo Kim. "Proof-of-familiarity: a privacy-preserved blockchain scheme for collaborative medical decision-making." In: *Applied Sciences* 9.7 (2019), p. 1370.

[Yu+18] Bin Yu, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. "Platform-Independent Secure Blockchain-Based Voting System." In: *ISC*. Vol. 11060. Lecture Notes in Computer Science. Springer, 2018, pp. 369–386.

[ZC15] Zhichao Zhao and T.-H. Hubert Chan. "How to Vote Privately Using Bitcoin." In: *ICICS*. Vol. 9543. Lecture Notes in Computer Science. Springer, 2015, pp. 82–96.

[Zhe+18] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. "Blockchain challenges and opportunities: a survey." In: *Int. J. Web Grid Serv.* 14.4 (2018), pp. 352–375.

[Zio+19] Rafael Ziolkowski, Geetha Parangi, Gianluca Miscione, and Gerhard Schwabe. "Examining Gentle Rivalry: Decision-Making in Blockchain Systems." In: *HICSS*. ScholarSpace, 2019, pp. 1–10.

WEB RESOURCES

[@Akh+19] Alexey Akhunov, Eli Ben Sasson, Tom Brand, Louis Guthmann, and Avihu Levy. *Transaction data gas cost reduction*. Accessed: 2023-05-14. 2019. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2028.md.

[@Ale19] Roman Storm Alexey Pertsev Roman Semenov. *Tornado Cash Privacy Solution Version 1.4*. Accessed: 2023-01-17. 2019. URL: https://berkeley-defi.github.io/assets/material/Tornado%20Cash%20Whitepaper.pdf.

[@But+19] EIP-1559: Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. *Fee market change for ETH 1.0 chain*. Accessed: 2023-01-17. 2019. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md.

[@But16] Vitalik Buterin. *Gas cost changes for IO-heavy operations*. Accessed: 2023-05-14. 2016. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-150.md.

[@But17a] Vitalik Buterin. *EIP-198: Big integer modular exponentiation*. Accessed: 2023-01-17. 2017. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-198.md.

[@But17b] Vitalik Buterin. *Proof of Stake FAQ*. Accessed: 2023-05-22. 2017. URL: https://vitalik.ca/general/2017/12/31/pos_faq.html.

[@But20] Vitalik Buterin. *Why Proof of Stake*. Accessed: 2023-05-22. 2020. URL: https://vitalik.ca/general/2020/11/06/pos2020.html.

[@But21] Vitalik Buterin. *Blockchain voting is overrated among uninformed people but underrated among informed people*. Accessed: 2023-01-17. 2021. URL: https://vitalik.ca/general/2021/05/25/voting2.html.

[@But23a] Vitalik Buterin. *An incomplete guide to stealth addresses*. Accessed: 2023-05-12. 2023. URL: https://vitalik.ca/general/2023/01/20/stealth.html.

[@But23b] Vitalik Buterin. *Ethereum whitepaper: A next generation smart contract and decentralized application platform*. Accessed: 2023-01-17. 2023. URL: https://ethereum.org/en/whitepaper/.

[@BS20] Vitalik Buterin and Martin Swende. *Gas cost increases for state access opcodes*. Accessed: 2023-05-14. 2020. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2929.md.

[@Chr16] Christopher Allen. *The path to self-sovereign identity*. Accessed: 2023-01-17. 2016. URL: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[@Coi23] CoinMarketCap.com. *Ethereum Market Cap*. Accessed: 2023-05-13. 2023. URL: https://coinmarketcap.com/currencies/ethereum/.

[@Con18] ConsenSys. *Quorum Whitepaper v0.2*. Accessed: 2023-01-16. 2018. URL: https://github.com/ConsenSys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf.

[@Dal19] Brady Dale. *Stellar Tried to Give Away 2B XLM Tokens on Keybase. Then the Spammers Came*. Accessed: 2023-05-10. 2019. URL: https://web.archive.org/web/20220118130355/https://www.coindesk.com/business/2019/12/13/stellar-tried-to-give-away-2b-xlm-tokens-on-keybase-then-the-spammers-came/.

[@Deg+17] Degewo, Howoge, Gesobau, Gewobag, Stadt&Land, and WBM. *Leitlinien für Partizipation im Wohnungsbau*. Accessed: 2023-01-16. 2017. URL: http://web.archive.org/web/20220207050753/https://www.degewo.de/fileadmin/user_upload/degewo/Wachstum/Partizipation_Unterseiten/Broschuere_Evaluation_Leitlinien_Wohnungsbau.pdf.

[@DD18] Evan Duffield and Daniel Diaz. *Dash: A Payments-Focused Cryptocurrency*. Accessed: 2023-01-13. 2018. URL: https://github.com/dashpay/dash/wiki/Whitepaper.

[@Est21] Brainbot Labs Est. *Raiden Network 3.0.1 Documentation*. Accessed: 2023-05-12. 2021. URL: https://raiden-network.readthedocs.io/en/v3.0.1/.

[@Eth23] Etherscan.io. *Ether Daily Price (USD) Chart*. Accessed: 2023-04-18. 2023. URL: https://etherscan.io/chart/etherprice.

[@Fou22a] Democracy Earth Foundation. *The Social Smart Contract*. Accessed: 2023-01-16. 2022. URL: https://github.com/DemocracyEarth/paper.

[@Fou22b] Hyperledger Foundation. *Hyperledger Indy SDK*. Accessed: 2023-05-17. 2022. URL: https://github.com/hyperledger/indy-sdk.

[@Hen+16] J Hendler, O Sosnik, G Prudner, D Chernicoff, J McCarthy, O Ree, and C Keefe. *Engines of Change: What Civic Tech Can Learn from Social Movements*. Accessed: 2023-01-16. 2016. URL: http://web.archive.org/web/20170113162056mp_/http://enginesofchange.omidyar.com/docs/OmidyarEnginesOfChange.pdf.

[@Hop+16] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. *Zcash protocol specification*. Accessed: 2023-05-12. 2016. URL: https://github.com/zcash/zips/blob/main/protocol/protocol.pdf.

[@Hyp18] Hyperledger White Paper Working Group. *An Introduction to Hyperledger*. Accessed: 2023-01-17. 2018. URL: https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf.

[@IAP14] IAP2 International Federation. *IAP2's Public Participation Spectrum*. Accessed: 2023-01-16. 2014. URL: http://web.archive.org/web/20190319061606/iap2.org.au/Tenant/C0000004/00000001/files/IAP2_Public_Participation_Spectrum.pdf.

[@Ide19] Iden3. *Pedersen Hash*. Accessed: 2023-01-17. 2019. URL: https://iden3-docs.readthedocs.io/en/latest/iden3_repos/research/publications/zkproof-standards-workshop-2/pedersen-hash/pedersen.html.

[@Jen16] Christoph Jentzsch. *Decentralized Autonomous Organization to Automate Governance*. Accessed: 2023-01-13. 2016. URL: http://web.archive.org/web/20190709152857/https://download.slock.it/public/DAO/WhitePaper.pdf.

[@JOL19] JOLOCOM. *A Decentralized, Open Source Solution for Digital Identity and Access Management (Whitepaper)*. Accessed: 2023-01-17. 2019. URL: https://jolocom.io/wp-content/uploads/2019/12/Jolocom-Whitepaper-v2.1-A-Decentralized-Open-Source-Solution-for-Digital-Identity-and-%20Access-Management.pdf.

[@KL18] Dmitry Khovratovich and Michael Lodder. *Anonymous credentials with type-3 revocation*. Accessed: 2023-01-17. 2018. URL: https://github.com/hyperledger/ursa-docs/blob/62bc87b/specs/anoncreds1/anoncreds.tex.

[@KV19a] Dmitry Khovratovich and Mikhail Vladimirov. *Tornado Circuit Audit*. Accessed: 2023-01-17. 2019. URL: http://web.archive.org/web/20220409180142/https:

//tornado.cash/audits/TornadoCash_circuit_audit_ABDK.pdf.

[@KV19b] Dmitry Khovratovich and Mikhail Vladimirov. *Tornado Privacy Solution Cryptographic Review*. Accessed: 2023-01-17. 2019. URL: http://web.archive.org/web/20220409180132/https://tornado.cash/audits/TornadoCash_cryptographic_review_ABDK.pdf.

[@Liq23] Liquid Democracy. *Adhocracy4*. Accessed: 2023-04-15. 2023. URL: https://github.com/liqd/adhocracy4.

[@LLW15] Eric Lombrozo, Johnson Lau, and Pieter Wuille. *Segregated Witness (Consensus layer)*. Accessed: 2023-05-29. 2015. URL: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

[@Mic19] Dmitry Khovratovich Michael Lodder. *Anonymous credentials 2.0*. Accessed: 2023-01-17. 2019. URL: https://wiki.hyperledger.org/download/attachments/6426712/Anoncreds2.1.pdf.

[@MiV18] MiVote. *Democracy Warrior Handbook*. Accessed: 2023-01-16. 2018. URL: https://web.archive.org/web/20190518215049/https://d3n8a8pro7vhmx.cloudfront.net/mivote/pages/349/attachments/original/1534738464/MIV_001_A5_DemocracyWarriorHandbook_FA2_digital.pdf?1534738464.

[@Mue18] Bernhard Mueller. *Smashing Ethereum Smart Contracts for Fun and Real Profit*. Accessed: 2023-01-17. 2018. URL: https://github.com/muellerberndt/smashing-smart-contracts/blob/master/smashing-smart-contracts-1of1.pdf.

[@Nol+19] Tobias Nolte, Andrew Witt, Olivia Heung, and James Yamada. *BBBlockchain*. Accessed: 2023-05-18. 2019. URL: https://web.archive.org/web/20221209101940/https://certainmeasures.com/BBBLOCKCHAIN.

[@Pan17] Sancho Panza. *Generalized version bits voting*. Accessed: 2023-05-24. 2017. URL: https://github.com/bitcoin/bips/blob/master/bip-0135.mediawiki.

[@PD16] Joseph Poon and Thaddeus Dryja. *The Bitcoin lightning network: Scalable off-chain instant payments*. Accessed: 2023-05-12. 2016. URL: https://lightning.network/lightning-network-paper.pdf.

[@RB20] Danny Ryan and Vitalik Buterin. *Serenity Phase 0*. Accessed: 2023-05-23. 2020. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2982.md.

[@Sov21] Sovrin Foundation. *Indy Walkthrough - A Developer Guide for Building Indy Clients Using Libindy*. Accessed: 2023-01-17. 2021. URL: https://github.com/hyperledger/indy-sdk/blob/master/docs/getting-started/indy-walkthrough.md.

[@The17] The Maker Team. *The Daj Stablecoin System*. Accessed: 2023-04-21. 2017. URL: https://web.archive.org/web/20221219131150/https://makerdao.com/whitepaper/Dai-Whitepaper-Dec17-en.pdf.

[@The20] The Social Coin. *Social Coin*. Accessed: 2023-04-25. 2020. URL: https://web.archive.org/web/20200903211207/https://thesocialcoin.com/?lang=en.

[@Van13] Nicolas Van Saberhagen. *CryptoNote v 2.0*. Accessed: 2023-05-12. 2013. URL: https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf.

[@VK19] Mikhail Vladimirov and Dmitry Khovratovich. *Tornado Cash Smart Contracts Audit*. Accessed: 2023-01-17. 2019. URL: http://web.archive.org/web/20220409180200/https://tornado.cash/audits/TornadoCash_contract_audit_ABDK.pdf.

[@Vla23] Aleksandr Vlasov. *web3swift*. Accessed: 2023-05-18. 2023. URL: https://github.com/web3swift-team/web3swift.

[@Vog19] Fabian Vogelsteller. *ERC-735: Claim Holder*. Accessed: 2023-01-17. 2019. URL: https://github.com/ethereum/EIPs/issues/735.

[@VB15] Fabian Vogelsteller and Vitalik Buterin. *EIP-20: Token Standard*. Accessed: 2023-01-16. 2015. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.

[@VY20] Fabian Vogelsteller and Tyler Yasaka. *ERC-725: Smart Contract Based Account*. Accessed: 2023-01-17. 2020. URL: https://github.com/ethereum/EIPs/issues/725.

[@WG19] Barry WhiteHat and Kobi Gurkan. *MicroMix*. Accessed: 2023-05-31. 2019. URL: https://hackmd.io/qlKORn5MSOes1WtsEznu_g.

[@Wil16] Jeffrey Wilcke. *To fork or not to fork*. Accessed: 2023-04-21. 2016. URL: https://blog.ethereum.org/2016/07/15/to-fork-or-not-to-fork.

[@Wor22] World Wide Web Consortium (W3C). *Verifiable Credentials Data Model v1.1 - Expressing verifiable information on the Web*. Accessed: 2023-01-17. 2022. URL: https://www.w3.org/TR/vc-data-model/.

[@Wui+21] Pieter Wuille, Peter Todd, Greg Maxwell, and Rusty Russell. *Version bits with timeout and delay*. Accessed: 2023-01-17. 2021. URL: https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki.

[@ZEX18] Zainan Victor Zhou, Evan, and Yin Xu. *Voting Interface*. Accessed: 2023-05-15. 2018. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1202.md.