International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V74I7P103

Volume 74 Issue 7, 23-42, July 2026

Review Article

Gasless Verification in Decentralized Proof Systems: Design Patterns and Tradeoffs

Kyrylo Sotnykov

Independent Researcher, Tampa, FL, USA.

Corresponding Author : kyrylo.sotnykov@gmail.com

Received: 26 May 2026 Revised: 30 June 2026 Accepted: 15 July 2026 Published: 30 July 2026

Abstract *- Blockchain platforms enable transparent and immutable solutions for verification of proofs of digital claims, permissions, states, and events. However, numerous decentralized applications still force their users to operate with wallets, native tokens, and transaction costs, even though they have nothing to do with their verification purpose. This issue makes decentralized systems less user-friendly and constrains their adoption among not technically proficient Web3 users. This paper explores different patterns for gasless verification of proofs on decentralized platforms. The research focuses on five architectural patterns of such decentralized proof systems, including read-only blockchain verification, off-chain signature verification, relayer-based meta-transactions, account abstraction with paymasters, and hybrid on-chain/off-chain proof anchoring. Each architecture is analyzed qualitatively considering its usability, cost-effectiveness, decentralization, security, scalability, and implementation complexity. The results show that gasless verification enhances the usability of such systems, but at the same time transfers responsibility of trust assumptions to relayers, paymasters, backend servers, signature protocols, and off-chain data availability mechanisms. In this regard, potential risks, which include replay attacks, centralization of relayers, malicious paymaster activities, uncertainty of signers' identity, and dependence on backend servers, are discussed. Besides, the paper offers a decision-making framework for choosing one of gasless verification architectures depending on the presence/absence of state change, authority of proofs, required verification frequency, degree of decentralization needed, and level of technical maturity of system users.*

Keywords *- Blockchain, Gasless Verification, Decentralized Proof Systems, Digital Signatures, Account Abstraction.*

1. Introduction

Blockchains have developed a novel approach to trustworthiness in the digital world since they made it possible to independently verify data, transactions, and state changes in the absence of a centralized authority. Public blockchains like Bitcoin and Ethereum showed that it is possible to verify independent interactions with the help of distributed consensus, cryptography, and transparent ledgers [1], [2]. Now, there are various kinds of decentralized applications where data is not money-related but includes information about identity, access rights, property, software attestations, audit trail events, and other types of proof.

Nevertheless, decentralization itself is not enough to make blockchain adoption convenient for ordinary users. The fact is that users of blockchains still have to deal with blockchain infrastructure to be able to do anything at all. Most blockchain applications require users to install a wallet, manage private keys, learn about network fees, store native tokens, and even approve transactions to perform even simple actions. These prerequisites might be okay for advanced Web3 users, but for mainstream users who only need to verify a proof, authorization, or any data, these requirements are just unnecessary friction. This friction becomes especially apparent when we talk about verification-centric decentralized applications, because here the main concern of a user is getting the answer on the validity of a proof, not performing a transaction.

In this situation, gas fees become an important barrier. Blockchain platforms like Ethereum make users pay with native tokens for any operation of computation and any state-changing action [3]. It is a good way to protect a network from spam and to allocate resources, but it adds cost, complexity, and inconvenience to the interaction process. Users who try to verify a proof cannot understand why they should get some tokens and pay fees just for the process of interacting with the verification workflow. Therefore, applications that are based on transactions paid by a user will face hard challenges to get adoption.

Gasless verification can be seen as one of the ways to overcome this barrier. In gasless verification, a user can participate in the verification process without paying

blockchain fees directly [5], [8]. The applications might rely on different methods such as off-chain signatures, relayers, sponsored transactions, meta-transactions patterns, paymasters, backend-assisted verification, and hybrid on-chain/off-chain architecture. Gasless verification can solve the problem of onboarding and make blockchain interactions more accessible, but it has additional tradeoffs between decentralization, security, trust assumptions, infrastructure dependency, replay protection, and system transparency.

The existing literature and technical resources provide information regarding blockchain verification, digital signatures, meta-transactions, account abstraction, decentralized identifiers, and verifiable credentials separately from each other. There is, however, still a lack of reviews addressing gasless verification as an architectural design problem in decentralized proof systems.

The existing literature and research tend to cover such topics as transaction sponsorship, identity models, privacy-preserving proofs, blockchain scalability, etc., independently from each other without paying much attention to their impact on verification workflows, trust assumptions, user experience and system design. Thus, there is room for the review of gasless verification patterns and comparison of the patterns' usability, cost-effectiveness, decentralization level, security, scalability and implementation.

The novelty of this review consists in the integration of multiple independent research areas into one architectural comparison. Previous researches of decentralized identity and verifiable credentials was mostly dedicated to identity representation, issuer-verifier relations, trust management, etc. Researches of zero-knowledge proofs was dedicated to privacy-preserving proof generation. Layer-2 researches covered the topic of scalability, rollups, data availability, settlement, etc. Researches of account abstraction and meta-transactions was mostly dedicated to gas sponsorship and transactions. In contrast to the previous research, the current review covers gasless verification as an architectural problem and compares its different implementations [15], [16], [21]–[28].

This paper is dedicated to the study of gasless verification as a design approach for decentralized proof systems. By the term "decentralized proof system", this paper defines any application where users or third parties can verify claims, permissions, ownership, events, records, attestations, etc., with the help of cryptography and blockchain technology. This paper is not aimed at a concrete industrial use case; rather, the paper focuses on the general architectural challenge of creating an accessible decentralized verification process.

The main contribution of this paper is the pattern-based approach to the analysis of gasless verification architectures. This paper discusses different models, including read-only verification on the blockchain, verification with off-chain signatures, verification with relayers, account abstraction using paymaster, and hybrid proof anchoring. Each of the models is analyzed in terms of usability, costs, decentralization, security, scalability, and implementation complexity. Besides, this paper discusses common risks associated with gasless verification, including relayer centralization, signature misuse, replay attack, backend trust, and reduced auditability.

The rest of this paper is structured as follows. Section 2 gives the introduction to blockchain verification, gas fees, digital signatures, meta-transactions, and account abstraction. Section 3 describes the problem of usability in decentralized proof systems. Section 4 describes major gasless verification patterns. Section 5 compares these patterns based on practical evaluation criteria. Section 6 discusses security and trust tradeoffs. Section 7 offers a decision-making framework for choosing a suitable gasless verification model. Section 8 concludes this paper and outlines the directions for future research.

2. Background

2.1. Blockchain-Based Verification

Blockchain networks offer verification through cryptographic hashing, digital signatures, consensus algorithms, and replicated ledger state. Public blockchains allow for individual verification of whether a certain transaction was performed, a certain state of a smart contract exists, or a certain operation was done in accordance with protocol rules. Blockchain may be used in systems requiring transparency, immutability and verifiability.

The concept of Bitcoin offered a peer-to-peer approach to verification of transactions based on cryptographic signatures and transaction history [1]. The concept of Ethereum introduced a general-purpose smart contract platform that allows one to deploy programmable logic capable of handling assets, permissions, registries and application state [2]. In Ethereum, smart contracts perform deterministically on network nodes, and their execution is incorporated into the blockchain state during transaction confirmation.

There are multiple ways blockchain can help in verification-centric applications. First, it can serve as a public registry to anchor proofs, identifiers or commitments. Second, it can serve as a transparent environment to perform logic like ownership, authorization, revocation or state transition checks.

Third, it can enable auditability as historical changes will be visible to outside observers. Nevertheless, the usage of blockchain does not necessarily eliminate all trust assumptions. Application creators need to decide what data to put on-chain, what logic to run off-chain, who to control the signing keys and how users interact with the system.

2.2. Gas Fees and User Friction

For Ethereum-like networks, transactional actions that change blockchain state must consume some amount of gas. Gas is a measure of the cost of execution of actions on the network [3]. Users pay gas fees in terms of the blockchain's native token. This is an important mechanism for securing the network since it limits computation and prevents spamming. However, it makes things hard for those apps that should target a broad audience.

While in the traditional Web2 application, a person could check information using a link or QR code or just opening a website, the verification via blockchain may require installing a wallet, switching to the right network, having tokens in hand, signing something and waiting for the transaction confirmation. All this is quite easy to do if everything is done correctly, but the number of necessary actions grows with each step. The higher the number of actions needed from a user, the less chances that it all will go fine in real life.

Here we have an obvious contradiction between decentralization and usability. Blockchain allows the user to have more control and more understanding of what is going on, but it demands some knowledge and even finances to operate directly with the blockchain. Gasless verification tries to overcome it by giving access to verification tools without transaction fees and any additional work with blockchain.

2.3. Digital Signatures and Off-Chain Proofs

The first essential component is a digital signature. A digital signature allows proving possession of a private key without revealing that key. Any person who has the public key can verify that a signature was created using this key and that the signed message has not been changed since signing. Modern signature schemes provide secure ways to create and validate digital signatures [14].

In many decentralized applications, signatures can be used to authorize some actions in the off-chain process prior to sending them to a blockchain. In some cases, similar schemes of authorization are implemented in permits or transfer authorizations [6], [7]. The idea is that one can sign the message with a description of a planned action and not send a transaction himself, but leave it to be sent by someone else. In this case, Ethereum Improvement Proposal 712 provides a standard way of structured data hashing and signing, which makes signed messages more readable and less prone to mistakes compared to just byte arrays [4].

Signature-based off-chain actions can be a part of a gasless verification process because validation does not always require a transaction paid by a user. In some cases, the application can validate a signed message locally in the browser, from a backend or from a read-only blockchain request. However, developers have to take into account certain limitations of such a system. For instance, one needs to protect against replay attacks, bind a signature to a certain domain or chain, define expiration policies, and make sure that users know what they sign.

2.4. Meta-Transactions and Relayers

Another widely used technique of building gasless DApp interactions is called meta-transactions. Meta-transaction design pattern allows users to sign a message approving an operation while the other party (relayer) performs the actual blockchain transaction and pays gas fees on behalf of the user [5]. The smart contract receiving the transaction checks the user's signature and performs the requested operation only in case of its validity.

The ERC-2771 standard proposes a way of implementing native meta-transactions with the help of a trusted forwarder [5]. In that case, the recipient smart contract relies on the forwarder contract to verify or forward the transaction data properly. The user does not directly invoke any methods of the application contract. Instead, the relayer submits the transaction via the forwarder, and the application contract restores the user context in the forwarded invocation. It can provide better UX since users do not have to keep native gas tokens to interact with the application.

Such gasless transaction frameworks in practice usually implement the described model using relayer infrastructure when the user signs the off-chain request, while the relayer performs the transaction paying the gas fee [9]-[11].

It makes gasless transaction systems very versatile, but creates additional issues of trust and operability. In case the relayer is not available, the whole gasless flow stops working. In case the relayer gets too much power, the system becomes more centralized. In case the contract fails to check signatures properly, attackers can perform some actions with the request, like replaying.

2.5. Account Abstraction and Paymasters

Account abstraction is a relatively novel approach that seeks to make blockchain accounts more programmable and friendly to users. ERC-4337 implements an account abstraction framework through the use of UserOperations, bundlers, and paymasters without needing modifications of the Ethereum consensus layer [8], [12]. Another production-ready account abstraction toolchain leverages paymasters and sponsored transactions to minimize the friction associated with wallets and gas for the final user [13]. In this framework, users can interact with smart contract accounts instead of traditional externally-owned accounts. Such interactions will have features including sponsored transactions, customized authorization logic, session keys, social recovery, and alternatives to gas payments.

Paymasters are particularly important when it comes to gasless verification. A paymaster can help in sponsoring

transaction fees on behalf of users according to certain pre-defined conditions. In one application, for instance, it can choose to pay for verification transactions due to the low value and high frequency of these actions that should not consume user-funded gas. It results in a more familiar flow of events for users of mainstream platforms since the application pays for infrastructure cost similarly to Web2 applications paying server cost.

Nevertheless, account abstraction still poses certain tradeoffs. Paymasters need funds and policies. Bundler infrastructure creates a new point of dependency. Smart contract accounts become more flexible but also more complex in comparison to traditional ones. Therefore, account abstraction offers substantial usability advantages but demands proper engineering efforts to avoid creating obfuscating or centralized verification flows.

The recent research on the topic of gas sponsorship and paymaster models makes it clear once again that account abstraction is an approach improving usability but poses new optimization, centralization, and security problems.

2.6. Decentralized Proof Systems

A decentralized proof system refers to any architecture of an application that enables a user to verify claims, states, permissions, events, or relationships with cryptographic or blockchain evidence. Proof systems could use methods such as smart contract execution, digital signatures, decentralized identifiers, public keys, hash commitments, timestamping, or off-chain attestations [15], [16]. The actual proof could be used to prove a variety of use cases, like access rights, membership, asset ownership, process completion, software integrity, audit events, etc.

Previous research on decentralized identifiers and Web3 identity systems also demonstrates that the choice of methods, levels of abstraction, and composition creates crucial tradeoffs in decentralized verification systems [19], [20].

Decentralized proof systems usually include both on-chain and off-chain components. It would be costly, inefficient, and even a violation of privacy to store everything on-chain. Hence, many proof systems only store minimal on-chain elements like hashes, identifiers, or statuses and store more detailed information off-chain. Then, the verification process will involve comparing the data stored off-chain with the on-chain commitment or the comparison of signatures and the authorized public key.

This design creates a need for gasless verification, since if verification is supposed to occur very often, then it would be unfeasible for each verification process to pay gas. On the other hand, moving all verification processes off-chain might decrease transparency and require trusted infrastructure. Thus, the critical part of the design process is to decide what should be decentralized and what off-chain and how to ensure reliable proof without unnecessary friction.

The flowchart shown in Figure 1 demonstrates the general process of gasless verification in a decentralized proof system. The process is divided into four separate parts: proof creation, cryptography verification, optional indexing by a backend, and blockchain-based commitment verification. Such a design allows conducting a basic verification without gas payment or blockchain transaction submission.

2.7. Related Work and Research Gap

Decentralized verification in recent literature has several distinct research streams. First of all, there is literature on decentralized identifiers and verifiable credentials. These papers explore how identities, issuers, holders, verifiers, verification processes, and trust relations could be modeled without complete dependence on centralized identity providers. DID and VC models play an important role in decentralized proof systems in terms of defining the concept of signer authority, the process of verification, and revocation. However, this research stream does not cover issues such as gasless verification processes, transaction sponsorship, and user experience issues connected with native token requirements [15], [16], [19]–[21], [24].

Secondly, the relevant research stream is focused on the concept of zero-knowledge proofs. ZKP-based techniques allow a party to prove that a certain statement is true without exposing information needed to verify it. ZKP techniques are highly applicable for privacy-preserving verification, identity sharing, and selective disclosure.

However, research on zero-knowledge proofs usually concentrates on cryptographic privacy, generation and verification of proofs, rather than on the architecture of gasless user interaction. For example, in real-life implementations of zero-knowledge proof-based verification, it may be necessary to choose how and where proofs will be created, who will pay for on-chain proof verification, and how the verification result will be presented to the user [22], [23], [27].

Thirdly, there is literature on Layer-2 scaling solutions, rollups, and data availability. Layer-2 systems aim at reducing the fee per transaction and increasing its throughput while trying to maintain security guarantees provided by the underlying blockchain. Such techniques are applicable to gasless verification due to reduced fees and off-chain execution that improve the scalability of the process. However, Layer-2 systems introduce additional assumptions that are connected with bridges, finality, proof submission, fraud, validity proofs, and data availability [25], [26], [28].

Fourthly, there is literature on meta-transaction frameworks and account abstraction. Meta-transactions enable separation between transaction authorization and execution, and account abstraction enables creation of smart

accounts, bundlers, and paymasters that can sponsor or customize transaction execution. Such mechanisms facilitate gasless verification processes; however, they bring with themselves issues of relayer dependency, paymaster misuse, infrastructure complexity, and trust assumptions [5], [8]–[13], [17], [18].

In spite of valuable contributions made to gasless verification processes, existing literature does not feature a comprehensive comparative analysis of such processes for decentralized proof systems. This paper aims to fill the gap with a comparison of read-only verification, off-chain signatures, relayer-based meta-transactions, paymaster-based account abstraction, and hybrid proof anchoring [21]–[28].

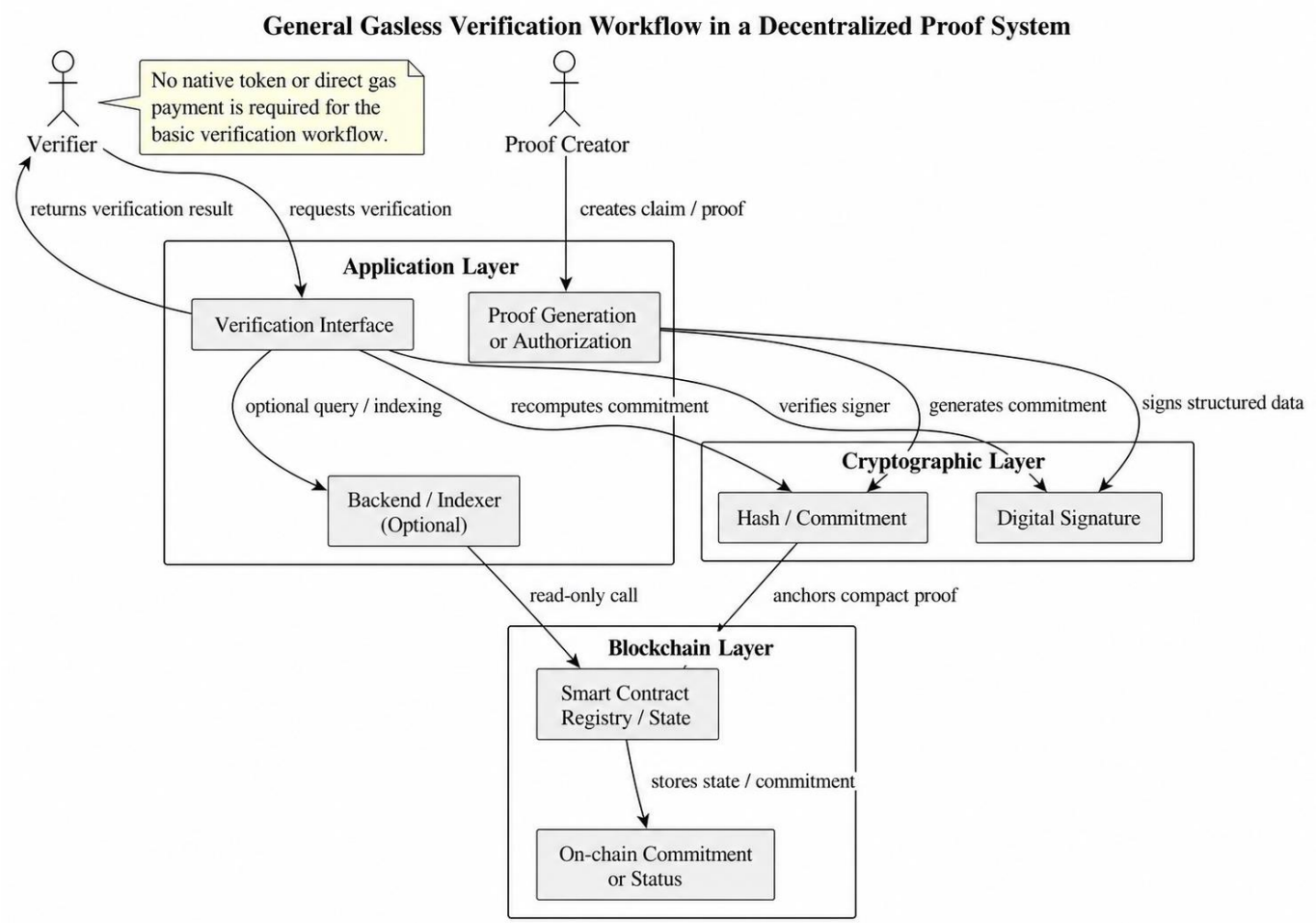

Fig. 1 General gasless verification workflow in a decentralized proof system

3. Problem Statement: User Friction in Decentralized Proof Systems

Decentralized proof systems aim at enhancing transparency by means of cryptographic and blockchain-based verification of claims, permissions, states or events. In theory, it provides a transparent and tamper-proof alternative to the existing centralized verification methods. However, in practice, the demands of the blockchain interface often contradict the needs of average users and organizations.

For a proof system to be valuable for the intended verifier, it is necessary for the verifier to gain access to the verification outcome. Should the verification process involve the installation of a wallet, getting hold of native tokens, network change, transaction approval or any other blockchain-specific error messages, the proof system may turn out to be inefficient even if the cryptography itself is sound.

It is especially relevant for verification-oriented applications because, in most cases, the aim of the verifier is simple: find out whether a claim is true, false, expired, revoked, inconsistent or any other state. The verifier does not necessarily want to be a part of the blockchain.

The following section highlights the major sources of friction that drive gasless verification patterns.

3.1. Wallet Dependency

Many blockchain products take it for granted that users will be using an appropriate wallet. Indeed, wallets are critical for signing transactions, handling private keys, and interfacing with the decentralized applications. However, wallet dependency raises an onboarding challenge for many users who are new to blockchain technology.

In the context of verification, this poses a problem of justification. A user would only need to verify something once or from time to time. To ask this user to go through all the processes of downloading a wallet, handling a seed phrase, and managing wallet permissions might be too much to ask for a simple action of verifying something.

This issue is particularly relevant in those situations where the verification does not imply any transfer of value or change of ownership on the part of the user. In these instances, wallet dependency makes a decentralized proof system seem less convenient than a centralized one.

Wallet dependency also leads to compatibility issues. Different users can have different wallets, different networks, different account types, and even different browsers. Mobile users could face additional difficulties in switching between different apps and signing in to pages.

3.2. Native Token Requirement

The other point of friction lies in the need to own a blockchain's native token for gas fee payments. Small gas fees already represent an obstacle in adoption, since the users have to obtain the needed token prior to interacting with the system. For this purpose, users need to use exchanges, bridge tokens, perform identity verification, or learn the specifics of fee payment in the respective network.

Verification-focused systems suffer from the mismatch between native token requirements and user expectations. In conventional online services, verification is generally viewed as an informational task rather than as a paid blockchain operation.

If the system asks for the gas to be paid to verify the proof, this system will seem complicated and illogical to the users. This would decrease adoption in the environment when users expect verification to be a quick and cheap procedure performed via familiar web interfaces.

Additionally, native token requirements generate operational uncertainty due to the variability of gas prices according to the level of network usage. What may be affordable at the moment might become costly during periods of congestion. It is hard to estimate gas fees for users and application providers alike. Gasless solutions attempt to solve the problem by taking transaction costs away from users and onto applications, relayers or sponsorship services.

3.3. Transaction Complexity

Blockchain transactions create terms that may confuse many users, such as gas limit, nonce, confirmation, pending, chain id, failed transactions, and block explorers. Even if the transaction is simple in a technical sense, the user interface may disclose aspects that are hard to understand for a layperson.

The output of verification processes should ideally be explicit – whether the proof is valid, invalid, expired, revoked or unverifiable. Nevertheless, direct blockchain transactions may make some intermediary states appear, which may obscure the result. It is possible that a user sees the status of a pending transaction and does not know the status of the proof validation. A failure of a transaction may not be clear if it happened because of an invalid proof, lack of gas, network problems, bugs in the contract, or wallet issues.

Such an approach may create a difference between technical perfection and user comprehension. A decentralized proof system may be highly reliable and transparent, but it may still be considered a failed product by the users if they are not able to perform all the steps without issues.

3.4. Verification Frequency and Cost

There are some proof systems that are likely to require verification often. For instance, an application might have to verify access permissions, events, membership, software integrity, or state commitments multiple times for different users and sessions. If the verification is to be carried out by a transaction that alters the state in the blockchain, then it will amount to a lot of gas.

All verification transactions do not have to modify the state on the blockchain. There are instances where the transactions can be done through read-only contracts, local signature verification, cached proofs, or computation off the chain with on-chain commitment. The key here is distinguishing between what has to be verified through the decentralized consensus system and what can be done without making modifications to the chain.

This is not only a user experience enhancement but also an architectural optimization technique. It helps save on cost, improves responsiveness, and enables scalability. However, in all these, the integrity of the verification result should never be compromised.

3.5. Trust Assumption Visibility

One possible issue with gasless verification is that increased usability would create misunderstandings about the underlying trust assumptions. For example, if the application pays for gas on behalf of the user, routes transactions via a relayer, does signature verification in the backend or shows only the result of verification in a simplified way, the user

might not be aware of which part of the process is decentralized and which one requires application-managed infrastructure.

The question here is whether there is enough transparency. It is possible to have a solution that is called "decentralized verification" but still relies very much on centralized services in terms of transaction submission, proof interpretation, caching or even UI rendering. Sometimes it can be acceptable when trust assumptions are stated clearly, and the user can validate the result on their own. Sometimes, such heavy backend dependencies will contradict the point of using blockchain technology.

So the problem is not only about the complexity of the process of blockchain verification for the user. The real challenge lies in the need to minimize user friction while avoiding misleading decentralization claims.

3.6. Design Goals

Given these difficulties, the verification architecture should try to meet certain design goals.

The first goal is that the system should reduce user onboarding needs. The users should not need to obtain tokens, learn about gas payments, or perform unnecessary wallet transactions to verify anything. The second one is that the verification outcome should be comprehensible and meaningful. The system should produce comprehensible status outputs rather than expose blockchain transaction intricacies. The third one is that the architecture should retain security via validation of signatures, protection from replay, and attachment of proofs to the right context. The fourth goal is transparency of the system, where verification is possible independently. The fifth goal is to be clear about the underlying trust assumptions, particularly when relayers, paymasters, or backend systems are used. This set of goals provides the basis for the design patterns described in the following section.

4. Gasless Verification Design Patterns

The approach of gasless verification can be done using various architectural patterns. The architectural patterns vary depending on the distribution of responsibilities among the users, application backends, smart contracts, relayers, paymasters, and off-chain verification modules. There is no single best pattern; the correct choice will depend on the security needs, number of users, cost structure, decentralization needs, and frequency of verification of the application.

The following section will cover five different patterns: blockchain read-only verification, off-chain signature verification, relayer meta-transactions, paymaster account abstraction, and proof anchoring.

4.1. Read-Only Blockchain Verification

The most straightforward approach to gasless verification is read-only blockchain verification. In this pattern, the application queries the existing blockchain state rather than submits a new transaction. As read calls do not change the state of the blockchain, they do not consume any gas from the user using the standard RPC call.

Read-only verification can be helpful if the necessary proof, registry entry, commitment, or status has already been saved in the blockchain. The application may check whether a certain identifier is registered, if the state commitment is correct, if the public key has the right to access, if the proof has been revoked, etc. In this way, the user obtains the proof without signing the transaction and paying for gas.

The benefits of read-only verification are its simplicity, speed, and low costs. It provides a close relationship with the public blockchain state as it operates the state directly from a smart contract or a node provider. Read-only verification also does not carry many risks related to transaction submission, such as transaction failure, problems with nonces, and incorrect gas estimates.

On the other hand, there are some disadvantages of this pattern. It is impossible to verify something that does not exist in the blockchain, to update the state, to trigger certain events or run smart contract logic that modifies the storage using read-only calls. Besides, in many cases, blockchain reads are performed by centralised RPC providers. If the user cannot select or validate the RPC source, then the result will depend on the infrastructure managed by a third party.

Read-only verification is the best choice for frequent checks where the proof state is already available on the blockchain and does not need any blockchain record creation.

4.2. Off-Chain Signature Verification

An off-chain signature verification involves the ability of the system to validate the cryptographic statement without an on-chain transaction. In this case, the party generates a signed statement, and then, its validity is checked using the corresponding public key or blockchain address.

The pattern is helpful in cases when the proof is based on the authenticity and integrity of the signed message rather than on the on-chain execution of a transaction. The signed message might denote authorization, consent, ownership declaration, timestamped confirmation, or any specific state-related claim. If the verifier can ensure that the signature is correct and the signing party is trusted or authorized, then no on-chain execution is needed.

There are some safety standards, like EIP-712 for structured data signing [4], which make the pattern more secure since they allow signing typed messages with specific

fields and domain separation. The other examples of the same type are permit style and transfer authorization standards, showing how the signed messages can authorize the operation and separate the user approval and transaction execution [6], [7]. Domain separation is crucial for this pattern since it helps to ensure that the signature intended for a specific application or chain cannot be used elsewhere. The pattern requires nonces, expiration dates, chain identifiers, and message formats that depend on the specific purpose.

The main advantage of the off-chain signature verification is that it is very efficient. It is fast, cheap and immune to network congestion. Also, it enhances the user experience because verification does not require gas payments.

However, the pattern relies mostly on good key management and trust in the signing parties. If the key is compromised, old proofs will still be valid until the system implements certain revocation or expiration measures. If the message format is not properly developed, users will sign data without understanding what it is.

Off-chain signature verification is the best choice for those systems where the validity of the proof can be determined based on the authority of the signing party and the integrity of the message.

4.3. Relayer-Based Meta-Transactions

In relayer-based meta-transactions, users can authorize the action without sponsoring the gas cost of the transaction. Here, the user signs a message with details about the desired operation. After that, the relayer forwards the transaction to the blockchain and sponsors the gas costs. The receiver smart contract verifies the signature before executing the requested operation.

This pattern allows separating the user's authorization from the execution process. Still, the user provides the cryptographic confirmation, but the relayer performs all interactions with the blockchain. There is a standardization of the pattern in the form of the ERC-2771 protocol. It uses the trusted forwarder for recovering the user context in the forwarded calls [5]. Also, there are real-world examples of implementation of the relayer-based meta-transaction flow via developer frameworks like OpenZeppelin Defender and OpenGSN [9]–[11]. It allows developers to provide a gasless interaction experience with the state-changing actions on the blockchain.

Relayer-based meta-transactions are useful for cases where the verification involves the registration of an event, updating the authorization status, marking a claim as used or publishing a new commitment. In those cases, the read-only verification alone will not be enough, and forcing the user to sponsor gas would add additional complexity.

As the main benefit of this pattern, we can name improved usability. The users will be able to interact with the system without having its native token. In addition, the application will have complete control over sponsoring, rate limiting, and optimizations of the transaction submission process. However, in relayer-based solutions, there is additional operational risk, as the failure of the relayer will prevent executing the gasless flow of the system. If the relayer censors or selectively submits requests, the user may not be able to access the system. Moreover, if the contract is too trustful to the forwarder, there will be some security issues.

For mitigating these risks, relayer-based solutions should employ such techniques as the strict signature verification, replay prevention, request expiration, domain separation and clear fallback options. Wherever possible, the user should be able to perform the transaction themselves in case the relayer is not available.

4.4. Account Abstraction with Paymasters

The gasless verification concept in account abstraction implies the ability of smart contract accounts to introduce their own transaction validation logic. ERC-4337 presents the architecture where users provide their transactions in the form of UserOperations to bundlers, while paymasters sponsor gas costs of these operations under certain conditions [8], [12]. Toolset and recent research demonstrate that paymaster sponsorship helps increase usability of gasless transactions but requires the proper policy design and security precautions to avoid abuses [13], [17], [18]. In such an architecture, applications can implement verification flows where users do not have to use native tokens.

In the paymaster-sponsored flow, applications or sponsors determine the conditions of operation sponsorship by setting certain rules. They may decide to sponsor only verification transactions, only cheap operations, only requests of eligible users, or even only operations satisfying certain conditions of validation. In other words, sponsorship gives more control over the process of gasless transactions compared to the relayer model.

There are also some extra possibilities of increased usability in the account abstraction approach. Smart contract accounts allow developers to implement session keys, multi-factor authorization, spend limit, social recovery, and any other forms of user identification. Such an approach could make decentralized proof systems available to users unfamiliar with blockchain wallets.

At the same time, account abstraction complicates the system architecture as it requires using of bundlers, paymasters, implementation of smart contract accounts, and other monitoring services. Paymasters have to be funded and secured from abuses. Sponsorship policies should protect users from having their funds robbed by repeatedly sending

invalid requests. Moreover, developers should also guarantee the transparency of verification results.

Paymaster-based account abstraction is suitable for applications with frequent interaction with a blockchain network that require the sponsorship infrastructure.

4.5. Hybrid On-Chain and Off-Chain Proof Anchoring

Hybrid anchoring refers to combining proof with on-chain commitments and off-chain data/computation. With this anchoring method, the blockchain contains only a small proof element, for example, a hash, identifier, commitment, root, or status flag. Detailed information will be stored off-chain. The process of verifying hybrid proof involves comparing the off-chain evidence with on-chain commitment and checking relevant signatures and metadata.

This kind of anchoring is popular because anchoring all data on the chain is usually expensive and inefficient in terms of scaling and privacy concerns. Thus, with the help of this approach, systems can keep tamper evidence while minimizing costs and maximizing efficiency. Gasless verification is possible due to off-chain checks or read-only operations with the blockchain.

For instance, a proof system may provide a verifier with an off-chain proof package that includes a signed claim, a timestamp, some metadata, and a link to an on-chain commitment. The application will calculate a hash of the off-chain information, will check whether it matches the commitment stored on the blockchain, will verify signatures, and will return a result without using a paid user's transaction. This transaction is needed only when the system requires updating the commitment or changing its state.

The main benefit of hybrid anchoring is the balance between efficiency, integrity, and auditability, which is provided by blockchain. However, it has to be developed carefully. For example, off-chain data may become unavailable, making commitment useless. Verifiers will not be able to reproduce the results if a centralized backend is used to interpret proof packages. Moreover, users may have no idea about the exact content of their proofs if commitments are not structured correctly.

Hybrid anchoring is suitable for proof systems that require integrity and auditability but do not allow complete on-chain storage or user-paid verification.

4.6. Pattern Comparison Overview

Patterns are different in their assumptions and tradeoffs. Verification of off-chain signatures is efficient and flexible but relies on signer authorities and protection against replay attacks. Verification via a relayer allows state-changing gasless transactions but requires the presence of a relayer. Read-only blockchain verification ensures a strong connection to the public state but is limited to read-only operations. Paymasters provide programmable sponsorship and improve user experience, but add to infrastructure complexity. Hybrid proof anchoring ensures scalability and auditability but requires well-coordinated efforts of off-chain entities that generate proofs and on-chain entities that store commitments to those proofs.

Patterns above are not mutually exclusive. An advanced decentralized proof system could utilize several patterns at once. For instance, it might perform authorization off-chain, use commitments to proofs on-chain for audit purposes, perform read-only queries from the blockchain to prove that the proof is consistent with public state, and use a paymaster to perform selected state-changing operations. An important architectural decision is not about the usage of gasless verification, but rather what part of the verification workflow can be gasless, what has to stay on-chain, and what trust assumptions have to be disclosed to users.

Figure 2 illustrates the major gasless verification patterns covered in this paper. It demonstrates how user interactions, signature verification, relayers, paymasters, smart contracts, and proof data could be used in various verification systems. Patterns above can be used in combination in multi-layer systems.

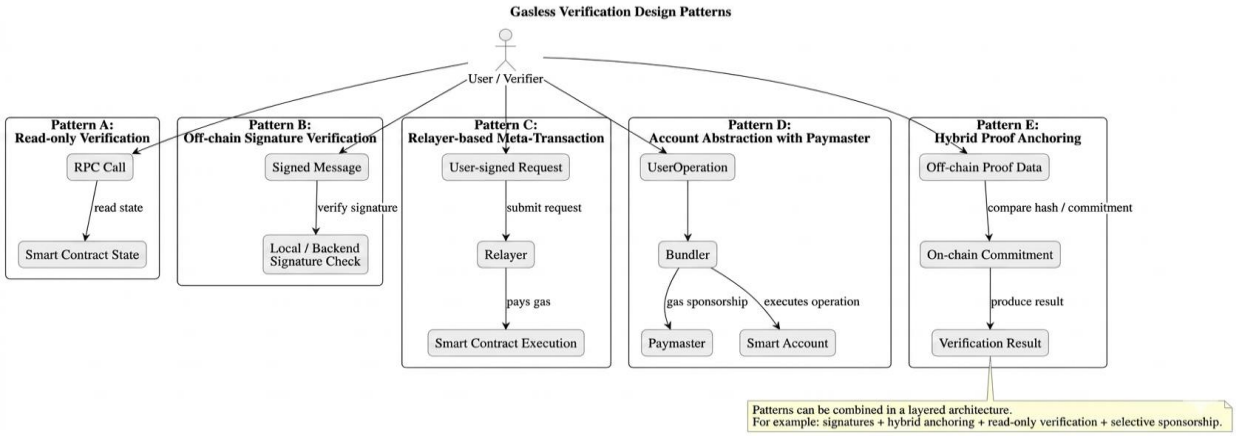

Fig. 2 Comparison of major gasless verification design patterns

5. Comparative Analysis of Gasless Verification Patterns

As seen in the patterns presented in Section 4, the problems related to the gasless verification have different solutions. There are some patterns that favor simplicity, and there are others that consider decentralization, programmability, cost efficiency or end-user experience. With the many variations of decentralized proofs systems and varying levels of risks involved, choosing an appropriate pattern is essential, hence the need for comparison.

This section evaluates the five patterns in terms of six attributes, including usability, cost efficiency, decentralization, security, scalability and implementation difficulty. These criteria are intended to help developers and system architects identify the most appropriate pattern for a given verification workflow.

5.1. Evaluation Criteria

Usability is defined by the ease of completion of a verification action for an average non-tech-savvy user. A highly usable pattern reduces the dependency on a wallet, native tokens, transactions, network switching, and blockchain-specific error handling. In verification-oriented systems, the usability is especially important since the user may reasonably expect a straightforward, immediate answer.

The cost efficiency is defined by the expenses required to execute a verification flow. A cost-efficient pattern should not use unnecessary state-changing transactions, minimize the gas, and reduce the infrastructure cost. The cost should be considered from both perspectives – the user and the application provider. The pattern could be cost-free for the user but still impose additional payments on the application (relayer, paymaster, RPC, backend costs).

Decentralization is measured by the possibility to reproduce a verification result using public blockchain state, cryptographic signatures, or a public verification algorithm. The gasless patterns typically increase usability by introducing additional intermediaries, but in turn, they make the verification less decentralized since it now requires trust in an additional service.

Security is measured by the resistance of the pattern to any form of misuse, manipulation, replay attacks, unintended or malicious executions, key thefts, and any form of abuse of the infrastructure. Not only is the security property intrinsic to the pattern itself, but it is also dependent on the specific implementation details, e.g., signature formats, nonce generation rules, expiration, access control, and contract validations.

Scalability is the ability of the pattern to support multiple verifications, either periodically or continuously. While some verification patterns could potentially have no scalability problems at all, others may require periodic verifications or be designed to scale verification processes of many users/automations.

The implementation complexity is defined by the engineering difficulty required to develop, deploy, run, and support a verification system pattern.

5.2. Read-Only Blockchain Verification

The Read-only blockchain verification pattern works great for usability and cost efficiency in case the proof state is already on-chain. There are no transactions submitted and no gas paid by the user. The application asks the state of the smart contract via RPC call and displays the result right away. It seems like an ideal choice for a verification workflow.

Speaking of the pattern's decentralization, read-only blockchain verification can be highly decentralized in case users are able to reproduce the same query using public addresses of smart contracts on their own. However, the actual degree of decentralization will be lower if the application queries a single RPC provider and/or does not provide users with information about the exact query. In other words, while the blockchain state is publicly available, the ability of the user to access this state is provided by centralized systems.

Security of this pattern is relatively easy to estimate as there is no modification of blockchain state, and therefore no user-paid transactions, risks associated with relayers' execution, and opportunities for sponsored transaction abuse. On the other hand, care needs to be taken in dealing with contract addresses, chain IDs, RPC responses, and cached data. A malicious frontend can show a wrong result even in the case of the right state stored on the blockchain.

Scalability of the read-only verification is decent and, in combination with caching, indexing, and multiple RPC providers, can be very strong. However, it should be noted that the pattern allows performing only verification tasks, which do not require recording a new state.

In summary, read-only verification is an optimal choice for low-risk and frequent checks in case the required evidence is on-chain.

5.3. Off-Chain Signature Verification

Off-chain signature verification offers great usability and cost-efficiency because the process of verification does not require any transactions performed on the blockchain. The verifier can check a signed message either locally or through an application service. That is why this solution works perfectly well for lightweight proof validation.

The level of decentralization depends on the way signer authority is established. If the public key of the signer or his

signing address is known independently, on-chain anchored, or distributed through some transparent registry, then the pattern can guarantee strong verification capabilities. If the verifier needs a centralized backend in order to understand whether a signer is trusted, then decentralization is weaker.

A high security level requires the correct design of messages. Secure signed messages should have a specified domain, intended action, nonce, expiration time, chain identifier if required, and context-specific fields. Absence of these features means that signatures can be used in different contexts than they were initially intended. Incorrect signature prompts can lead to mistakes by users who will agree to perform an action without understanding its purpose.

Scalability is high since the operation of signature verification is less costly than any transaction on the blockchain network. Proof verification can be performed frequently at no additional cost. Nevertheless, off-chain signature solutions need mechanisms of proof revocation and key rotation in case of keys being leaked or the signer losing its authority.

Implementation complexity is medium. Signatures verification itself is a simple process, yet secure production usage requires correct handling of data encoding, canonicalization, protection from replay attacks, setting expiration time, and proper design of the signers' registry.

5.4. Relayer-Based Meta-Transactions

Relayer-based meta-transactions increase usability because they allow users to authorize state-changing operations without the need to pay the gas price. User signs requests, and then the relayer sends transactions and pays for the network. Therefore, this pattern is applicable in the case when the verification process requires changing state on the blockchain, but does not require the user to have native tokens.

Cost efficiency depends on who will pay for the relayer service and transaction frequency. While the user experience will be gasless, the application will still be paying gas and infrastructure fees. Therefore, a relayer-based pattern should have rate limiting, request validation, fee checking, and protection from abuse.

Decentralization characteristics are mixed in this pattern. On one side, state changes performed by the final transactions are on the blockchain and are publicly verifiable. However, on the other hand, the relayer becomes a central point. If users do not have an opportunity to go around the relayer, then the relayer will have control over whether signed requests will be sent or not. Thus, it may lead to censure and other problems regarding availability and transaction prioritization.

Security of the pattern will depend on how strictly the request is validated at the contract level. Contract should not trust all data provided by the relayer and should perform checks like verification of the signature of the request, expiration time of the request, checking nonces and correctness of forwarded call.

Scalability will depend on gas price, relayer processing capabilities, and network congestion. Relayer could batch and prioritize transactions, but it cannot reduce the gas cost of on-chain processing. High-volume systems using a relayer-based approach will be expensive unless some selected actions are performed on the chain.

Complexity of implementation is increased in compare with read-only or off-chain verification patterns. Additional logic needs to be added to manage the relayer, forwarder contracts, request queue, transaction failures, and abuse protection mechanisms. Still, this pattern is useful when gasless state changes are required, and transaction costs can be paid.

5.5. Account Abstraction with Paymasters

Account abstraction with paymasters is a more programmatic and gasless pattern compared to other designs relying on relayers. Sponsorship can be defined with the help of paymasters, allowing applications to sponsor specific operations in accordance with certain policies. Thus, it is particularly suitable for decentralized proof systems involving multiple blockchain operations without sacrificing usability.

Usability is high as users can perform actions in smart contract accounts and do not have to possess native tokens. Session keys, alternative authentications, and authorization policies can further enhance usability. In particular, in the eyes of users, the system might be close to an ordinary web application.

Cost efficiency is dependent on the sponsorship policy implemented. Paymasters can prevent operations from being sponsored and thus decrease costs. However, the application needs to compensate for sponsored gas and protect the paymaster from possible abuse. Malicious users could drain the paymaster's balance due to poor configuration.

Decentralization is influenced by the presence and openness of the infrastructure for account abstraction. The pattern includes bundlers, paymasters, smart contract accounts, and application-defined policies. If these elements belong to one service provider, there might be no practical decentralization of the workflow. If users can change service providers or submit operations using different infrastructure, the decentralization will be higher.

Security is complicated for account abstraction. Flexibility of the approach increases security risks. Logic of smart contract account, paymaster validation, bundler behavior, session key permission management, and sponsorship policies need to be thoroughly considered. The

system has to prevent possible attacks on unauthorized operations, replay attacks, abuse of the paymaster, and privilege escalation.

Scalability is good if the account abstraction infrastructure is well developed and sponsorship policies are optimized. However, the pattern requires more operation and monitoring knowledge than a simple one. Therefore, the pattern is recommended to be applied to those applications that have a complicated user interface.

5.6. Hybrid Proof Anchoring

Hybrid proof anchoring combines on-chain integrity with off-chain scalability. In such a scheme, storing only small-sized commitments or status flags on-chain enables reducing gas costs while maintaining a public reference point for verification. Detailed evidence will be stored off-chain, where it is easier to handle.

Usability is good since, in such cases, the application displays the result of verification without involving users in interacting with blockchain technology. Also, cost-efficiency is good since most of the verifications can happen offline, off-chain, or using read-only calls to the blockchain. Only chosen events, commitments, or updates should be committed to the blockchain.

Decentralization level will depend on the ability of verifiers to access and verify the evidence off-chain. In case all proof package data, verification algorithm, and on-chain commitment are publicly available, decentralized verification becomes possible. In case only the application backend has the capabilities of interpretation and access to evidence, decentralization will be lower.

Security will depend on the integrity of both on-chain and off-chain elements. Hash commitments can be used to detect modifications, but they cannot guarantee availability. In case the off-chain evidence is missing, unavailable, or managed by one side, the hash commitment will not be enough. Therefore, availability mechanisms should be implemented, the proof format should be transparent, and commitments interpretation rules should be defined.

One of the major strengths of this anchoring pattern is scalability. Hybrid anchoring allows avoiding redundant on-chain storage and supports frequent verifications. Complexity of implementation is medium to high, depending on the implementation of proof packages, storage, signing, indexing, and verification capabilities.

Hybrid proof anchoring is usually the most balanced pattern for real-world decentralized proof systems because it allows using blockchain for providing tampering proof and audibility without forcing every verification to be an on-chain operation.

5.7. Summary of Comparative Findings

As seen from the comparison table, gasless verification is not one technique but a range of architectural options. Read-only verification is easy and fast, but it works only for the on-chain state already present. Off-chain signature verification is efficient and scalable, but it relies on the signer and replay prevention. Relayer meta-transactions allow gasless state transitions but require relayers. Account abstraction using paymasters is very powerful in terms of programmability, but difficult to operate. Hybrid proof anchoring can be seen as a good compromise among transparency, costs, and scalability, but it needs to take care of coordination between on-chain/off-chain parts.

The following Table 1 is an overview of comparative results on the basis of five gasless verification patterns. It describes the most suitable use case, main advantage, limitation, and security considerations for each pattern.

5.8. Emerging Verification Approaches

Apart from the above-mentioned five key categories of gasless verification design patterns, there are a number of emerging technologies that influence the design of decentralized proof systems.

Zero-knowledge proofs represent a privacy-preserving method of verifying proofs where a prover can prove the validity of a certain statement without disclosing underlying information about this statement. This method is applicable when verification relies on selective disclosure and when the underlying data needs to be kept private. However, zero-knowledge proofs increase the complexity of proof construction, circuit building, trusted setup assumptions of certain designs, verification, and development tools. Moreover, while zero-knowledge proofs help improve privacy in gasless verification, they do not address other major challenges like transaction sponsorship, reliance on relayers and user onboarding [22], [23], [27].

Decentralized identifiers and verifiable credentials represent another key direction. DID and VC models define the way signers' authorities, issuers' identities, verification and revocation status could be represented in decentralized trust systems. DID and VC models are crucial for the design of proof systems because they provide for portability and independent verification of claims. However, the DID/VC standards alone do not define who will pay for blockchain transactions, what constitutes a correct implementation of gasless workflows, or how backend-assisted verification should reveal trust assumptions [15], [16], [19]–[21], [24].

Scaling and Layer-2 solutions, as well as rollup architectures, are another set of technologies that influence the design of proof systems. By reducing transaction fees and offloading transactions from the blockchain itself, such approaches help build gasless verification workflows. At the

same time, they introduce a new set of challenges related to the security of bridges, delayed finality, data availability, validity proofs, fraud proofs and dependencies on Layer-2 infrastructures. In other words, Layer-2 solutions can reduce gas friction but add new verification assumptions that need to be made clear for users and auditors [25], [26], [28].

Another group of technologies includes growing account abstraction tooling around smart accounts, paymasters, bundlers, session keys and sponsorship policies. This approach allows making decentralized apps resemble conventional web apps in terms of hiding native token payment and allowing multiple operations. However, the same technologies bring risks associated with paymaster misuse, sponsorship policy mistakes, dependency on bundlers, security of smart accounts, responsibility for transaction signing [8], [12], [13], [17], [18].

All these emerging technologies show that gasless verification should not be regarded as a task limited only to paying for transaction fees. This is an architectural problem related to privacy, identity, scalability, infrastructure dependency and trust representation.

Table 1. Comparative matrix of gasless verification design patterns.

Pattern	Best Use Case	Main Advantage	Main Limitation	Security Concern
Read-only blockchain verification	Checking existing on-chain state, registry entries, commitments, or status flags.	Simple, low-cost, no user-paid gas, strong connection to the public state.	Cannot update blockchain state or record new events.	Incorrect RPC data, misleading frontend output, and hidden provider dependency.
Off-chain signature verification	Validating signed claims, permissions, approvals, or attestations.	Fast, scalable, no transaction required.	Depends on the signer authority and key management.	Replay attacks, vague signing prompts, expired or compromised keys.
Relayer-based meta-transactions	Gasless state-changing operations authorized by user signatures.	Users can trigger blockchain actions without holding native tokens.	Relayer becomes an operational dependency.	Relayer censorship, replay risk, weak forwarder validation.
Account abstraction with paymasters	Sponsored interactions, programmable accounts, repeated gasless workflows.	Flexible user experience and policy-based gas sponsorship.	Higher infrastructure and smart contract complexity.	Paymaster abuse, bundler dependency, smart account vulnerabilities.
Hybrid proof anchoring	Systems needing off-chain scalability with on-chain integrity guarantees.	Balances cost, privacy, auditability, and scalability.	Requires reliable off-chain data availability.	Lost proof data, opaque backend interpretation, weak commitment design.

6. Security and Trust Tradeoffs

Although gasless verification enhances usability, it does not eliminate the necessity for thorough security considerations. Sometimes, the use of gasless verification strategies might actually pose an increased threat since it introduces intermediaries or dependencies in the system. A user-friendly approach should not sacrifice the core properties that make the usage of blockchain systems attractive.

In this section, we will consider the key security and trust implications of gasless verification.

6.1. Relayer Centralization

Relayers enhance usability through the submission of transactions on behalf of users. However, relayers can also turn out to be central points of control. Where the system relies on one particular relayer, the relayer will make the decision of whether user requests will be submitted, delayed, rejected or prioritized. It poses availability and censorship risks.

A centralized relayer is even more dangerous when the users do not have an alternative way out. In case the relayer fails, the gasless flow will be disrupted.

Also, in case the operator changes its policy, users might be practically locked out of the system, although the smart contracts are deployed. There will be a distinction between theoretical and practical decentralization.

In order to address such an issue, the system can use multiple relayers, disclose relayer rules, enable direct submission of transactions by the user as an alternative and design the smart contract such that relayers pass requests without making decisions on their validity.

6.2. Replay Attacks

A replay attack happens when a correctly signed message is used outside its intended scope. This is a critical problem for gasless schemes since users approve their actions by signing off-chain messages. If such a signed request does not provide enough information about its context, it can be misused in different chains, contracts, sessions, or during different time intervals.

The following aspects should be considered when designing replay-protection into a gasless scheme. First, nonces make sure that one transaction cannot be executed twice. Second, expirations make a request valid only for a certain period of time. Third, domain separation ties the message to a particular application, chain, contract, or purpose. Fourth, chains' identification helps avoid cross-chain replays. Fifth, proper message schemas allow clearly identify what the user has approved.

Replay protection should be implemented in the system where a request gets validated, and not just in the frontend/backend. Otherwise, an attacker can circumvent it by interacting with smart contracts or relayers. Smart contracts and verification libraries should check the signatures on expiration, duplication, and context.

6.3. Signature Misuse and User Consent

The strength of digital signatures lies in allowing users to approve actions while not sending the transaction themselves. On the other hand, digital signatures can become very dangerous when users do not know what they are signing. An ambiguous or non-transparent signing prompt may result in a user approving actions that imply consequences beyond his or her understanding.

Signed messages in typed structured form increase security in that regard, making the message understandable and relevant to its context. Nevertheless, it does not suffice to make a signed message easily readable. Applications should avoid asking for users' signatures on overbroad, unlimited and opaque authorizations.

Another potential threat lies in signing messages insecurely or using them via untrusted media. In case a signed message is taken from a user by an attacker and is still active, he or she may try to execute the transaction via a relayer or a smart contract. Thus, the importance of expiration time, single-use nonces, secure transport and binding of requests becomes evident.

In case of gasless verification, signatures should be considered security-critical elements. They are not merely some login tokens or interface confirmation prompts. They may possess authorization power and should be treated as such.

6.4. Paymaster Abuse

In account abstraction systems, paymasters are able to sponsor transaction fees for users. It guarantees a great user experience, but at the same time, it opens up a new financial attack vector. An adversary is able to provoke sponsorships of multiple operations, which could drain the paymaster of funds. Even invalid or failed operations could lead to costs if they are not validated properly.

Abuse of paymasters could be minimized by introducing a strict sponsorship policy. Paymaster could define what type of operations are eligible for sponsorship, how frequently the user or address could ask for sponsorship, what contracts could be called, and what max gas cost is acceptable. The system could use rate limiting, allowlisting, risk scoring, deposits, and proof-of-eligibility checks.

However, too strict sponsorship policies could decrease the usability of the system. A system could reject too many requests. Thus, paymaster policy design is a balance between accessibility and cost security. The application needs to understand which types of verifications are worth being paid for by paymasters and which need payment from the user.

Monitoring of the paymaster is also necessary. Operators need to monitor gas consumption, failures, unusual request patterns, and funding level. A gasless system could malfunction not only because of some technical issues, but also because of a lack of funds on the paymaster side.

6.5. Backend Trust Dependency

Most backend-based verification methods use backend services to make life easier for the user. They may be used to construct messages, submit requests to relayers, make queries to the blockchain, cache responses, process proof packages, and deliver results of verification. Although backend services help the user experience, there is a danger of backend becoming a trust dependency.

The backend trust dependency appears when the user cannot verify the result independently. If the backend is the only part of the system that can correctly interpret a proof, then the entire process may become centralized. In such cases, although the blockchain contains commits/events, users have to rely on a backend to interpret them and deliver some meaningful statuses to the user.

In order to avoid this, verification logic should be published, contracts' addresses disclosed, proof data delivered in reproducible formats, and, when possible, independent verification should be done. Also, the frontend should distinguish between cryptographic, blockchain state and application-level verifications. This will let users and auditors know what claims can be proven mathematically and what depends on the system policy.

6.6. Data Availability and Proof Persistence

Proof systems often store extensive evidentiary data off-chain while storing smaller commitments on-chain. This approach is highly efficient; however, it raises data availability issues. In the on-chain hash, one can be sure that there was some data or it matched a specific commitment; however, one cannot be sure whether this data will stay available in the future.

In case the off-chain proof data is lost, deleted, restricted or controlled by a single entity, future verification might be impossible, even if the on-chain commitment exists, but the verifier will not have the necessary proof data to verify the result. It is crucial for systems that need long-term auditability.

Potential solutions include redundant storage, content-addressed storage, open proof format, exportable verification package, public commitment scheme and data retention policy. There should be a defined list of information that should stay available and an organization responsible for its preservation.

Data availability is not just an issue of storage; it is also an issue of trust. A proof system should prevent the scenario in which there is a permanent blockchain record, but a verification process relies on ephemeral private infrastructures.

6.7. Transparency of Trust Assumptions

One of the recurring risks in all gasless systems is the obscuration of trust assumptions by usability benefits. The user would see a simple verification result without understanding whether the verification came directly from blockchain, from a backend cache, from a relayed transaction, from an off-chain signature check or from a paymaster-sponsorship operation.

Such obscuration could be used to mislead regarding the decentralization of the system. In particular, the use of blockchain itself might suggest decentralization, while the actual verification procedure could rely on some centralized components. In such cases, there is no reason to consider the design invalid, yet it must be revealed properly.

A good gasless verification design should explicitly reveal its trust assumptions in a practical manner. The design does not have to reveal all its technical details to the end-user, yet it has to reveal sufficient information in order to let any advanced user (auditor or integrator) replicate the verification procedure. It includes the list of contract addresses, chain identifiers, signature schemes, proof formats, relaying policy, and sources of verification.

The transparency is crucial since gasless verification design modifies who pays, who sends, who interprets and who replicates the procedure.

6.8. Security Design Principles

There are certain universal principles that can help increase the security of gasless verification methods.

First, authorization and transaction creation need to be separated. It means that a relayer or backend can make a request, but cannot create any authority that was not granted by the user or signer. Second, signed messages need to be specific, limited and context-aware. Third, there is a need for a replay protection using nonces, expirations and domain separations. Fourth, a fallback mechanism needs to be provided for users in case of need, e.g., using the public blockchain state for verification. Fifth, sponsorship methods need to have abuse prevention and cost monitoring capabilities. Sixth, the verification process needs to be reproducible whenever it is possible.

These principles will not remove all the risks, but they will guarantee that gasless verification provides better usability without damaging the integrity of decentralized proof methods.

7. Decision Framework for Selecting a Gasless Verification Model

As demonstrated by the preceding sections, gasless verification is not an architecture itself; rather, it comprises multiple design patterns, which differ with regard to their usability, security, costs, and level of decentralization. Thus, the choice of a model should be guided by the specifics of the verification process rather than by the common objective of eliminating gas costs from the user's perspective.

This section offers a practical approach to the selection of a gasless verification model in the context of decentralized proof mechanisms. The suggested approach is based on answering five specific questions, including whether the verification process needs any state changes, the mechanism of establishing the proof authority, the frequency of verification processes, the degree of decentralization required, and the level of technical sophistication of users.

7.1. Determine Whether Verification Requires On-Chain State Changes

The first design choice is whether verification requires changing the blockchain state. If verification only requires checking the state as it exists, then read-only blockchain verification might be enough. This is the simplest and cheapest gasless pattern since no one submits any transactions or pays gas. The application queries the state of the public contract and gets the result.

If the verification requires recording a new event, consuming a proof, changing the status, or authorizations state, then the state-changing transaction is needed. Then the system has to determine who submits and pays for that

transaction. It may be reasonable to use meta-transactions provided by relayers or account abstraction with paymasters if the application wants to sponsor gas to the user.

A frequent design error is when designers write proof verification on-chain even if it is not critical for security. Every verification write increases the cost and complexity, and does not improve the trustworthiness of the system.

Designers have to separate proof validation from audit logging. If audit logging is still required, the system can consider batching, selective anchoring or commitment updates instead of on-chain writing of each proof verification action.

Decision rule: if proof can be verified just by querying existing public state or signature verification, then a state-changing transaction is unnecessary. Otherwise, relayers and paymasters may be used only for such actions.

7.2. Identify the Source of Proof Authority

The second question to consider is the method by which the system verifies a proof's validity. In one class of systems, this validation depends on the blockchain state, while in another, it depends on an authenticated digital signature from an authorized party. In hybrid systems, proof validity may depend on both an authorization statement and an on-chain anchor.

If the proof authority is a signer, then off-chain signature verification may be the most efficient method. The verifier would verify the signature and ensure the signer's authority.

This, however, requires a way to establish the signer's authority reliably. It could be contained in a public registry, associated with an address, anchored on-chain or distributed via a known trust framework.

If the proof authority is a smart contract state, then the verifier might be able to perform read-only verification. The verifier would query the contract and check for the presence of the required state. If the state needs to be changed as part of the verification, then a meta-transaction or a paymaster-sponsorship transaction might be required.

If the proof authority is distributed among multiple entities, hybrid anchoring might be the right solution. The system uses off-chain evidence to establish details of the proof while anchoring commitments to the blockchain.

Decision rule: Use off-chain signature verification if the signer authority is established and proof validity does not involve any additional on-chain execution. Use on-chain verification if the public state is the source of truth. Use hybrid anchoring if the proof integrity depends on blockchain commitments, but its details do not.

7.3. Evaluate Verification Frequency and Cost Sensitivity

The third question is about the frequency of verification. Verification at low frequency allows more expensive and complex workflows. Verification at high frequency needs special care for cost optimization.

High-frequency verification is better served with read-only blockchain requests and off-chain signatures validation since they are free from repeated gas payments. Hybrid anchoring is another model that allows handling a high frequency of verifications since it uses the commitment stored on-chain for verification, and all other checks are made off-chain. This way, the system can scale without having to make every single verification transaction.

Relayer-based meta-transactions and paymaster-funded operations are more expensive because it means that the application will pay for on-chain execution. They should be used only when state changes are needed, or the value of the sponsored operation justifies its cost. In the case of high-frequency systems, sponsorship rules should be strict and predictable.

Cost sensitivity also depends on the blockchain network. What works well on one blockchain network in terms of fees might not work well on another network, especially during a congestion period. Systems should try to avoid designs dependent on a stable price for gas usage without any fee limit, fallbacks or off-chain workarounds.

Decision rule: for high-frequency verification, use read-only, off-chain or hybrid approaches. Be selective with sponsored state-changing transactions.

7.4. Assess Required Decentralization Level

The fourth design question is about how independently verifiable the result should be. In some cases, the application should provide very high levels of decentralization since there should be a possibility for third-party verification of the results without any dependency on the application provider. In other cases, it will be fine to have a more pragmatic approach where blockchain will provide tampering evidence while the application backend will enhance the usability.

If high decentralization is important, the system should have minimal dependency on the backend. It means that contract addresses, chain identifiers, proof format, signature schema, and verification logic should be public. There should be an opportunity to repeat verification results using independent tools and infrastructure. There are read-only blockchain verification, open signature verification, and hybrid anchoring techniques, which are useful in this case.

If intermediate decentralization is accepted, the backend services can help in indexing, caching, proof parsing and usability. However, in this case, the system should clearly

separate the cryptographic facts, blockchain facts, and conclusions of the application itself. For example, the application can state that the signature is correct, there is some commitment on the blockchain, and the application's conclusion is active or expired.

Low decentralization means that the system becomes a conventional backend service with blockchain references. It can be useful, but the level of decentralization in this case should not be overstated.

The fact that the system uses blockchain does not mean that the verification process is decentralized if the user cannot independently verify the result.

Decision rule: if independent verification is critical, publish the verification logic and make minimum use of required intermediaries. If the backend is used, specify which components of the result depend on it.

7.5. Consider User Technical Maturity

The fifth design question is about who the verifier is. A blockchain developer-oriented design is able to show more technical details compared to a user-oriented system. The technical maturity of the user influences wallet prompts, signing prompts, error messages, and blockchain context visualization.

In the case of a non-technical user, there must be no need for wallet prompts, tokens, switching between different networks, or transaction-related terms. Verification output must use simple terms like 'valid', 'invalid', 'expired', 'revoked' or 'cannot be verified'. Technical information can be available through 'advanced view' or 'audit mode'.

For technical users, a system can show contract address, transaction hash, payload, chain IDs, proof packages and raw verification information. This information will allow developers, auditors, and integrators to examine the verification result independently.

Thus, a good system should be able to serve both categories of users through a progressive disclosure approach. User-friendly interface for verification result and technical information that allows reproducing verification output will allow the system to make improvements in usability without concealing trust assumptions.

Decision rule: design a default workflow for the least technical user, but provide enough technical evidence for independent verification.

7.6. Pattern Selection Matrix

A feasible gasless verification architecture can be chosen based on system requirements that match the design pattern.

Read-only blockchain verification is a good solution when the proof state is already present on-chain, there is no need to alter it, and a straightforward result of verification is required. It is a perfect choice for publicly available status checking, registries, authorization state checks, and commitment verification.

Signature-based off-chain proof verification is a proper solution when the authenticity and integrity of the signed message determine the proof validity. It is the best solution when there is clarity regarding the signers' authority, the structure of the message is clear, and replay attacks are prevented by using nonces, expiration time, and domain separation.

Meta-transactions with relayer authorization are a good fit when users authorize a state-changing action, but do not want to make gas payments themselves. This is the proper choice for sponsored workflows, but requires relayer monitoring, fraud prevention, and some fallback strategies.

Paymaster-based account abstraction is the proper choice when the application requires a more programmable way of working with gasless transactions. It is a good fit for frequent actions, smart account capabilities, session authorization, or policy-based sponsoring. However, it is harder to implement due to higher infrastructural complexity and security challenges.

Hybrid proof anchoring on-chain is the optimal choice when there is a need to have proof data off-chain, while on-chain blockchain can provide integrity, timestamping, auditing, or public commitments.

7.7. Recommended Architecture for General-Purpose Proof Systems

In the case of general-purpose decentralized proof systems, the most efficient architecture is a layered hybrid scheme where proofs are created using off-chain signatures, tampering evidence is provided by on-chain commitments, public verification uses read-only blockchain calls, and sponsoring is used only for select state-changing operations.

The described architecture offers several benefits. There are no unnecessary gas fees for standard verification. Users do not have to own native tokens to receive results. It allows maintaining a public point of reference for audits. Advanced users can replicate verification processes. Moreover, such a design minimizes the risks associated with gas sponsorship since the process applies to only selected operations requiring relayers or paymasters.

The general workflow might include the following steps. First, an authorized user creates or signs a proof. Then the system stores or distributes the proof data off-chain. The next step is making a concise on-chain commitment to this proof.

After that, a verifier verifies the proof by checking the signature, comparing the evidence to the on-chain commitment, and reading the contract status. If state changing operation is required, a relayer or paymaster sponsors it according to pre-defined conditions.

Such an architecture does not remove all trust assumptions but makes them easier to control. The application manages to improve user experience without sacrificing cryptographic integrity and public audibility of the process. The key to success is not treating gasless design as a way out. Gasless verification should remain an architectural layer improving accessibility rather than hiding trust management processes.

8. Future Research Directions

Future research on gasless verification should examine several technical directions that are becoming increasingly important for decentralized proof systems.

8.1. Zero-Knowledge Proofs for Privacy-Preserving Verification

Zero-knowledge proof can enhance the decentralized verification process by allowing users to demonstrate particular facts about the underlying data without disclosing the data itself. Future research should focus on the integration of ZKP-based verification in combination with gasless architecture, especially where proof creation occurs off-chain while verification happens via read-only transactions, Layer-2 solutions or sponsored transactions. Further research on proof generation cost, verification cost, developer experience, and user experience across different ZKP implementations is necessary as well [22], [23], [27].

8.2. DID and Verifiable Credential Integration

Decentralized identifiers and verifiable credentials can serve as a basis for structured representation of signer authority, issuer identity, verification method and revocation state. Future research should focus on how DID and VC models can be integrated into gasless verification flow without the necessity to use tokens and perform any transaction [15], [16], [19]–[21], [24].

8.3. Layer-2 and Rollup-Based Verification

Layer-2 systems can potentially lower the cost of verification and make frequent blockchain-verifying transactions more economical. Nevertheless, further research should look into the trust assumptions of Layer-1 and Layer-2 verifications with respect to finality, bridge risks, time until fraud-proofed, validity-proofs verification, sequencer dependence, and data availability. Gasless verification systems implemented on Layer-2 chains need to explicitly clarify which assumptions pertain to the underlying layer and which belong to the Layer-2 system [25], [26], [28].

8.4. Decentralized Relayer and Paymaster Networks

Relayers and paymasters are a good solution for getting rid of direct payments of gas fees; however, they also have the potential to create new points of centralization. Further research needs to examine decentralized relayers, transparent paymasters, shared sponsorships, resilient to abuse paymasters, and alternatives like fallback schemes where users will be able to conduct verification/transaction independently in case the sponsoring infrastructure is not available [5], [8]–[13], [17], [18].

8.5. Empirical Evaluation and User Studies

This review offers a qualitative comparison of the gasless verification systems architectures. Future research should perform empirical evaluation of these gasless verification patterns by assessing the following factors: verification time, number of user actions, transaction fee, failure rate, user comprehension, and infrastructure robustness. In particular, a user study would be valuable for assessing how simplified gasless interfaces contribute to the increase of trust.

9. Conclusion

The necessity for users to pay transaction fees and to interact with blockchain infrastructure is one of the most significant obstacles to the development of blockchain. Within the decentralized proofs, it becomes even more important since users do not need any complex interaction with blockchains. They want to check whether a statement is correct, whether a certain permission exists, a state holds, an event occurred, or any other proof is valid.

This paper explored gasless verification as a way to design decentralized proofs. Five main approaches to achieve the goal have been reviewed: read-only blockchain verification, off-chain signature verification, relayer-based meta-transactions, account abstraction and paymasters, and hybrid on-chain/off-chain proof anchoring. Each of these patterns provides a unique combination of accessibility, cost, decentralization, security, scalability, and implementation complexity.

It is clear that the gasless verification increases the usability and decreases costs, but at the same time, it leads to additional challenges. The usage of relayers makes the process simpler for the user, but increases the risk of centralization of the gateway. An off-chain signature can save transaction fees for the user but requires additional measures for avoiding replay attacks and establishing signer authority. Paymasters can finance the transaction of the user, but increase the risks if the control mechanisms are weak. Hybrid anchoring saves cost and increases scalability, but is based on data availability and a transparent proof format.

One of the key conclusions from the analysis was that the gasless verification must not be estimated only by the question

of whether the user pays for the gas. More important is the question of how the system creates trust. The gasless workflow is useful only if it preserves the integrity, auditability, and reproducibility of verification results. If a gasless design hides the centralized dependencies and trust assumptions, it undermines the essence of the usage of decentralized infrastructure.

The proposed decision-making framework allows choosing the adequate gasless verification design by answering the following questions: does verification require an on-chain state change, how the proof authority is established, how often the verification is needed, what decentralization is needed, and what technical maturity is expected from users. The answers to these questions can help avoid overengineering and choosing the pattern that increases usability and undermines security and transparency.

For many proof systems, the hybrid layered architecture is the best choice. Such architecture can utilize off-chain signatures for proof creation, on-chain commitments for tampering evidence, read-only calls for verification, and sponsorship for state changes only when it is necessary. It decreases user friction and, at the same time, preserves the possibility to verify the evidence independently.

Further investigation should include the development of the standard evaluation methodology for gasless verification systems, better paymaster abuse prevention mechanisms, a decentralized relayer network, reusable proof schemas, and user interface patterns for trust assumptions. As more and more blockchain applications move from the Web3 community toward mainstream usage, gasless verification becomes a more relevant design issue.

References

[1] Satoshi Nakamoto, *Bitcoin: A Peer-to-peer Electronic Cash System*, 2008. [Google Scholar]

[2] Vitalik Buterin, "Ethereum White Paper," *GitHub Repository*, vol. 1, no. 22-23, pp. 5-7, 2013. [Google Scholar]

[3] Gavin Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger (Shanghai Version)," *Ethereum Project Yellow Paper*, vol. 151, pp. 1-32, 2014. [Google Scholar]

[4] Ethereum Improvement Proposals, EIP-712: Typed Structured Data Hashing and Signing, 2017. [Online]. Available: https://eips.ethereum.org/EIPS/eip-712

[5] Ethereum Improvement Proposals, ERC-2771: Secure Protocol for Native Meta Transactions, 2020. [Online]. Available: https://eips.ethereum.org/EIPS/eip-2771

[6] Ethereum Improvement Proposals, ERC-2612: Permit Extension for EIP-20 Signed Approvals, 2020. [Online]. Available: https://eips.ethereum.org/EIPS/eip-2612

[7] Ethereum Improvement Proposals, ERC-3009: Transfer with Authorization, 2020. [Online]. Available: https://eips.ethereum.org/EIPS/eip-3009

[8] Ethereum Improvement Proposals, ERC-4337: Account Abstraction using alt Mempool, 2021. [Online]. Available: https://eips.ethereum.org/EIPS/eip-4337

[9] OpenZeppelin, Sending Gasless Transactions. [Online]. Available: https://docs.openzeppelin.com/contracts/5.x/learn/sending-gasless-transactions

[10] OpenZeppelin, Relaying Gasless Meta-Transactions with a Web App. [Online]. Available: https://docs.openzeppelin.com/defender/guide/meta-tx

[11] OpenGSN, Ethereum Gas Station Network (GSN). [Online]. Available: https://docs.opengsn.org/

[12] ERC-4337 Documentation. [Online]. Available: https://docs.erc4337.io/index.html

[13] Biconomy, Introduction to Biconomy. [Online]. Available: https://docs.biconomy.io/overview/introduction

[14] National Institute of Standards and Technology, FIPS 186-5: Digital Signature Standard, 2023. [Online]. Available: https://csrc.nist.gov/pubs/fips/186-5/final

[15] World Wide Web Consortium, Decentralized Identifiers (DIDs) v1.0: Core Architecture, Data Model, and Representations, 2022. [Online]. Available: https://www.w3.org/TR/did-1.0/

[16] World Wide Web Consortium, Verifiable Credentials Data Model v2.0, W3C Recommendation, 2025. [Online]. Available: https://www.w3.org/TR/vc-data-model-2.0/

[17] Hongxu Su et al., "GasLiteAA: Optimizing ERC-4337 for Efficient and Secure Gas Sponsorship," *2026 IEEE International Conference on Blockchain and Cryptocurrency*, Brisbane, Australia, pp. 1-9, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[18] Huifeng Jiao, and Nathapon Udomlertsakul, "SuperPaymaster: Eliminating Centralized Signer Authority Via Asset-oriented Abstraction to Reconcile Usability and Decentralization in Account Abstraction," *arXiv Preprint*, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[19] Felix Hoops et al., "A Taxonomy of Decentralized Identifier Methods for Practitioners," *2023 IEEE International Conference on Decentralized Applications and Infrastructures*, Athens, Greece, pp. 57-65, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Shrey Jain, Leon Erichsen, and Glen Weyl, "A Plural Decentralized Identity Frontier: Abstraction v. Composability Tradeoffs in Web3," *arXiv preprint*, pp. 1-15, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Carlo Mazzocca et al., "A Survey on Decentralized Identifiers and Verifiable Credentials," *IEEE Communications Surveys and Tutorials*, vol. 27, no. 6, pp. 3641-3671, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Lu Zhou et al., "Leveraging Zero Knowledge Proofs for Blockchain-based Identity Sharing: A Survey of Advancements, Challenges and Opportunities," *Journal of Information Security and Applications*, vol. 80, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[23] Ryan Lavin et al., "A Survey on the Applications of Zero-Knowledge Proofs," *arXiv preprint*, pp. 1-30, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[24] Zhiyue Yan et al., "Blockchain-Driven Decentralized Identity Management: An Interdisciplinary Review and Research Agenda," *Information and Management*, vol. 61, no. 7, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Muhammad Bin Saif, Sara Migliorini, and Fausto Spoto, "A Survey on Data Availability in Layer 2 Blockchain Rollups: Open Challenges and Future Improvements," *Future Internet*, vol. 16, no. 9, pp. 1-17, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Chengpeng Huang et al., "Data Availability and Decentralization: New Techniques for zk-Rollups in Layer 2 Blockchain Networks," *arXiv preprint*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Nojan Sheybani et al., "Zero-Knowledge Proof Frameworks: A Systematic Survey," *arXiv preprint*, pp. 1-21, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[28] Awid Vaziry, Kaustabh Barman, and Patrick Herbke, "SoK: Bridging Trust into the Blockchain. A Systematic Review on On-Chain Identity," *2024 6th Conference on Blockchain Research and Applications for Innovative Networks and Services (BRAINS)*, Berlin, Germany, pp. 1-9, 2024. [CrossRef] [Google Scholar] [Publisher Link]