Secure proof verification blockchain patterns

Tiphaine Henry, Sara Tucci Piergiovanni

Secure Proof Verification Blockchain Patterns

Tiphaine Henry[1 [0000−0002−7981−8934]]
and Sara Tucci-Piergiovanni[1 [0000−0001−9738−9021]]

Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France

Abstract. In blockchain networks, transaction data is accessible to all participants by design and hence verifiable. This verifiability feature of data stored on the ledger by any participant fosters trust within data, especially in uncertain environments. However, the public nature of onchain data imposes limitations across various scenarios as subsets of data should be kept private. Zero-knowledge proofs (ZKPs) have emerged as a solution within the literature to overcome this issue. The raw data is not published onchain, only a proof of knowledge of this data is. Hence, the blockchain is used as a trustworthy means for proof verification without requiring data disclosure. Despite their effective use in many scenarios, the formalization of zero-knowledge proof techniques within blockchain settings remains under-explored in current literature, and makes their integration difficult for non-expert blockchain practitioners due to the plurality and complexity of zero knowledge proofs. Software engineering patterns are used in the literature to formalize recurring software engineering practices stemming from the literature and experience of practitioners. Several patterns have been proposed to formalize blockchain-based architecture practices. However, no blockchain patterns tailored to confidential proofs using ZKPs have been designed in the literature. Hence, this paper aims to address this gap by formalizing key blockchain patterns relying on ZKP to handle secure proof verification identified in the literature. We formalize a general pattern called Secure Proof Verification pattern and three related sub-patterns, two of them focusing on efficient or trustless proof verification, and one specifically designed for interval membership verification to aid practitioners in selecting the most suitable non-interactive ZKP design for a blockchain-based application.

Keywords: Blockchain · Software Patterns · Zero-knowledge proofs

1 Introduction

Zero knowledge proofs (ZKPs) have emerged as a key technique in the recent years to share a proof of a knowledge without divulging the inner content of this data [26, 8]. This has many practical applications regarding confidentiality-aware scenarios. For instance, in identity verification scenarios, individuals can share a proof of their identity without disclosing additional details such as birth date or location. Similarly, in circular economy contexts, where a manufacturer must demonstrate the non-toxicity of a product to a recycler, the manufacturer can share a proof that its products do not contain a given toxic substance without revealing detailed information about the product recipe, thus safeguarding its industrial know-how [30].

However, in certain scenarios, a lack of trust between the prover and verifiers necessitates a system architecture that ensures the integrity and trustworthiness of zero-knowledge proof generation and verification, guaranteeing that the proof remains untampered [23]. Furthermore, oftentimes, direct interaction between the verifier and the proof issuer is not feasible. For example, a manufacturing company that issued a proof of non-toxicity of a product may have ceased operations,

or alternatively, the proof issuer—the manufacturing company—may opt against providing direct access to its information system to unknown verifiers due to cybersecurity concerns. However, subcontracting the custody of the proof to a third party may instill distrust on the verifier's side. Hence, a need arises for an independent, asynchronous, and reliable verification mechanism for ZKPs.

Blockchain has been leveraged as a support for trustworthy proof verification of non-interactive ZKPs in the literature. The non-interactive variant of zero-knowledge proof enables for the asynchronous verification of proofs which fits the blockchain design where participants can interact in an asynchronous fashion. The proof, generated offchain, is stored onchain and can be accessed by any participant of the network. The integrity property of transactions stored onchain ensures that the proof remains tamper proof after its generation. Moreover, as it is by design accessible to any participant for verifiability purposes, this ensures a reliable and asynchronous access to the proof. Moreover, a verification system can be built on top of the blockchain network using smart contracts. Hence, reliability in the verification system can be ensured in a decentralized fashion in uncertain scenarios. Moreover, the integration of non-interactive ZKPs in blockchain based applications addresses key limitation concern of blockchains regarding confidential data handling as any data stored in the ledger is accessible and verifiable by all members of the network [5, 24]. Numerous studies integrate non-interactive ZKPs into blockchain architectures to blind transaction content while ensuring verifiability and computational efficiency, primarily utilizing succinct non-interactive arguments of knowledge (SNARKs) [32, 30]. However, despite their potential, the scattered nature of information across literature sources and the lack of formalized patterns make selecting suitable non-interactive ZKPs mechanisms challenging for non-expert audiences [8].

Software design patterns are proposed in the software engineering literature to aid practitioners, such as software architects, developers, system administrators, or technical leads, in addressing recurring architectural problems [1, 2, 17]. These patterns formally describe architectural solutions using standardized formats like GOF or the Alexandrian form format. In recent years, software design patterns have found application in the blockchain field [2, 36, 10], facilitating streamlined integration of blockchain using best practices. Several papers aim to formalize patterns at various levels, including data management or security patterns [46, 7, 41]. However, while some papers identify blockchain patterns suitable for confidential data handling [46, 20], these collections often lack comprehensiveness. For instance, the encryption pattern introduced by Xu et al. [46, 41] remains high-level and does not mention zero-knowledge proof techniques [49, 39]. Moreover, these papers often overlook identifying network assumptions and trust models derived from these assumptions, which are crucial in the decision-making process for blockchain application architects. Consequently, there is a research gap in defining blockchain patterns for confidential data handling based on ZKPs. To address this issue, we propose the formalization of the NI-ZKP blockchain integration pattern, that we refer to as the Secure Proof Verification blockchain pattern. We also formalize three corresponding sub-patterns, two of them focusing on issuing data-related statements, with a focus on proof generation and verification efficiency for the first pattern, and a trustless setup for the second pattern, and the Interval Membership Verification pattern that aims at sharing a proof of a value belonging to a range.

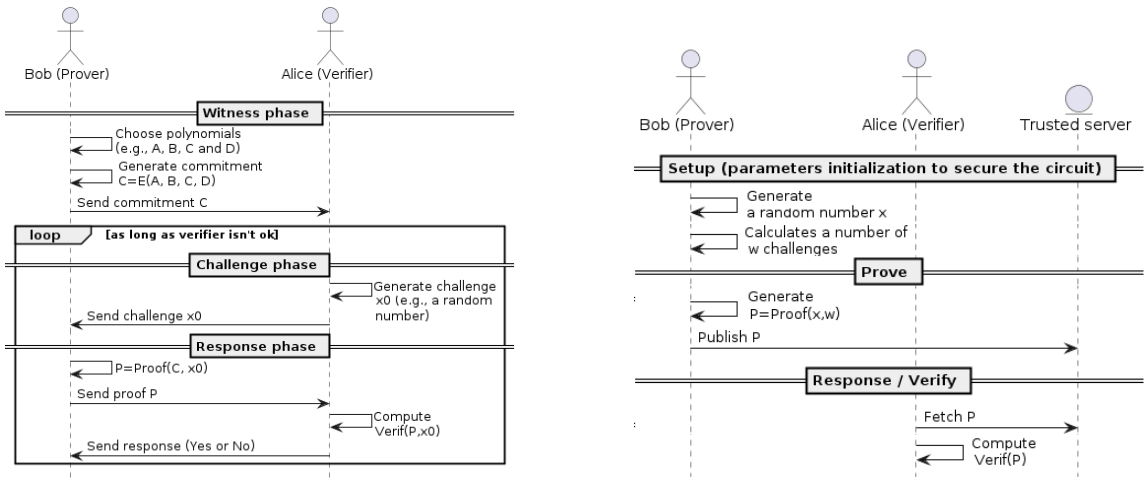
The remainder of this paper is structured as follows. Section 2 introduces key concepts on blockchains and ZKPs. Section 3 discusses related work on blockchain patterns and confidential data handling patterns published in the literature. Section 4 presents the methodology used to construct this survey. In Section 5, we present the key blockchain patterns for confidential data handling based on NI-ZKP formalized with the Alexandrian form format. Section 6 concludes the paper with a summary and discussion of the results.

2 Preliminaries

2.1 Blockchains and Blockchain Patterns

A DLT is a shared immutable ledger based on a peer-to-peer network, i.e., a ledger spread across multiple computing nodes [44]. DLTs have originally emerged to keep track of digital transfers in financial applications, but they can be used to store any kind of data. Data recorded in a DLT is replicated at each peer of the network. In order to add new data to the ledger a user will issue a so-called transaction containting the data. The transaction is then collaboratively validated by blockchain network peers, through a consensus mechanism, before being added to each copy of the ledger. A smart contract is a program hosted on a blockchain network activated by a transaction [43]. When a smart contract executes, its updated state is registered by a new transaction also stored in the blockchain ledger. This makes the transaction immutable and tamper-proof [4]. A smart contract comprises both variables and functions. Smart contract functions can execute arbitrary code, access the state of the variables and optionally update them. The blockchain ledger history is the collection of all blocks generated by the blockchain. Each block tracks all the included transactions and resulting states after executing those transactions and smart contracts [7]. More precisely, each block holds (1) the reference to the previous block, (2) a tamper evident digest of the transaction history to attest the integrity and the ordering of the blocks, and (3) the list of the transactions to record [40].

Software engineering patterns provide well-tested solutions for frequently encountered application development use cases [2]. Standardized formatting such as the Alexandrian Form Format systematically include for each pattern the description, the forces or tradeoffs at stake, the benefits and drawbacks, and their main applications. As mentioned in the introduction, a family of patterns focusing on blockchain patterns has recently emerged in the literature [46, 41], formalizing each use such as data management, architectural, or access control strategies.



(a) I-ZKP (b) NI-ZKP

Fig. 1: Illustration of the exchanges between prover and verifier during I-ZKP and NI-ZKP

2.2 NI-ZKPs

ZKPs, formally defined by Goldreich et al. in 1991 [26], serve as cryptographic tools enabling one party to convince another party of the truth of a statement without revealing any additional information beyond the statement's validity. An illustrative analogy often employed to explain ZKPs is the "Alibaba cave" scenario, proposed by Jean-Jacques Quisquater and Louis Guillou in 1989 [38]. In this scenario, consider two individuals, Alice (the prover) and Bob (the verifier), exploring a mysterious cave. Alice possesses knowledge of a secret passage within the cave and aims to convince Bob of her knowledge without disclosing the passage's exact location. Through a series of back-and-forth interactions within the cave, Alice demonstrates her ability to navigate within the cave, convincing Bob of her knowledge without revealing the specific route. The Alibaba cave analogy illustrates the essence of ZKPs: proving knowledge of a fact or secret without revealing the information itself, and without ambiguity.

ZKPs involve providing a *statement* with a validity *proof* to a verifier. Statements can be defined as a formal declaration based on a secret that is subject to verification. An example could be the statement "I know a number x such that $x^2 \equiv 9 \pmod{13}$," which can be verified using modular arithmetic. The proof provides cryptographic evidence of the truth of a statement without revealing underlying information. The proof is computed with the secret data and a commitment to this secret data. For instance, Alice would compute the proof of the statement by providing the proof generation algorithm with the secret number x and the commitment to x by computing a commitment $C = H(x)$, where H is a cryptographic hash function. As the statement is forwarded with its proof to a recipient, the recipient can use the proof to ensure the validity of the statement without accessing the secret data used to build the statement. ZKPs exhibit several key properties essential for ensuring the integrity and confidentiality of verified statements. These properties include completeness, which guarantees that an honest claim can convince an honest verifier; soundness, which minimizes the potential for deception by an untrustworthy prover; and zero knowledge, wherein the verifier gains confirmation of the claim's validity without acquiring any additional information beyond what is already known [26]. Some ZKP protocols rely on a trusted setup, which consists of multiple participants generating initial parameters for a cryptographic system, ensuring their integrity. An assumption is made about the honesty of the participants in following each step of the setup.

The choice of the ZKP protocol impacts the proof size, the proof time generation, and the proof time verification. Moreover, two families of ZKPs exist based on the proof verification design (interactive or non-interactive). Figure 1a illustrates the interactive zero-knowledge proof scenario. Two honest but curious parties, the prover and verifier, engage in a series of back-and-forth interactions where (1) the prover sends a commitment using private input data to a verifier, (2) the verifier responds with a challenge, and (3) the prover computes the proof using the commitment and the challenge. These steps loop until the verifier is convinced of the commitment. A non-interactive ZKP (NI-ZKP) scheme extends the concept of ZKPs to scenarios where interaction between the prover and verifier is minimized or eliminated entirely [14]. Figure 1b illustrates the NI-ZKP scheme. The prover generates a proof offline, independently of the verifier. The verifier can subsequently verify the proof without further interaction with the prover [28].

The suitability of interactive schemes within blockchain environments has been questioned due to concerns such as communication complexity, network performance degradation, and the necessity for both parties to be online simultaneously [28]. Consequently, NI-ZKPs best apply for blockchain-based scenarios.

2.3 Motivating example

We now introduce a motivating example anchored in the circular economy to identify key properties needed to handle a proof statement verification scenario that can be addressed by blockchain.

In a circular economy supply chain, participants sometimes need to prove and verify some claims but without the possibility to share or access the content of the statement. For example, a battery manufacturer could need to prove various claims related to the performance, authenticity, and sustainability of lithium-ion batteries without disclosing sensitive information about their formulation, manufacturing process, or environmental impact. These scenarios necessitate a method to verify claims without compromising the confidentiality of the underlying information.

The following key properties are needed in this scenario: *(P1) traceability of exchanges:* tracing proof issuance (e.g., for the verifier to be sure of the identity of the issuer and the integrity of the proof) and tracing proof verification (e.g., in cases where the manufacturer wishes to manage who can verify its claims); *(P2) asynchrony of the verification:* avoid dependency on the proof issuer - separate the verification from the issuing for long term scenarios - e.g., recycling of a product where the manufacturer does no longer exist: how to still access proof that the product is not toxic?; *(P3) trust in the verification step:* being sure that the claim is rightly verified, with correct information.

Blockchain can be a fit to fill these properties. First, transactions recorded onchain offer properties such as transparency and verifiability. Moreover, smart contracts can enforce predefined rules for securing data sharing protocols such as enforcing a predefined access control [16]. Hence, proof issuance and each verification request can be tracked onchain, which addresses the requirements for the traceability of the exchanges *(P1)*. The proof being stored onchain, the verifier does not need to interact directly with the issuer, which answers the requirement for the asynchrony of the verification *(P2)*. Finally, blockchain history is immutable and accessible to all participants of the network, which prevents malicious actors from modifying data or code. As data cannot be tampered, and credentials can be generated regarding issuers for example, this makes the verification step trustworthy *(P3)*. Hence, an interest arise in formalizing integration strategies for NI-ZKP in blockchain-based scenarios.

3 Related work

The aim of defining blockchain patterns is to summarize practical solutions today identified but scattered in the literature to aid practitioners in selecting appropriate mechanisms while building decentralized applications (dApps). A first set of papers describe formally blockchain patterns. A recent systematic literature review conducted by Six et al. [41] presents a taxonomy of these existing blockchain patterns, with some addressing concerns related to confidential data handling. This work includes the preliminary work by Xu et al. [46] which presents an initial set of data management patterns from which three patterns can be used for confidential data handling, namely "On-chain encryption," "Off-chain data storage," and "State Channels". Six work extends the "Onchain Encryption" pattern with two sub-patterns: "Commit and Reveal" and "On-chain encryption". However, no mention is made of ZKPs in the identified patterns. Moreover, identified patterns fail to provide trust models or assumptions on network requirements, critical elements in infrastructure design. Another subset of papers describe informally blockchain technics and decision models for confidential data handling. One paper proposes a decision model for handling sensitive health data in blockchain systems, highlighting encryption strategies but without systematizing encryption techniques through patterns, and missing ZKP-based technics [22]. Three works summarize

confidential data handling strategies and mention ZKPs, however without in depth description nor following the Alexandrian form format [49, 39, 37]. Zeiselmair et al. and Qi et al. [49, 37, 39] summarize verifiable computing techniques with blockchains, however without a proper formalization and with missing details on the integration of ZKPs in blockchain environments. Hence, there is a research gap regarding the formalized description of NI-ZKP blockchain patterns.

4 Methodology

Blockchain data management patterns found in the scientific literature overlook confidential data management with proofs. As such techniques are discussed in both scientific and grey literature sources (e.g., white papers, online news sites, blogs, and videos), we follow a Multivocal Literature Review (MLR) to uncover blockchain patterns for transactional confidentiality. An MLR is an extension of a systematic review utilizing both scientific and grey literature [25]. The MLR process begins with formulating the research question outlined in Section 1. From this question, the following search query is constructed: *(((crypto OR zero-knowledge) proof OR ZKP OR SNARK OR STARK OR proof) OR confidentiality OR confidential OR privacy) AND (blockchain OR (blockchain AND pattern) OR dApp OR "smart contract")*. Scientific literature sources are searched through ScienceDirect, ACM Digital Library, IEEE, SpringerLink, and AISEL, while grey literature sources are searched using the Google Search Engine. Snowball sampling is employed to expand the pool of potential sources by tracing citations and links from already identified sources, iterating until theoretical saturation is achieved. Subsequently, the pool of sources is filtered based on predefined inclusion and exclusion criteria, such as whether the technique is rooted in a blockchain-based information system, leverages ZKPs, and is well-defined. Techniques related to identity privacy and access control are excluded. We also exclude protocols focusing solely on blockchain architectures to enhance scalability or interoperability such as rollups or bridges as our goal is set on technics to handle proofs of confidential onchain. Preference is given to information sources from blockchain platforms and DApp governance bodies in the grey literature over third-party sources. The resulting pool of sources comprises 32 papers from formal literature (11 describing blockchain patterns and corresponding decision models, 12 focusing on literature surveys, and 11 describing techniques for confidential data handling on-chain without formal patternization) and 21 web pages from grey literature. Filtering criteria include factors such as articles focusing solely on transactional data and providing detailed explanations. Subsequently, the vetted pool of sources undergoes analysis using open and axial coding, iteratively deriving a general pattern and a set of sub-patterns. We follow the Alexandrian format [1] as this format aligns with the prevalent structure found in blockchain software pattern papers [46, 41]. For each sub-pattern, additional information is provided, including the trust model (honest, honest but curious, or malicious actors), communication hypotheses (synchronous or asynchronous), and confidentiality requirements for the transactional data involved.

5 Secure Proof Verification Blockchain Patterns

5.1 General Secure Proof Verification Blockchain Pattern

Summary: The pattern enables secure and efficient proof validation of sensitive data points referred to as secrets without interaction nor accessing the secret directly. NI-ZKP statements whose proof is generated offchain are verified by a smart contract and stored in the ledger, enabling future verifiers to accept the claim without recomputation (Figure 2).

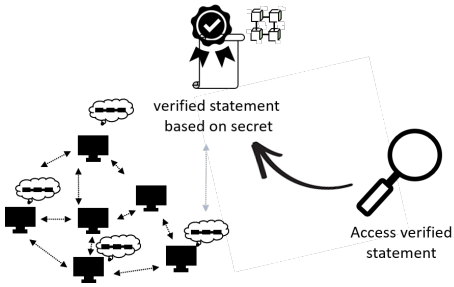


Fig. 2: Illustration of the General Secure Proof Verification blockchain pattern.

Context: A participant needs to share a statement based on a secret onchain without disclosing this secret to blockchain participants.

Problem : The challenge is to share a statement building on a secret with a validity proof onchain without disclosing the secret the statement is based on to each participant of the network nor engaging in a direct interaction with them.

Forces : The problem requires balancing the following forces: (i) Statement integrity, availability and verifiability: These three properties can be applied simultaneously to any statement stored onchain which enables the prover to share a statement to any verifier in the long term without a direct interaction; (ii) Confidentiality preservation: Since all data stored on a blockchain is inherently public by design, preserving confidentiality requires careful consideration as it is not possible to store sensitive data onchain directly.

Solution Depending on the NI-ZKP protocol chosen, the pattern can start with a trusted setup, i.e., an initial step that establishes the security foundation for the ZKP system, relying on trust in the participants to perform the setup honestly and correctly [28]. The purpose of this setup is to generate a set of public parameters whose nature depends on the specific ZKP protocol. These public parameters may include random elements, mathematical structures needed for proof computation, commitment schemes, or verification keys. After setup, these parameters are made publicly available within the blockchain network for later use during NI-ZKP proof generation or verification. Afterward, a prover generates a proof off-chain using public parameters for the NI-ZKP scheme stored on-chain and its private information. The proof is submitted to the blockchain network. Finally, the verification stage can take place. Network participants (e.g., network nodes, smart contracts, or other participants with access to the blockchain ledger) publicly verify the proof on-chain using cryptographic parameters and the proof itself. The result, typically a Boolean indicating acceptance or rejection, is recorded in the blockchain ledger. If the proof is valid, indicating that the assertions made by the prover are correct, the verification process is successful. Future verifiers will then have the possibility to accept the claim without needing to recompute the verification algorithm. Finally, the pattern entails several requirements. Asynchronous communication between the prover and verifier is necessary, contrasting with interactive ZKPs that rely on synchronicity due to exchanges during proof verification. The statement and the corresponding proof should be public. In case of a trusted setup, the security parameters must remain private and discarded at the end of the setup, while public parameters must be public. The functions for proof generation and verification should

be publicly accessible, allowing any party to generate and verify proofs as needed. At last, the trust model hypothesises that the prover and verifiers do not inherently trust each other [23].

Consequences :

- *Benefits:*
  - Data confidentiality. Data is not stored directly onchain but is embodied into a statement stored onchain.
  - Proof verification asynchrony. The prover does not need to be connected synchronously with the verifier as the proof is stored in the blockchain.
  - Parallel verifications. The proof can be used by several verifiers simultaneously accessing the ledger where the proof is stored.
- *Drawbacks:*
  - Efficiency. Proof generation duration varies depending on the chosen schemes (from ms to hour), which may impact transaction processing efficiency if not chosen appropriately [28, 34].
  - Setup. Some schemes may require a trusted setup, introducing dependencies on trusted parties and potentially compromising decentralization [20].
  - Tooling. Though toolboxes exist to help implement ZKPs [21, 33], the tools have heterogeneous maturities and varying developer community sizes [3].
  - Security. Certain ZKP protocols rely on a random oracle model for security, which may introduce vulnerabilities and compromise overall system security [14].
  - Applicability. Not all NI-ZKP sub-patterns handle proofs based on two or more secrets [11].

Related Patterns This pattern can serve as a foundation for the offchain oracle pattern introduced by Xu et al. [46] to prove the integrity of an API call such as a delegated computation.

Known Uses: This pattern can be used for proving balance thresholds [50], validating computations [11, 23], and sharing product properties across supply chains [18].

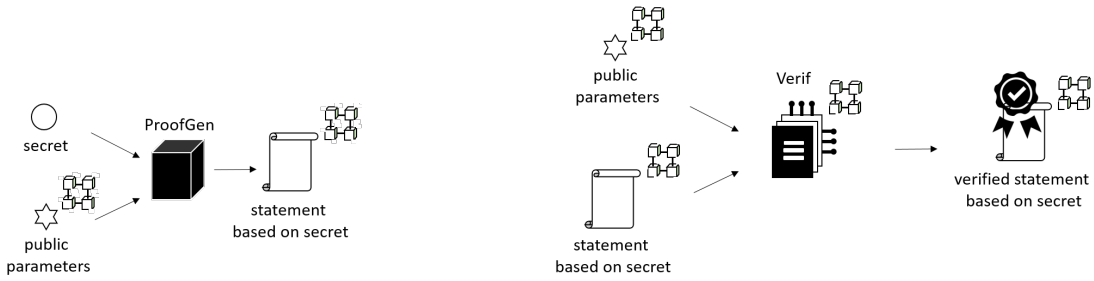
We describe below the main Secure Proof Verification sub-patterns identified in our literature review, namely (1) Efficient Secure Proof Verification, (2) Trustless Secure Batch Proof Verification, and (3) Interval Membership Verification blockchain sub-patterns.

5.2 Efficient Secure Proof Verification Blockchain Sub-pattern

Summary: A pattern relying on a specific type of NI-ZKP (SNARKs) that offers succinct proofs[1] and an efficient verification. This sub-pattern requires an off-chain trusted setup during which the public parameters required for the proof generation and proof verification are generated. Figure 3b illustrates the pattern.

Context: A participant needs to share a statement based on a secret onchain without disclosing this secret to blockchain participants in an efficient manner. The generated statement should be light and its proof should be easily computed by any participant.

[1] A succinct proof is a cryptographic construct that enables the verification of complex computations or statements using a compact proof size, typically much smaller than the original computation.



(a) SNARK proof generation using public parameters stored onchain

(b) SNARK proof verification carried by a smart contract

Fig. 3: SNARK blockchain pattern intuition

Problem : The challenge is to share a statement building on a secret with a validity proof onchain without disclosing the secret the statement is based on to each participant of the network nor engaging in a direct interaction with them. Usability issues may arise as the validity proof generation and verification may require heavy computing power.

Forces: The problem inherits from the forces identified for the general pattern. It additionally requires balancing the following force: (i) Proof verification efficiency: the proof verification should not cause scalability breaks due to the proof size or the verification time.

Solution: The solution involves using a Zcash-style (non-interactive) SNARKS protocol, known as Succinct Non-Interactive Argument of Knowledge which builds on the *Hiding queries to polynomials* strategy and Quadratic R1CS[32] [2]. Key actors of this pattern include a trusted authority or a consortium of blockchain participants conducting the trusted setup offchain, the prover, and a smart contract responsible for storing the SNARK public parameters and executing verification. The solution includes the following steps. First, a trusted setup takes place. The public parameters required for SNARKs are generated through a trusted setup process, such as the Power of Tau ceremony, involving the selection of x_0 and computation of homomorphic encryption. x_0 is discarded after setup to maintain security. The setup is conducted off-chain by the SNARK Authority. Then, public parameters generated during setup are published on-chain. Afterward, the prover generates a succinct proof off-chain using private inputs and public parameters. The proof is submitted to the blockchain. Finally, the verifier performs verification of the proof using public parameters stored onchain.

Consequences:

- *Benefits:*
  - Proof Sizing: zk-SNARK proofs are small, requiring less gas for validation compared to other schemes, estimated at only 24% of the gas required for STARKs [3].

[2] Another ZKP building on Hiding queries to polynomials are *Multi-use circuits* (Reducing computational problems into arithmetic circuits). However, no scheme has been identified for a blockchain use

- Efficient Verification: Verification is polynomial time, scaling polylogarithmically in n, with no quasi-linear proving [30].
- Non-Interactive: Becomes non-interactive after the trusted setup.
- Code Maturity: large community and available tools (e.g., Zokrates library and lib-SNARK). Moreover, SNARK integration in Ethereum is considered in EIPs 196 and 197 [3, 9, 20].

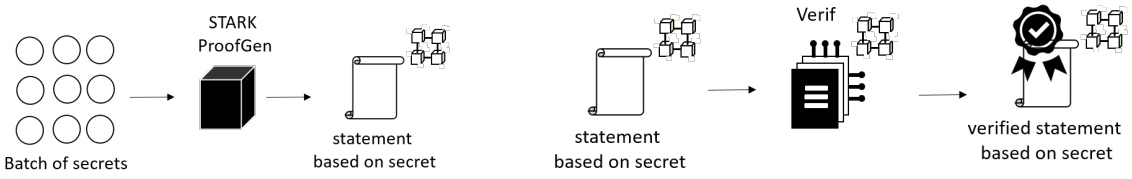
– *Drawbacks:*
- Lack of Transparency: Users must trust the correct execution of the setup and the destruction of associated secrets [24, 27].
- High Computational Costs: Generating zk-SNARK proofs is computationally intensive, taking 1 to 2 minutes per transaction [24, 27].
- Non-Succinct Trusted Setup: The setup process is often complex and inefficient, relying on initial participants without guaranteeing security.
- Quantum Assumptions: Vulnerabilities exist due to possible backdoors in elliptic curve random number generators, which are not post-quantum secure [3].
- Side-Channel Attack Vulnerabilities: Mathematical operations and optimizations may inadvertently leak information [3].

Related Patterns: See other Secure Proof Verification patterns.

Known Uses: Zcash employs zero-knowledge proofs (ZKPs) to ensure transaction privacy, where a transaction sender produces a ZKP of its ability to spend more than or equal to the transaction value, thereby protecting transaction details and ensuring anonymity. Additionally, Zokrates is utilized for processing Know Your Customer (KYC) procedures on-chain [28]. Moreover, ZKPs are employed for proving the correct execution of delegated off-chain computations [20, 35], with proofs designed to ensure verification cost independence from the complexity of the off-chained computation.

5.3 Trustless Secure Batch Proof Verification Blockchain Sub-pattern

Summary: This pattern leverages an off-chain proof generator using STARKs and an on-chain verification smart contract to enable the verification of batches of statements within blockchains. By using hash functions, it avoids the need for a trusted setup, enhancing transparency and auditability. The verified proofs are stored on the ledger, making the certified statements accessible to all blockchain participants. Figure 4 depicts the pattern.



(a) STARK proof generation (b) STARK proof verified by a smart contract

Fig. 4: Trustless Secure Batch Proof Verification Blockchain Sub-pattern intuition

Context: A participant needs to share several secrets onchain without disclosing the secrets to blockchain participants. Additionally, participants of the blockchain network may be malicious and have not identified a trusted authority.

Problem: The challenge is to share a statement building on a batch secret with a validity proof onchain without disclosing the secret the statement is based on to each participant of the network nor engaging in a direct interaction with them. Moreover, participants do not or cannot rely on a trusted setup to generate the public parameters, which makes impossible the use of a trusted setup to generate the initial parameters usually required to generate NI-ZKP proofs.

Forces: The problem inherits from the forces identified for the general pattern. It additionally requires balancing the following force: (i) No trusted setup: no trusted intermediaries or centralized authorities should be responsible for the verification.

Solution: This pattern leverages two bricks, an off-chain proof generator using a STARK which relies on hash functions [3] and an on-chain verification smart contract. The pattern consists in three steps. First, a smart contract integrating the STARK proof verification logic is deployed onchain. It is designed to receive the proof and verify its correctness using the selected STARK library. Then, the prover generates an off-chain proof of a public statement using the secret data and corresponding private commitment. The proof possibly includes a batch of statements. Finally, the smart contract proceeds to the on-chain verification of the proof to validate the correctness of the statement. The outcome of the proof verification is stored on the ledger and accessible to all blockchain participants.

Consequences:

- *Benefits:*
  - Proof generation and verification computational efficiency (linear proving time).
  - Transparency: no trusted setup, minimum trust assumptions.
  - Post-quantum security: hash functions allow for quantum resistance.
- *Drawbacks:*
  - Proof size: Larger than SNARKs.
  - Verification time: Larger than SNARKs.
  - Code maturity: Lack of developer documentation despite nascent STARK proof generation libraries[4].

Related Patterns: See other Secure Proof Verification patterns.

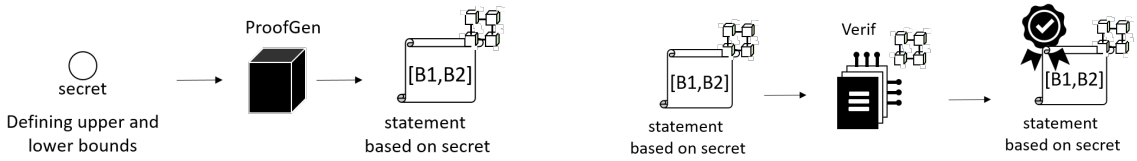
Known Uses: The Cairo toolchain is used for off-chain proof generation of general computation [33, 42]. These proofs and computation results are sent on-chain to a Verifier smart contract and used in applications such as DeversiFi and dYdX on Ethereum, or Immutable on Polygon.

[3] Two advantages from using hash functions: (1) quantum resistance, (2) no trusted set-up (more transparent and auditable than zk-SNARKs) [8, 31, 3]

[4] List of identified STARK libraries at the time of writing: libSTARK, STARKware STARKDEX alpha, STARKExchange, distaff, Cairo

5.4 Interval Membership Verification Blockchain Sub-pattern

Summary: Publish onchain a statement on a secret integer belonging to a public interval upper and lower bounds without revealing the secret integer to a set of possibly unknown verifiers. Figure 5 depicts the pattern variant with bulletproofs. The proof, generated offchain, consists in defining publicly the upper and lower bounds of the secret. It is verified by a smart contract and the generated certified statement is then accessible to all the participants of the network. In other variants of this pattern, a trusted setup used to generate public parameters is additionally necessary.



(a) Range proof generation - Bulletproof illustration

(b) Range proof verification - Bulletproof illustration

Fig. 5: Interval Membership Verification Blockchain Pattern intuition

Context: A participant needs to share a statement ensuring that a value falls within a specified range based on a secret onchain without disclosing this secret to blockchain participants.

Problem : The challenge is to share a statement building on a secret with a validity proof onchain without disclosing the secret the statement is based on to each participant of the network nor engaging in a direct interaction with them. Meanwhile, traditional verification methods often require revealing the exact number onchain, which may not be desirable due to privacy or security concerns.

Forces: The problem inherits from the forces identified for the general pattern. It additionally requires balancing the following forces: (i) trustlessness of the verification mechanisms: no trusted intermediaries or centralized authorities should be responsible for the verification; (ii) proof generation and computation efficiency: the proof should be easily generated and easily verifiable in terms of latency and proof size.

Solution: The pattern aims at proving the validity of a statement claiming a secret data belongs to a specified range. The pattern involves two main approaches, leveraging or not a a trusted setup, depending on the chosen range proof protocol and the range size (larger or small) [30, 19]. In general, small secrets require less computational effort and resources to generate and verify range proofs. With a trusted setup, the pattern comprises three key steps: deploying a smart contract onchain to integrate range proof verification logic, generating range proofs off-chain using optimized schemes tailored to different secret sizes [30, 19], and verifying these proofs on-chain using the deployed smart contract. The square decomposition protocol is optimized for large secrets while multi-base decomposition is optimized for small secrets. It breaks down secrets into their binary representations, facilitating boolean arithmetic to establish interval membership. We refer to the Efficient Secure Proof Verification Blockchain Sub-pattern for details on the steps. In contrast,

without a trusted setup, the pattern adopts a trustless secure batch proof verification approach: first, off-chain proof generation using techniques like Bulletproofs, which do not require a trusted setup [12]; second, on-chain verification of the proofs, ensuring the validity of the data range. This approach enhances transparency and scalability while maintaining security, making it particularly suitable for blockchain environments with minimized trust assumptions. We refer to the Trustless Secure Batch Proof Verification Blockchain Sub-pattern for details on the steps.

Consequences:

- *Benefits:*
  - Proving value within a range without compromise on data confidentiality.
  - Several methods that adapt to several use cases, such as Bulletproofs for succient proofs and multi-base decomposition for small secrets.
  - Shorter proofs and reduced communication complexity compared to SNARKS or STARKS (argument length [0.2-1kB] [19, 30]).
- *Drawbacks:*
  - Computing and storage resources.
  - Quantum attack susceptibility.
  - Reliance on newer, less studied assumptions.
  - Trusted setup depending on the protocol.

Related Patterns: A common use case is to use this pattern for homomorphic commitments: when verifying the legitimacy of a transaction, the range proof protocol is used to verify that a transaction amount is non-negative [19].

Known Uses: Monero uses Bulletproofs to prove that a committed value is within a given range [19]. Mimblewimble employs Pedersen commitments to hide transaction amounts [30]. Provisions prove users' solvency using range proofs combined with Multi-Party Computation (MPC) in Bitcoin [30]. Credential validation is achieved with Hashwires [15]. Additionally, ZKPs are used for proving set membership [13, 6, 48, 47].

6 Discussion and conclusion

In this paper, we propose a new blockchain pattern called Secure Proof Verification Blockchain Pattern that formalizes the integration of NI-ZKPs within blockchain architectures, addressing the need for secure and reliable proof sharing and verification. We moreover identify and formalize three key related sub-patterns relying on the SNARKs, STARKs, and range proof ZKP protocols, each serving distinct use cases, respectively providing a proof of knowledge of a value, a batch of values, or a value range verification. We find that the use of the Secure Proof Verification pattern is advocated in situations where the proof must be provided to more than one verifier, and in an untrusted environment (the prover and the verifiers distrust each others, or are not sure they will be able to interact in a peer to peer fashion in the long run).

Secure Proof Verification patterns are to be enriched in the future with complementary sub-patterns for sharing proofs of confidential data onchain. Indeed, zero-knowledge proof protocols are still evolving [50, 34], e.g., towards light-weight cryptographic algorithms with stronger privacy under weaker assumptions, or lattice-based NI-ZKP for post quantum era. Moreover, the collection

of subpatterns presented in this paper does not include recent integration architectures such as ZKP-based oracles, zkVMs, or private smart contracts. Future work should also focus on expanding the categorization of blockchain patterns to include other cryptographic patterns used to manage data onchain (e.g., to share confidential data, or to compute confidential data).

Finally, to complete the set of blockchain patterns leveraging ZKPs apart from sharing proofs of confidential data, formalizing patterns focusing on the use of ZKPs to improve blockchain scalability or support crosschain interoperability (e.g., zk-rollups, or zk-bridges) could be formalized to provide a complete overview of the use of ZKPs in blockchains [29, 45].

References

1. Alexander, C.: A pattern language: towns, buildings, construction. Oxford university press (1977)
2. Alexander, C.: The timeless way of building, vol. 1. New york: Oxford university press (1979)
3. Asher, M.: Zero-Knowledge Proofs: STARKs vs SNARKs. `https://consensys.io/blog/zero-knowledge-proofs-starks-vs-snarks` (2021), accessed: 10/10/2023
4. Ayub, M., Saleem, T., Janjua, M., Ahmad, T.: Storage state analysis and extraction of ethereum blockchain smart contracts. ACM TOSEM **32**(3), 1–32 (2023)
5. Azgad-Tromer, S., Garcia, J., Tromer, E.: The case for on chain privacy and compliance. Stanford Journal of Blockchain Law & Policy **6**(2) (2023)
6. Bai, T., Hu, Y., He, J., Fan, H., An, Z.: Health-zkidm: a healthcare identity system based on fabric blockchain and zero-knowledge proof. Sensors **22**(20), 7716 (2022)
7. Bandara, H.D., Xu, X., Weber, I.: Patterns for blockchain data migration. In: EuroPlop. pp. 1–19 (2020)
8. Ben-Sasson, E.: A cambrian explosion of crypto proofs. NAKAMOTO. Jan **8** (2020)
9. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: IEEE SP. pp. 459–474. IEEE (2014)
10. Benedetti, A., Henry, T., Tucci-Piergiovanni, S.: Gas cost analysis of proxy and diamond patterns: Towards trusted smart contract engineering in evm blockchains, in press. In: FC - WTSC (2024)
11. Benhamouda, F., Halevi, S., Halevi, T.: Supporting private data on hyperledger fabric with secure multiparty computation. IBM Journal of Research and Development **63**(2/3), 3–1 (2019)
12. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: IEEE SP. pp. 315–334. IEEE (2018)
13. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: ASIACRYPT. pp. 234–252. Springer (2008)
14. Chaabouni, R., Lipmaa, H., Zhang, B.: A non-interactive range proof with constant communication. In: FC 2012. pp. 179–199. Springer (2012)
15. Chalkias, K., Cohen, S., Lewi, K., Moezinia, F., Romailler, Y.: Hashwires: Hyperefficient credential-based range proofs. Cryptology ePrint Archive, Paper 2021/297 (2021)
16. Chen, C.L., Deng, Y.Y., Weng, W., Sun, H., Zhou, M.: A blockchain-based secure inter-hospital emr sharing system. Applied Sciences **10**(14), 4958 (2020)
17. Chia, S.Y., Xu, X., Paik, H.Y., Zhu, L.: Analysis of privacy patterns from an architectural perspective. In: ICSA-C. pp. 60–67. IEEE (2022)
18. Circularise: *Take control of your supply chain with digital product passports* (2023), `https://www.circularise.com/dpp` [Accessed: 10/03:2023]
19. Deng, C., Fan, J., Wang, Z., Luo, Y., Zheng, Y., Li, Y., Ding, J.: A survey on range proof and its applications on blockchain. In: CyberC. pp. 1–8. IEEE (2019)
20. Eberhardt, J., Tai, S.: On or off the blockchain? insights on off-chaining computation and data. In: ESOCC 2017. pp. 3–15. Springer (2017)
21. Eberhardt, J., Tai, S.: Zokrates-scalable privacy-preserving off-chain computations. In: iThings. IEEE (2018)

22. Erler, C., Schinle, M., Dietrich, M., Stork, W.: Decision model to design a blockchain-based system for storing sensitive health data. In: ECIS (2022)
23. Ernstberger, J., Chaliasos, S., Zhou, L., Jovanovic, P., Gervais, A.: Do you need a zero knowledge proof? Cryptology ePrint Archive (2024)
24. Feng, Q., He, D., Zeadally, S., Khan, M.K., Kumar, N.: A survey on privacy protection in blockchain system. Journal of network and computer applications **126**, 45–58 (2019)
25. Garousi, V., Felderer, M., Mäntylä, M.V.: Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and software technology **106** (2019)
26. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. Journal of the ACM (JACM) **38**(3), 690–728 (1991)
27. Huynh, T.T., Nguyen, T.D., Tan, H.: A survey on security and privacy issues of blockchain technology. In: 2019 international conference on system science and engineering (ICSSE). pp. 362–367. IEEE (2019)
28. Konkin, A., Zapechnikov, S.: Privacy methods and zero-knowledge poof for corporate blockchain. Procedia Computer Science **190**, 471–478 (2021)
29. Lavaur, T., Lacan, J., Chanel, C.P.: Enabling blockchain services for ioe with zk-rollups. Sensors (2022)
30. Morais, E., Koens, T., Van Wijk, C., Koren, A.: A survey on zero knowledge range proofs and applications. SN Applied Sciences **1**, 1–17 (2019)
31. Márquez Solís, S.: "zero trust chain: A design pattern for improved interoperability and security in polkadot. arXiv preprint arXiv:2304.14730 (2023)
32. Nitulescu, A.: zk-snarks: a gentle introduction (2020)
33. Open Zeppelin: Cairo contracts (2024), `https://github.com/OpenZeppelin/cairo-contracts`
34. Oude Roelink, B., El-Hajj, M., Sarmah, D.: Systematic review: Comparing zk-snark, zk-stark, and bulletproof protocols for privacy-preserving authentication. Security and Privacy (2024)
35. Partisia Blockchain Foundation: Documentation (2023), `https://partisiablockchain.gitlab.io/`
36. Porru, S., Pinna, A., Marchesi, M., Tonelli, R.: Blockchain-oriented software engineering: challenges and new directions. In: ICSE-C. pp. 169–171. IEEE (2017)
37. Qi, H., Xu, M., Yu, D., Cheng, X.: Sok: Privacy-preserving smart contract. High-Confidence Computing **4**(1), 100183 (2024)
38. Quisquater, J.J., Quisquater, M., Quisquater, M., Quisquater, M., Guillou, L., et al.: How to explain zero-knowledge protocols to your children. In: EUROCRYPT. pp. 628–631. Springer (1989)
39. Sedlmeir, J., Lautenschlager, J., Fridgen, G., Urbach, N.: The transparency challenge of blockchain in organizations. Electronic Markets **32**(3), 1779–1794 (2022)
40. Singh, A., et al.: Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities. Computers & Security **88**, 101654 (2020)
41. Six, N., Herbaut, N., Salinesi, C.: Blockchain software patterns for the design of decentralized applications: A systematic literature review. Blockchain: Research and Applications **3**(2), 100061 (2022)
42. Starkware: Hello, cairo! (2020), `https://medium.com/starkware/hello-cairo-3cb43b13b209`
43. Szabo, N.: Formalizing and securing relationships on public networks. First Monday **2**(9) (1997)
44. Wang, H., et al.: Blockchain challenges and opportunities: a survey. International Journal of Web and Grid Services **14**(4), 352 (2018)
45. Xie, T., Zhang, J., Cheng, Z., Zhang, F., Zhang, Y., Jia, Y., Boneh, D., Song, D.: zkbridge: Trustless cross-chain bridges made practical. In: ACM SIGSAC (2022)
46. Xu, X., Pautasso, C., Zhu, L., Lu, Q., Weber, I.: A pattern collection for blockchain-based applications. In: Proceedings of the 23rd European Conference on Pattern Languages of Programs. pp. 1–20 (2018)
47. Xu, Z., Chen, L.: Div: Resolving the dynamic issues of zero-knowledge set membership proof in the blockchain. In: ACM SIGMOD. pp. 2036–2048 (2021)
48. Yang, X., Li, W.: A zero-knowledge-proof-based digital identity management scheme in blockchain. Computers & Security **99**, 102050 (2020)
49. Zeiselmair, A., Steinkopf, B., Gallersdörfer, U., et al.: Analysis and application of verifiable computation techniques in blockchain systems for the energy sector. Frontiers in Blockchain (2021)
50. Zhang, R., Xue, R., Liu, L.: Security and privacy on blockchain. ACM CSUR (2019)