*electronics*

MDPI

*Article*

# Novel Blockchain and Zero-Knowledge Proof Technology-Driven Car Insurance

**Zhuoliang Qiu [1], Zhijun Xie [1,*], Xianliang Jiang [1], Chuan Ran [1] and Kewei Chen [2]**

[1] Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315000, China; 2111082374@nbu.edu.cn (Z.Q.); jiangxianliang@nbu.edu.cn (X.J.); 2211100275@nbu.edu.cn (C.R.)
[2] Faculty of Mechanical Engineering and Mechanis, Ningbo University, Ningbo 315000, China; chenkewei@nbu.edu.cn
* Correspondence: xiezhijun@nbu.edu.cn

**Abstract:** It is crucial to ensure the privacy and authenticity of the owner's information in car insurance claims. However, the current traditional car insurance claims scenario suffers from inefficiency, complex service, unreliable data, and data leakage. Therefore, considering the privacy and sensitivity of insurance information and car owner data, we can use blockchain, smart contracts, and zero-knowledge proof technology to improve the current problems. This paper proposes a novel car insurance claim scheme based on smart contracts, blockchain, and zero-knowledge proof. Our scheme focuses on preserving privacy in the car insurance authorization and claim process. We design a private smart contract for the creation and revocation of car insurance and public smart contract for the authorization and validation of car insurance. By using ZoKrates, generating zero-knowledge proofs off chain and verifying the proofs on chain reduces the amount of data storage and computation on chain and provides privacy protection for sensitive information. Experimental results confirm the efficacy of our scheme in terms of security and performance.

**Keywords:** car insurance; blockchain; smart contract; zero-knowledge proof

check for updates

**Citation:** Qiu, Z.; Xie, Z.; Jiang, X.; Ran, C.; Chen, K. Novel Blockchain and Zero-Knowledge Proof Technology-Driven Car Insurance. *Electronics* **2023**, *12*, 3869. https://doi.org/10.3390/electronics12183869

Academic Editor: Hamed Taherdoost

Received: 24 August 2023
Revised: 10 September 2023
Accepted: 11 September 2023
Published: 13 September 2023

**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Insurance is one of the most widely used forms of protection worldwide [1]. An insurance policy represents an agreement between individuals or entities and an insurance company, providing financial assistance or reimbursement in the event of a loss. A digital transformation is currently underway in the insurance industry to adapt to the needs of modern society [2]. In the car insurance industry, car insurance claims management is facilitated through the collaboration of various entities from different fields, such as the police, county administrators, insurance agents, and healthcare professionals [3]. This collaborative sharing of multi-source information is crucial for insurance companies to make accurate decisions regarding policyholders' claims.

While insurance plans are prevalent, settling and processing insurance claims can be challenging and error free [4]. Insurance companies often manipulate terms and conditions to avoid paying policyholders, while fraudulent claims can pose problems for insurers [5]. The advantages of blockchain and smart contracts can make insurance contracts more transparent, efficient, and resistant to fraud [6]. Several blockchain-based solutions have been proposed [7,8], with the core idea of using blockchain to establish a trust mechanism between customers and insurance companies, thus effectively confirming the content of car insurance payouts. Automated smart contracts can speed up claims processing and reduce insurers' operating costs.

Indeed, using blockchain-based car insurance plans still presents two significant challenges. Firstly, the identities and insurance details of users participating in the insurance are public, which could lead to privacy breaches and information misuse [9]. Attackers could access all transaction data by downloading a copy of the ledger or trace relationships

between transactions and accounts by analyzing the transaction data in the ledger. Secondly, the car insurance claims process relies on the automatic execution of smart contracts, which may require sensitive information to be received on the blockchain to invoke smart contracts [10], such as the vehicle owner's identity. Since these inputs are publicly transparent, they could expose the vehicle owner's privacy. To address these challenges, further advancements in privacy-preserving techniques and data encryption on the blockchain are necessary.

Zero-knowledge proofs are interactive verification protocols [11]. In this protocol, based on predefined actions, a verifier can be convinced that a prover possesses specific secret data without revealing any private information, including the prover's data, the verifier's identity, and the prover's identity. The verifier only knows that the prover has access to this data. The application of zero-knowledge proofs technology in the blockchain-based insurance sector not only helps to protect the privacy of owners and reduce the risk of information asymmetry but also optimizes the execution process of insurance contracts [12–14]. Our research aims to address the problem of insurance and user information leakage in the blockchain-based car insurance industry by incorporating zero-knowledge proofs.

This paper proposes a solution based on blockchain, smart contracts, and zero-knowledge proofs to address privacy issues in traditional blockchain-based car insurance systems. In this solution, the blockchain ensures the integrity and immutability of insurance data, while smart contracts enable the decentralized execution of insurance claims processes. Additionally, zero-knowledge proofs are used to maintain the privacy of insurance data and user identities.

The contributions of this paper can be summarized as follows:

1. We propose a hybrid smart contract proxy model. Using a private smart contract for creating car insurance protects insurance data from third-party access. A public smart contract is employed for insurance verification, achieving identity authentication without revealing sensitive user information.
2. The utilization of ZoKrates enables zero-knowledge authorization and verification for car insurance. This avoids the exposure of privacy attributes' ownership in a publicly transparent distributed ledger, ensuring non-linkability between vehicle owners and their insurance details.
3. The paper includes a thorough security analysis, demonstrating the privacy and security of our proposed solution. Additionally, comprehensive performance evaluations were conducted to showcase the effectiveness of the proposed approach.

The remaining sections of this paper are organized as follows. In Section 2, we present the background to blockchain-based insurance schemes. In Section 3, we present the preliminary knowledge of the methods used. In Section 4, we propose a model for a vehicle insurance scheme based on blockchain and zero-knowledge proof. In Section 5, we perform a security and performance analysis of the proposed system. Finally, we conclude the paper in Section 6.

## 2. Related Work

The use of blockchain as a system service to design distributed platforms to support the execution of transactions in insurance processes is a core concept to solve the problems of traditional insurance platforms [15]. The insurance industry has adopted blockchain to automate insurance operations by transforming various policies into smart contracts [16].

Many efforts have been made to address car insurance registration using blockchain. Yadav et al. [17] proposed a blockchain-based framework for car insurance to simplify the submission of accident reports and insurance claims. Nizamuddin et al. [18] provided a decentralized blockchain and IPFS-based framework for the auto insurance industry to regulate auto insurance claims and automated payment activities. Lamberti et al. [19] presented a blockchain and sensor-based framework for car insurance that uses smart contracts and sensor data to implement an on-demand insurance system. Bader et al. [20]

proposed a blockchain smart contract-based ecosystem for simple and transparent car insurance, using smart contracts to automate the insurance process. Chiu et al. [21] used blockchain to decentralize data and services and smart contracts as insurance products for insurance companies for bicycle insurance systems. Nanda et al. [22] designed a decentralized system model for the car insurance process based on blockchain technology using Ethereum and smart contracts, with a decentralized application (DApp) for the car insurance purchase and claim process.

Blockchain is also present in other areas of insurance. Kumar et al. [23] proposed a blockchain-based trusted fire brigade service and insurance claim framework to provide immediate fire brigade service to enterprises and prevent insurance fraud while proposing a sensor network and connectivity model to detect real fires and send emergency service requests to monitoring stations. Pawar et al. [24] proposed a blockchain-based insurance system to share health insurance information between hospitals, patients, and insurance companies. Iyer et al. [25] built a decentralized peer-to-peer crop insurance system to cover the risk of excessive rainfall. Jha et al. [26] proposed a blockchain-based crop insurance system to ensure that farmers benefit from the insurance on time.

However, after analyzing existing blockchain-based car insurance schemes, we find that the authors mainly focus on achieving a decentralized insurance system, ignoring the blockchain's information transparency and privacy leakage issues. Therefore, there is a need to expand the scope of policyholder privacy and consider the privacy protection of real and sensitive data during the insurance approval and verification process.

## 3. Preliminary

This section reviews some of the technical preparations required for this paper.

### *3.1. Blockchain and Smart Contract*

The advent of cryptocurrencies has profoundly impacted conventional finance ever since the inception of Bitcoin in 2009 [27]. Positioned as a distributed ledger technology, blockchain facilitates data exchange among designated participants. By aggregating and disseminating transaction data from various data sources, blockchain is structured as a sequence of blocks, each encapsulating the information of multiple transactions interconnected through cryptographic algorithms to form an immutable chain [28]. Diverging from conventional centralized databases, blockchain data are distributed across numerous network nodes, wherein each node possesses a complete ledger copy, necessitating a consensus mechanism for validating and appending new blocks [29]. This decentralized nature endows blockchain with heightened security and resilience against attacks, empowering it to establish a robust trust system within a distributed and untrusted environment.

A smart contract represents an automated contract that utilizes blockchain technology and is expressed as a computer program, facilitating autonomous execution and producing irreversible outcomes [30]. Employing a distributed ledger to store these contracts ensures transactional accuracy without reliance on intermediaries, given the assured reliability of the blockchain [31]. Smart contracts empower users to implement personalized code logic on the blockchain, enabling the establishment of decentralized systems. The key features of smart contracts, including decentralization, autonomy, observability, verifiability, and information sharing, significantly contribute to developing decentralized systems.

### *3.2. Cryptographic Primitives*

#### 3.2.1. Non-Interactive Zero-Knowledge Proof

The fundamental concept of zero-knowledge proofs revolves around proving a statement through interactive protocols [32]. In this process, the prover presents a set of information to the verifier, enabling the verifier to validate the accuracy of this information and gain confidence in its truthfulness without acquiring knowledge of how the prover obtained the information. This information may pertain to the prover's knowledge of the original image of a hash or awareness of the members within a Merkle tree with known

Merkle roots. Practical structures for non-interactive zero-knowledge proof (NIZK) have been demonstrated in Ethernet [33]. The formal definition of NIZK is described below:

- $KeyGen\ (1^{\lambda}) \rightarrow crs$: The input is the safety parameter $\lambda$; the output is the common reference string $crs$.
- $Prove(crs, u, w) \rightarrow \pi$: The inputs are the instance $u$ of some NP-language $L_R$ and the witnesses $w$; the output is a zero-knowledge proof $\pi$.
- $Verify(crs, u, \pi) \rightarrow 1/0$: The input is the proof $\pi$; the output is 1 for acceptance or 0 for rejection.

#### 3.2.2. Fiat–Shamir Heuristic

The Fiat–Shamir heuristic is a technique employed to transform an interactive zero-knowledge proof protocol into a non-interactive version [34]. In conventional interactive zero-knowledge proofs, the prover and verifier engage in multiple rounds of interaction to accomplish the proof process, potentially incurring significant communication costs. Conversely, the Fiat–Shamir heuristic mitigates the communication overhead by converting the interactive protocol into a one-way non-interactive form, achieved through a hash function and a random number generator. The overall process of the Fiat–Shamir heuristic is as follows:

1. The prover runs $Prove(crs, u, w)$ and generates the proof $\pi$. He/she hashes the $(crs, u, \pi)$ to $e$ and sends $\pi$ and $e$ to the verifier.
2. The verifier checks if the equation $e = H(crs, u, \pi)$ holds and runs $Verify(crs, u, \pi)$ to decide whether to accept.

### *3.3. ZoKrates*

ZoKrates [35] is an open-source tool set extensively utilized in the blockchain and cryptocurrency domains for developing and deploying zero-knowledge proofs. It offers a processing model and features a user-friendly domain specific language (DSL), allowing developers to describe intricate computational tasks and generate corresponding zero-knowledge proofs succinctly. These proofs enable the verification of computational results without necessitating an understanding of the specific computations involved. Furthermore, ZoKrates supports diverse zero-knowledge proof systems, including zk-SNARKs (zero-knowledge extensible non-interactive parameters), which streamline the generation and verification of proofs, rendering the process highly efficient and swift. The details of the implementation of the proofs of zero knowledge in ZoKrates are given below:

- Compile: To prove specific computations, circuit designs need to be developed. ZoKrates utilizes a domain specific language (DSL) to describe these circuits. Additionally, ZoKrates provides libraries for commonly used circuits, such as SHA256 and elliptic curve computation.
- Setup: Before generating a proof for each circuit, a one-time setup is required to create a common reference string (CRS).
- Compute witness: ZoKrates automatically computes the corresponding witness based on the circuit when private or public inputs are provided.
- Generate proof: This step involves generating proof information for the given computation.
- Export verifier: ZoKrates allows the exporting of proof-verifier contracts, which can be deployed on Ethereum.

## 4. Proposed System

In this section, we present an overview of our proposed system, which aims to safeguard the privacy of vehicle owners and their car insurance through NIZK and an Ethereum-based distributed ledger. The notations employed in this paper are presented in Table 1.

**Table 1.** Notation setting.

| Notations | Description |
|---|---|
| $C$ | Insurance company |
| $U$ | Vehicle owner |
| $pk_c, pk_u$ | Public keys for insurance company and vehicle owner |
| $sk_c, sk_u$ | Private keys for insurance company and vehicle owner |
| $addr_c, addr_u$ | Blockchain addresses for insurance company and vehicle owner |
| $A$ | Unique asset identifier for insurance |
| $R_A$ | Authorization record for insurance $A$ |
| $\varepsilon$ | Random number |
| $H$ | Cryptographic hash function |

Our new framework focuses on two different scenarios: the car insurance authorization phase and the car insurance claim authentication phase. In the car insurance authorization phase, the insurance company invokes a private smart contract to apply a hash function to the insurance information to generate an asset identifier. Subsequently, using zero-knowledge proof technology, the asset identifier, the owner's public key, and random numbers are hashed to generate an authorization record, and the private authorization of car insurance is achieved by verifying the zero-knowledge proof in the public contract, thus effectively hiding the asset identifier and the owner's information.

Moving on to the car insurance claim authentication phase, the insurance company devises a secret function and transmits it with a proof key to the vehicle owner. The vehicle owner then solves the secret function to generate a witness, which, in conjunction with the proof key, is utilized to produce proof, demonstrating awareness of the secret function as originally drafted by the insurance company. The vehicle owner subsequently interacts with the public smart contract, submitting the generated proof. Once the contract successfully verifies the proof, it can ascertain the legitimacy of the vehicle owner as the rightful policyholder, all without divulging any specific information about the vehicle owner. This process ultimately enables the realization of the insurance claim.

### *4.1. System Overview*

Our system involves six main entities: the blockchain, vehicle owner, insurance company, service providers, smart contracts, and zero-knowledge proof tool. The architecture and workflow of our proposed system, as follows, are shown in Figure 1:

- Blockchain: The blockchain is responsible for deploying carefully designed smart contracts. Our design choice is to reduce computational overhead and avoid using complex cryptographic tools, such as zero-knowledge proofs, on chain.
- Vehicle owner: In the blockchain, the vehicle owner, as a signatory to the insurance policy, owns the identity attributes stored in the blockchain and receives insurance claims by proving ownership of his identity identifier and insurance attributes.
- Insurance company: An insurance company is an organization that provides insurance products and services. Their main responsibility is to issue car insurance policies to vehicle owners, and process claim payments in the event of an accident. By utilizing blockchain technology and smart contracts, insurance companies can create accounts on the blockchain to streamline subsequent insurance operations and improve efficiency and transparency.
- Service providers: Service providers are other entities related to the insurance business, such as vehicle workshops and emergency service providers, responsible for providing specific services to vehicle owners, such as vehicle repairs and emergency assistance. In blockchain, service providers verify the legitimacy of the vehicle owner's identity before providing services.
- Smart contracts: We designed private and public contracts, where private contracts are used to create and revoke car insurance, and public contracts are used for insurance authorization and vehicle owner authentication. We incorporated the zero-knowledge

proof verification contracts in the public contract that enable the vehicle owner to prove the validity of his identity by providing proofs and public parameters as inputs.

- Zero-knowledge proof tool: We use ZoKrates as our tool to implement zero-knowledge proofs. It performs off-chain calculations of zero-knowledge proofs and on-chain verification of their correctness.

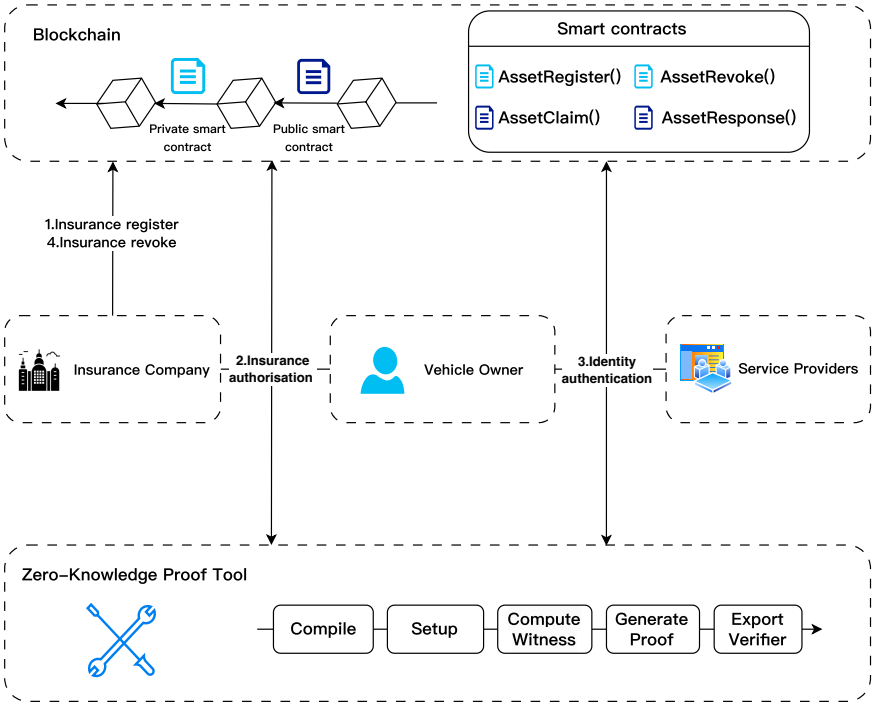


**Figure 1.** Proposed system model.

### *4.2. Insurance Register Phase*

In the proposed system, the insurance company $C$ possesses $pk_c$ and $sk_c$, and the vehicle owner $U$ possesses $pk_u$ and $sk_u$. $C$ must first complete the registration process for the insurance assets by the specific agreement to perform subsequent authorization operations on them. The private smart contract is deployed under $C$'s blockchain address $addr_c$, and only $C$ can invoke it. Algorithm 1 for $AssetRegister()$ is as follows.

**Algorithm 1** AssetRegister

**Require:** $Value, Information$
**Ensure:** $A$
1: $A = sha256(abi.encodePacked(Value, Information, msg.sender, block.number))$;
2: $Insurance[A].value = Value$;
3: $Insurance[A].information = Information$;
4: $Insurance[A].creator = msg.sender$;
5: $Insurance[A].exist = true$;
6: $Insurance[A].claimed = false$;
7: **emit** $LogAssetRegister(A, Value, Information, msg.sender, block.number)$;
8: **return** $A$;

When $C$ calls $AssetRegister()$ in the private smart contract, a unique asset identifier $A$ for insurance is generated. This $A$ is utilized to retrieve and store the mapping from $A$ to the insurance attributes. To prevent the disclosure of their original values after a potential cyber attack, the actual inputs are concatenated with $A$. This ensures that sensitive information remains protected and secure. The generation of $A$ is abstracted into the following equation:

$$A = H(Value, Information, msg.sender, block.number) \quad (1)$$

The *Value* field is utilized to record the insurance's worth. This field is specified by *C* during the insurance registration process, ensuring that relevant financial compensations can be promptly confirmed in the event of accidents or insurance claims. The immutability and transparency of the blockchain guarantee the accuracy and credibility of the *Value* field, eliminating the possibility of human tampering with the value. The $Information$ field is employed to record the specific content and terms of the insurance. During the insurance registration process, the creator provides information regarding the insurance plan, scope of liability, and compensation conditions. These detailed pieces of information are permanently recorded, ensuring the immutability of the insurance contract and mitigating the risks of information loss or tampering. The *msg.sender* field serves to identify the address of the insurance contract creator, also known as $addr_c$. By registering this field in the smart contract, we confirm and record the identity of the insurance creator, which aids in the subsequent authorization process for verifying identities. The *block.number* field is used to increase the security and unpredictability of hash operations, which is included in the hash input data during the transaction preparation stage. This does not change the basic way the blockchain works, but it does allow for a specific hash to be generated based on the block number in which the transaction was created. The reason for using the block number instead of the timestamp is that it is determined by a consensus algorithm on the blockchain network and cannot be changed by miners. In contrast, timestamps are set by the miners and can therefore be artificially manipulated by miners. The certainty of the block number makes it safer to use the block number in a contract.

Simultaneously, we incorporated the *exist* field and the *claim* field in our design to determine the registration status and authorization of an insurance policy within the system. The *exist* field is utilized to ascertain whether an insurance policy has been successfully registered in the system. Upon registration of a vehicle insurance policy, the value of this field is set to *True*, indicating that the insurance policy has been effectively added to the blockchain. Conversely, if the registration is unsuccessful, the value of this field will remain *False*, thereby preventing duplicate or invalid insurance contracts. On the other hand, the *claim* field serves to identify whether the insurance has been authorized by the owner. Once the insurance is active, the owner of the insured vehicle will be authorized in this field and will be entitled to make a claim in case of an accident. By recording the authorization status through the smart contract, we ensure that only the legitimate vehicle owner can receive insurance compensation, thereby enhancing the security and credibility of the insurance system.

### *4.3. Insurance Authorization Phase*

During the insurance authorization stage, *C* can authorize the registered insurance asset *A* to *U* through an anonymous authorization process. This process effectively establishes the mapping between *A* and $pk_u$ on the blockchain while ensuring that this sensitive information remains concealed from public access. By employing this approach, we safeguard the privacy of both *U*'s identity and the specific insurance assets allocated to them, bolstering the overall security and confidentiality of the insurance system. Algorithm 2 for $AssetClaim()$ is as follows.

**Algorithm 2** AssetClaim

**Require:** $A$,$input$,$proof$
**Ensure:** *True* **or** *False*
1: $result = verifyTx(input, proof)$;
2: **require** $(result,$"The proof has not been verified by the contract.");
3: **require** $(creatorQuery(A) == msg.sender,$"You are not the creator of A.");
4: **require** $(claimedQuery(A) == false,$"This A has been claimed.");
5: **require** $(existQuery(A) == true,$This A has been revocationed.);
6: $claims[A] = A$;
7: $Insurance[A].claimed = true$;
8: **emit** $LogAssetClaim(A, msg.sender)$;
9: **return** *True*;

To facilitate the transfer of ownership attributes generated during the registration phase to $U$, we incorporate a hash operation along with the introduction of a random number $\varepsilon$. This approach prevents attackers from cracking hash records by enumerating insurance asset identifier registered on the blockchain and also avoids replay attacks, increasing the security of the system. The process for insurance authorization is as follows:

- Step 1: $C$ formulates a circuit **C** in accordance with the insurance authorization requirements, defining the logic for $A$'s authorization operation within **C**.
- Step 2: $C$ inputs $A$, $pk_u$ and $\varepsilon$, the algorithm computes $R_A = (A, pk_u, \varepsilon)$ within **C**, where $R_A$ and $A$ are set as the public inputs as well as $pk_u$ and $\varepsilon$ are set as the private inputs. Following this, the witness value *witness* is calculated, representing a valid assignment to a variable that encompasses the computation result.
- Step 3: The algorithm generates zero-knowledge proof key pairs $pk$ and $vk$ based on the *witness*, employing a random source commonly referred to as "toxic waste". For generating these zero-knowledge proof key pairs, we employ the efficient Groth16 algorithm, which ensures a balance between the size of the generated proof data and the speed of operation.
- Step 4: $C$ inputs $pk$, $A$, $pk_u$, $\varepsilon$ and $R_A$, the algorithm produces zero-knowledge proof $\pi$.
- Step 5: During the verification phase, the smart contract automatically assesses the correctness of the provided inputs. The zero-knowledge proof $\pi$ undergoes verification using $vk$. The insurance policy is deemed authorized to the owner only if the above validation holds true. This process guarantees secure and accurate authorization of insurance ownership while preserving privacy and confidentiality.

The validation contract is generated and deployed on the public smart contract via ZoKrates and is named $AssetClaim()$. After passing zero-knowledge verification, it is also necessary to determine whether the account address invoking the contract is the same as the one used to register $A$ and whether $A$ has been registered and authorized. Once all the above operations have been passed, $R_A$ is recorded in the authorization record, and the *claim* of $A$ is changed to *True*.

In this way, we can effectively authorize insurance to $U$ while concealing the ownership relationship through zk-SNARK, safeguarding both $U$'s privacy and the security of the insurance assets.

### *4.4. Identity Authentication Phase*

During an insurance claim for a vehicle involved in an accident, the vehicle owner must provide sufficient information to establish their legal ownership of the insurance. However, relying on the traditional blockchain-based vehicle insurance claim process may expose the owner's information through the input of smart contracts, posing a risk of privacy leakage. To address this problem, we implement owner authentication with privacy-preserving features based on the Fiat–Shamir heuristic. Algorithm 3 for $AssetResponse()$ is as follows.

**Algorithm 3** AssetResponse

**Require:** $A$,$input'$,$proof'$
**Ensure:** *True* **or** *False*
1: $result = verifyRes(input', proof')$;
2: **require** $(result,$"The proof has not been verified by the contract."$)$;
3: **require** $(existQuery(A) == true,$This A has been revocationed.$)$;
4: **require** $(calims(A) == A,$This A has not been claimed.$)$;
5: **return** *True*;

To authenticate as the generator of the insurance record $R_A$ on the blockchain, $U$ needs to provide a zero-knowledge proof to demonstrate the truth of the following two statements:

1. $R_A$ can be recomputed: First, $U$ has the $sk_u$, which allows the generation of $pk_u$ by a hash function. Second, by combining $pk_u$ with the unique hash value of the insurance $A$ and a random number $\varepsilon$, the insurance record $R_A$ can be recalculated.
2. The recomputed $R_A$ is saved on the blockchain.

The first point is essentially proof of the existence of a specific computational process, demonstrating that $U$ possesses the necessary information to generate the insurance record. The second point is essentially proof of the existence of a specific element in a set, which, in this case, is the insurance record $R_A$ saved on the blockchain. However, the proof must maintain the confidentiality of information regarding the specific element being referred to, ensuring that sensitive details about $R_A$ are not disclosed. In the above proof process, both $\varepsilon$ and $U$'s identity ($pk_u$) are kept confidential, ensuring privacy protection. The authentication process proceeds as follows:

- Step 1: A new circuit $\mathbf{C}'$ is designed, the logic of which is for $U$ to prove to the service provider that he/she is the rightful owner of $R_A$.
- Step 2: $U$ inputs $sk_u$, $A$, $pk_u$ and $\varepsilon$, the algorithm computes $pk'_u = H(sk_u)$ and $R'_A = H(A, pk_u, \varepsilon)$, where $pk'_u$, $R'_A$ and $A$ are set as the public inputs as well as $sk_u$, $pk_u$ and $\varepsilon$ being set as the private inputs. Following this, the witness value $witness'$ is calculated, representing a valid assignment to a variable that encompasses the computation result.
- Step 3: The algorithm generates zero-knowledge proof key pairs $pk'$ and $vk'$ based on $witness'$.
- Step 4: $U$ inputs $pk'$, $sk_u$, $A$, $pk_u$, $\varepsilon$, $pk'_u$ and $R_A$, and the algorithm produces zero-knowledge proof $\pi'$.
- Step 5: During the verification phase, the smart contract automatically assesses the correctness of the provided inputs. The zero-knowledge proof $\pi'$ undergoes verification using $vk'$. And the algorithm compare whether $pk'_u$ is the same as $pk_u$ recorded in $R_A$ and whether $R'_A$ is the same as $R_A$. If all the above proofs are valid, the algorithm returns *True*.

The validation contract is generated and deployed on the public smart contract via ZoKrates and is named $AssetResponse()$. After passing the zero-knowledge verification, it is also necessary to determine whether $R_A$ is the same as the one recorded on the blockchain and whether $A$ is registered. Once all of these operations have been passed, $U$ is determined to be the legal owner of $R_A$ without revealing any identifying information about the vehicle owner in the process. The insurance claim process can then be carried out.

### *4.5. Insurance Revoke Phase*

To revoke $A$, the $AssetRevoke()$ function is called by $C$ within the private smart contract. It is important to enforce that the account address initiating the revoke operation matches the account address used to register $A$. Once this condition is satisfied, the smart contract updates the *exist* status of $A$ to *False*, effectively stopping any subsequent calls to $AssetClaim()$ and $AssetResponse()$. In doing so, the smart contract ensures that the

entitlement of $A$ is revoked and prevents any further interaction with it. Algorithm 4 for $AssetRevoke()$ is as follows.

**Algorithm 4** AssetRevoke

**Require:** $A$
**Ensure:** *True* **or** *False*
1: $creator = creatorQuery(A)$;
2: **if** $msg.sender == creator$ **then**
3: $Insurance[A].claimed = true$;
4: **emit** $LogAssetRevocation(A)$;
5: **return** *True*;
6: **end if**
7: **return** *False*;

The process for asset revoke is as follows:

- Step 1: The algorithm determines whether the address of the account that initiated the undo operation is the address of the account that created $A$.
- Step 2: The algorithm sets the *exist* field of $A$ to *False*.

## 5. Analysis of System

In this section, we analyzed the proposed system in various ways.

### *5.1. Privacy and Security Analysis*

- Security of zero Knowledge: ZoKrates offers several alternative zero-knowledge proof schemes, among which Groth16 [36] is a typical and proven secure scheme.
- Unlinkability of identity: The insurance data are stored on the private smart contract, which remains inaccessible to anyone except the insurance company. The authorization process for insurance is implemented through zero-knowledge proofs. To attempt to reveal the owner's private information through ZoKrates, an attacker would need to perform a brute-force attack on the private token within the hash statement. However, given the current computing power, calculating $2^{256}$ hashes is practically impossible.
- Prevention of replay attack: By adding additional data $\varepsilon$ to the computation and incorporating it into the hash calculation, the result of each computation becomes unique even if the same $A$ and $pk_u$ are used. This prevents replay attacks because $\varepsilon$ is different each time, making it impossible for an attacker to reuse previous proofs.
- Security of data transmission: All private data transmission is secured through digital signatures and hash encryption. Vehicle owners, insurance companies, and service providers can verify each other's communications through digital signatures. Ensuring the security of the certificate authority that issues the digital signatures and symmetric keys prevents attackers from executing man-in-the-middle attacks by eavesdropping on messages.

### *5.2. Efficiency Analysis*

The performance evaluation of blockchain primarily encompasses two crucial metrics: transaction throughput and latency. Transaction throughput refers to the number of transactions processed within a specific time frame, while latency signifies the response and processing time of transactions. Low throughput may be influenced by factors such as block capacity limitations. Latency is closely associated with algorithm efficiency, and while network bandwidth constraints can also impact latency, its core reasons lie in algorithmic effectiveness rather than inherent issues of the blockchain itself. To enhance throughput, increasing block generation speed is one approach, but this could lead to blockchain forks, thereby compromising system security. To achieve increased block throughput without compromising system security, zero-knowledge proofs present an optimal solution. Exceptional zero-knowledge proof algorithms can significantly minimize latency while

simultaneously ensuring the integrity and correctness of remote computation processes without divulging any private information.

Our approach employs the Groth16 algorithm from zk-SNARK to achieve privacy protection. This algorithm relies on the security of solving the elliptic curve discrete logarithm problem. We compared Groth16 with several other common zk-SNARK solutions. As there is no unified benchmark for each construction, we analyzed them based on proof size, benchmark metrics from the respective papers, or estimates from data provided by the inventors. Partala et al. [37] made statistics, and the comparative results are presented in Table 2. In the table, **C** denotes the circuit, **|C|** represents the number of gates in **C**, and **N** indicates the length of computed inputs and outputs. It is evident from the table that each solution has notable strengths and weaknesses, but Groth16 still stands out in terms of proof data size and speed.

**Table 2.** Time complexity of different algorithms.

| | **Compiling** | **Sizes** | **Prover** | **Verifier** |
|---|---|---|---|---|
| Groth16 | $O(\|C\|^2)$ | $O(1)$ | $O(\|C\|^2)$ | $O(\|C\|)$ |
| Stark | *No* | $O(log^2\|C\|)$ | $O(\|C\|log^2\|C\|)$ | $O(\|C\|)$ |
| Aurora | *No* | $O(log^2\|C\|)$ | $O(\|C\|log\|C\|)$ | $O(\|C\|)$ |
| Marlin | $O(\|C\|log\|C\|)$ | $O(\|C\|)$ | $O(\|C\|log\|C\|)$ | $O(N+log\|C\|)$ |
| Sonic | $O(\|C\|log\|C\|)$ | $O(1)$ | $O(\|C\|log\|C\|)$ | $O(N+log\|C\|)$ |
| SuperSonic | $O(\|C\|log\|C\|)$ | $O(log\|C\|)$ | $O(\|C\|log\|C\|)$ | $O(log\|C\|)$ |

### *5.3. Performance Analysis*

To accurately assess the feasibility of the solution, we tested the number of constraints and proof sizes for zero-knowledge proofs, the time consumption for zero-knowledge proofs, the gas consumption caused by smart contract operations and zero-knowledge proof operations, and compared them with other work.

We chose Ethereum as the smart contract platform and used Solidity0.8.0 [38] for smart contract development. The experiments are based on Remix0.34.1, which supports the testing, debugging, and deploying of smart contracts on Ethereum. The consensus algorithm implemented is PoS [39]. In addition, we utilized Web3.js1.10.0 to interact with Ethereum nodes. To simulate the Ethereum network environment, we used Ganache7.9.0 [40] as a personal blockchain for Ethereum development and created a test system using Truffle5.11.2 [41], the most popular development framework for Ethereum. We deployed smart contracts on Truffle and used the Truffle console to simulate data and test smart contracts. Ethereum is the most reliable and widely available blockchain and can develop and execute advanced and customized smart contracts using the Solidity programming language. All zero-knowledge proof operations were implemented on ZoKrates0.8.4.

#### 5.3.1. Number of Constraints and Key Size

In the setup phase of the algorithm, the number of computational constraints and key results obtained by compiling two specific computations in ZoKrates are shown in Table 3. The more computational constraints that are generated, the more complex the specific computations become, resulting in larger key sizes.

**Table 3.** Results of particular computation pairs.

| | **Constraints** | **Proving Key (Mbytes)** | **Verification Key (bytes)** |
|---|---|---|---|
| AssetClaim | 104,486 | 41.6 | 2000 |
| AssetResponse | 131,042 | 50.1 | 3000 |

5.3.2. Time Cost

In the local client, we conducted performance evaluations by generating witnesses and proofs for two specific computations, and the recorded times are presented in Figure 2. Each result in the figure represents the average of 100 test runs, ensuring the accuracy and reliability of the measurements. With this configuration, the time taken to generate the proofs is deemed acceptable, while the time required for generating zk-SNARK proofs depends on various factors, including the computational resources allocated by the prover, the logic of the code, and the complexity of the computation.

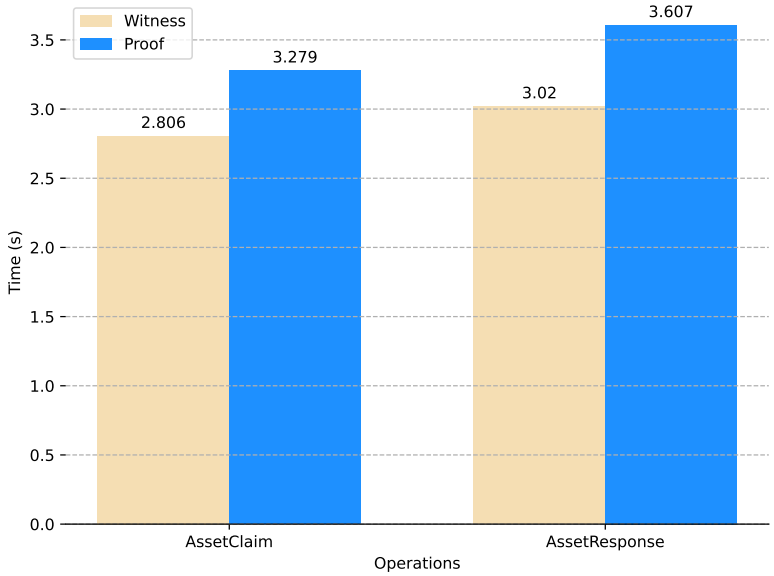


**Figure 2.** Average latency of witness and proof.

Meanwhile, we also tested the time consumption of key function operations in public and private smart contracts, and the results are shown in Figure 3. Each result in the figure represents the average of 100 test runs, ensuring the accuracy and reliability of the measurements. *AssetRevoke*() is the least time consuming, as this function only contains one compare and one change operation. *AssetResponse*() is the most time consuming because of this function's complex zero-knowledge proof operation.

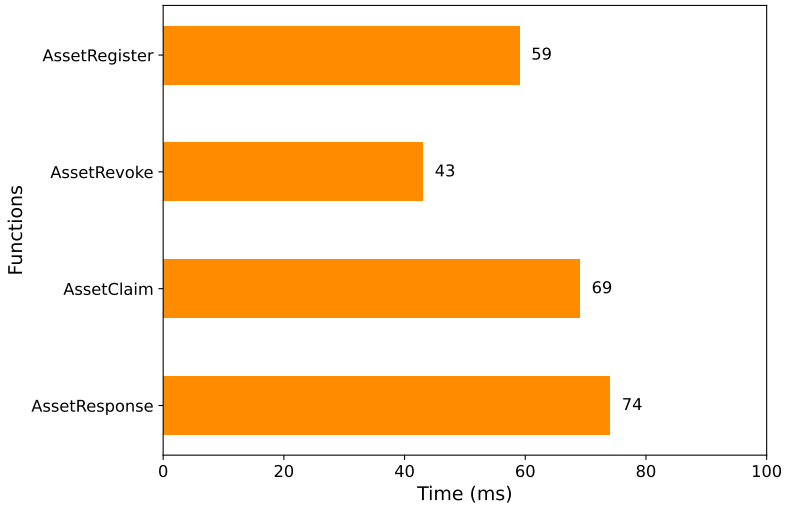


**Figure 3.** Average latency of four smart contract functions.

We further conducted experiments to evaluate the time consumption of implementing the *AssetClaim*() method using three distinct zk-SNARK algorithms within the ZoKrates framework as depicted in Figure 4. Groth16 is the fastest in compilation settings, witness computation, and proof generation.

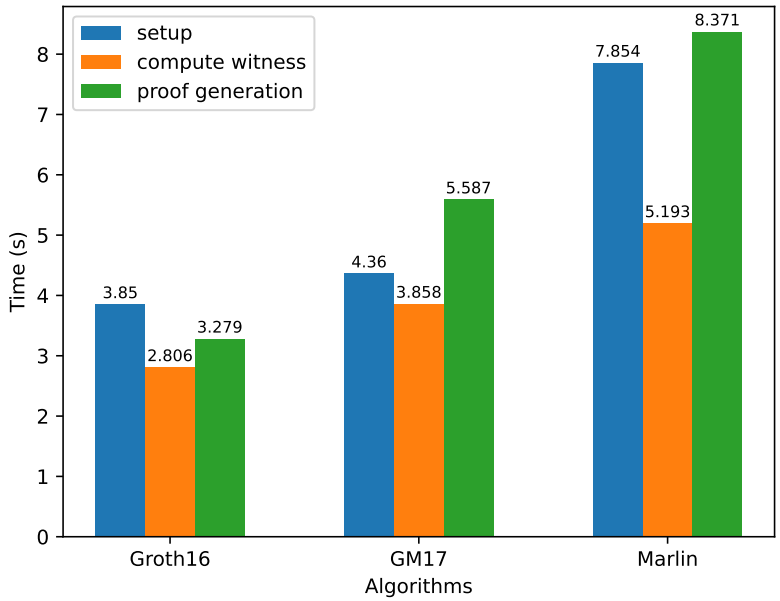


**Figure 4.** Average latency of three zk-SNARK algorithms.

### 5.3.3. Gas Consumption

In our proposed scheme, various operations, such as insurance creation, insurance revocation, insurance privacy authorization, and verification of the owner's identity, require transactions to be sent to the smart contract for execution. Insurance authorization and identity verification also necessitate submitting public inputs and zero-knowledge proofs. It is important to note that invoking smart contracts on the blockchain incurs a significant amount of gas consumption as depicted in Table 4. We set the gas price to the average of 20 Gwei (0.00000002 Eth). Gas consumption costs vary depending on the specific smart contract operations being performed. As can be seen from the table, the gas consumption and ether price for each functional operation are perfectly acceptable.

**Table 4.** Transaction fee statistics.

| Contract Operations | $\mathbf{Gas_{used}}$ | $\mathbf{Fee_{eth}}$ |
|---|---|---|
| AssetRegister | 98,210 | 0.00196420 |
| AssetRevoke | 23,071 | 0.00046142 |
| AssetClaim | 319,284 | 0.00638568 |
| AssetResponse | 298,255 | 0.00596510 |

### 5.3.4. Characteristic Comparison

In this section, we compare our proposed scheme and other similar insurance schemes in Table 5. Our scheme stands out by meeting the requirement for legal car insurance claims while ensuring the authenticity and privacy of the insurance authorization and the owner authentication processes, a combination not fully achieved by other related schemes. Y means yes, and N means not available.

**Table 5.** Comparison with other related schemes.

| | Demir [42] | Roriz [43] | Liu [44] | Bhadra [45] | Our Scheme |
|---|---|---|---|---|---|
| Blockchain based | Y | Y | Y | Y | Y |
| Identity protection | Y | N | Y | Y | Y |
| Privacy authorization | N | N | N | N | Y |
| Data authentication | N | Y | Y | Y | Y |

We also performed a performance comparison of similar solution [46], and the results are shown in Figure 5. We merged the *AssetRegister* function and the *AssetClaim* function into the *CreateInsurance* function. Except for the higher gas consumption of the revoke insurance operation, the scheme proposed in this paper outperforms other schemes in the rest of the metrics because it improves the algorithmic process of policy creation and claim verification by smart contracts.

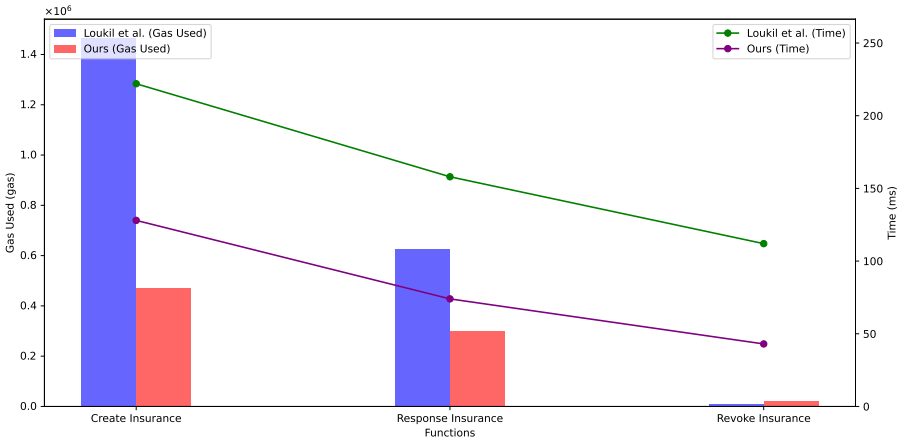


**Figure 5.** Comparison of time and gas used with similar solution [46].

## 6. Conclusions

This paper proposes a decentralized zero-knowledge proof-based car insurance claim framework to address the privacy leakage problem in car insurance schemes under the traditional blockchain framework. Currently, in the popular blockchain car insurance schemes, the contents of the insurance, ownership, and transfer records are fully public to all nodes in the chain. During identity verification at the claim stage, the vehicle owner must enter private information into the smart contract to verify the legitimacy of their identity, and this process also carries the risk of privacy leakage. Our goal is to achieve secret authorization during the insurance authorization process and secret verification of the vehicle owner's identity at the claim stage. Compared to traditional car insurance schemes in the blockchain framework, our proposed scheme achieves privacy protection by adding zero-knowledge proof technology on top of decentralization. We design both private and public smart contracts, where insurance authorization and identity verification processes are implemented on public smart contracts. Proofs are optimized using ZoKrates to reduce the size of the proof, which reduces on-chain overhead and provides privacy features. Experimental results show that the scheme performs well in terms of security and performance. A comprehensive comparative analysis with other schemes proves that our scheme achieves both secret authorization and privacy protection.

Our proposed solution increases standardization and reliability within the processes of the car insurance industry. However, the scalability of blockchain technology and the efficiency of zero-knowledge proof algorithms may present new challenges. For the future work, we will further optimize the performance of the zero-knowledge proof algorithm and focus on implementing the model in an automated manner.

**Author Contributions:** Conceptualization, Z.Q. and Z.X.; methodology, Z.Q. and X.J.; software, Z.Q.; validation, Z.Q., Z.X. and X.J.; formal analysis, Z.Q.; investigation, Z.Q. and Z.X.; resources, Z.Q.; data curation, Z.Q.; writing—original draft preparation, Z.Q.; writing—review and editing, Z.Q. and Z.X.; visualization, Z.Q. and C.R.; supervision, Z.Q. and Z.X.; project administration, Z.X.; funding acquisition, Z.X. and K.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation of China (Grant No. U20A20121); Ningbo public welfare project (Grant No. 202002N3109, 2022S094); Natural Science Foundation of Zhejiang Province (Grant No. LY21F020006); The international cooperation project of Ningbo (Grant No. 2023H012, 2023H007); Science and Technology Innovation 2025 Major Project of Ningbo (Grant No. 2019B10125, 2019B10028, 2020Z016, 2021Z031, 2022Z074, 2022Z241, 2023Z132, 2023Z133, 2023Z216); Ningbo Fenghua District industrial chain key core technology "unveiled the commander" project (Grant No. 202106206).

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Acknowledgments:** The authors would like to acknowledge Ningbo University for its lab facilities and necessary technical support.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ewold, F. Insurance and risk. *Foucault Eff. Stud. Gov.* **1991**, *197210*, 201–202.
2. Satuluri, R.K. Digital transformation in Indian insurance industry. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2021**, *12*, 310–324.
3. Catlin, T.; Lorenz, J.T.; Nandan, J.; Sharma, S.; Waschto, A. *Insurance beyond Digital: The Rise of Ecosystems and Platforms*; McKinsey & Company: New York, NY, USA, 2018.
4. Huang, C.; Wang, W.; Liu, D.; Lu, R.; Shen, X. Blockchain-assisted personalized car insurance with privacy preservation and fraud resistance. *IEEE Trans. Veh. Technol.* **2022**, *72*, 3777–3792. [CrossRef]
5. Derrig, R.A. Insurance fraud. *J. Risk Insur.* **2002**, *69*, 271–287. [CrossRef]
6. Gatteschi, V.; Lamberti, F.; Demartini, C.; Pranteda, C.; Santamaría, V. Blockchain and smart contracts for insurance: Is the technology mature enough? *Future Internet* **2018**, *10*, 20. [CrossRef]
7. Wang, K.; Safavi, A. *Blockchain is Empowering the Future of Insurance*; TechCrunch AOL Inc.: San Francisco, CA, USA, 2016; Volume 7.
8. Bhamidipati, N.R.; Vakkavanthula, V.; Stafford, G.; Dahir, M.; Neupane, R.; Bonnah, E.; Wang, S.; Murthy, J.; Hoque, K.A.; Calyam, P. Claimchain: Secure blockchain platform for handling insurance claims processing. In Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, 6–8 December 2021; pp. 55–64.
9. Qi, H.; Wan, Z.; Guan, Z.; Cheng, X. Scalable decentralized privacy-preserving usage-based insurance for vehicles. *IEEE Internet Things J.* **2020**, *8*, 4472–4484. [CrossRef]
10. Khan, S.N.; Loukil, F.; Ghedira-Guegan, C.; Benkhelifa, E.; Bani-Hani, A. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-To-Peer Netw. Appl.* **2021**, *14*, 2901–2925. [CrossRef] [PubMed]
11. Fiege, U.; Fiat, A.; Shamir, A. Zero knowledge proofs of identity. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 25–27 May 1987; pp. 210–217.
12. Sharma, B.; Halder, R.; Singh, J. Blockchain-based interoperable healthcare using zero-knowledge proofs and proxy re-encryption. In Proceedings of the 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 7–11 January 2020; pp. 1–6.
13. Wan, Z.; Guan, Z.; Zhou, Y.; Ren, K. Zk-AuthFeed: How to feed authenticated data into smart contract with zero knowledge. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 83–90.
14. Wan, Z.; Zhou, Y.; Ren, K. Zk-AuthFeed: Protecting data feed to smart contracts with authenticated zero knowledge proof. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 1335–1347. [CrossRef]
15. Raikwar, M.; Mazumdar, S.; Ruj, S.; Gupta, S.S.; Chattopadhyay, A.; Lam, K.Y. A blockchain framework for insurance processes. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; pp. 1–4.
16. Brophy, R. Blockchain and insurance: A review for operations and regulation. *J. Financ. Regul. Compliance* **2020**, *28*, 215–234. [CrossRef]
17. Yadav, A.S.; Charles, V.; Pandey, D.K.; Gupta, S.; Gherman, T.; Kushwaha, D.S. Blockchain-based secure privacy-preserving vehicle accident and insurance registration. *Expert Syst. Appl.* **2023**, *230*, 120651. [CrossRef]
18. Nizamuddin, N.; Abugabah, A. Blockchain for automotive: An insight towards the IPFS blockchain-based auto insurance sector. *Int. J. Electr. Comput. Eng. (IJECE)* **2021**, *11*, 2443–2456. [CrossRef]

19. Lamberti, F.; Gatteschi, V.; Demartini, C.; Pelissier, M.; Gomez, A.; Santamaria, V. Blockchains can work for car insurance: Using smart contracts and sensors to provide on-demand coverage. *IEEE Consum. Electron. Mag.* **2018**, *7*, 72–81. [CrossRef]
20. Bader, L.; Bürger, J.C.; Matzutt, R.; Wehrle, K. Smart contract-based car insurance policies. In Proceedings of the 2018 IEEE Globecom Workshops (GC wkshps), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–7.
21. Chiu, W.Y.; Meng, W. Towards decentralized bicycle insurance system based on blockchain. In Proceedings of the 36th Annual ACM Symposium on Applied Computing, Virtual, 22–26 March 2021; pp. 249–256.
22. Nanda, S.K.; Panda, S.K.; Das, M.; Satapathy, S.C. Automating vehicle insurance process using smart contract and Ethereum. In Proceedings of the Advances in Micro-Electronics, Embedded Systems and IoT: Proceedings of Sixth International Conference on Microelectronics, Electromagnetics and Telecommunications (ICMEET 2021), Bhubaneswar, India, 27–28 August 2021; Springer: Berlin/Heidelberg, Germany, 2022; pp. 237–247.
23. Kumar, S.; Dohare, U.; Kaiwartya, O. FLAME: Trusted fire brigade service and insurance claim system using blockchain for enterprises. *IEEE Trans. Ind. Inform.* **2022**, *19*, 7517–7527.
24. Pawar, V.; Sachdeva, S. ParallelChain: A scalable healthcare framework with low-energy consumption using blockchain. *Int. Trans. Oper. Res.* **2023**. [CrossRef]
25. Iyer, V.; Shah, K.; Rane, S.; Shankarmani, R. Decentralised Peer-to-Peer Crop Insurance. In Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure, Virtual, 7 June 2021; pp. 3–12.
26. Jha, N.; Prashar, D.; Khalaf, O.I.; Alotaibi, Y.; Alsufyani, A.; Alghamdi, S. Blockchain based crop insurance: A decentralized insurance system for modernization of Indian farmers. *Sustainability* **2021**, *13*, 8921. [CrossRef]
27. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, *13*, 21260.
28. Nofer, M.; Gomber, P.; Hinz, O.; Schiereck, D. Blockchain. *Bus. Inf. Syst. Eng.* **2017**, *59*, 183–187. [CrossRef]
29. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
30. Buterin, V. A next-generation smart contract and decentralized application platform. *White Pap.* **2014**, *3*, 1–36.
31. Hewa, T.; Ylianttila, M.; Liyanage, M. Survey on blockchain based smart contracts: Applications, opportunities and challenges. *J. Netw. Comput. Appl.* **2021**, *177*, 102857. [CrossRef]
32. Goldwasser, S.; Micali, S.; Rackoff, C. The knowledge complexity of interactive proof-systems. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*; ACM: New York, NY, USA, 2019; pp. 203–225.
33. Gennaro, R.; Gentry, C.; Parno, B.; Raykova, M. Quadratic span programs and succinct NIZKs without PCPs. In Proceedings of the Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, 26–30 May 2013; Proceedings 32; Springer: Berlin/Heidelberg, Germany, 2013; pp. 626–645.
34. Canetti, R.; Chen, Y.; Holmgren, J.; Lombardi, A.; Rothblum, G.N.; Rothblum, R.D.; Wichs, D. Fiat-Shamir: From practice to theory. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, Phoenix, AZ, USA, 23–26 June 2019; pp. 1082–1090.
35. Eberhardt, J.; Tai, S. Zokrates-scalable privacy-preserving off-chain computations. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1084–1091.
36. Baghery, K.; Pindado, Z.; Ràfols, C. Simulation extractable versions of Groth's zk-SNARK revisited. In Proceedings of the International Conference on Cryptology and Network Security, Vienna, Austria, 14–16 December 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 453–461.
37. Partala, J.; Nguyen, T.H.; Pirttikangas, S. Non-interactive zero-knowledge for blockchain: A survey. *IEEE Access* **2020**, *8*, 227945–227961. [CrossRef]
38. Dannen, C. *Introducing Ethereum and Solidity*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 1.
39. Bentov, I.; Lee, C.; Mizrahi, A.; Rosenfeld, M. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *Acm Sigmetrics Perform. Eval. Rev.* **2014**, *42*, 34–37. [CrossRef]
40. Lee, W.M.; Lee, W.M. Testing smart contracts using ganache. In *Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity, and JavaScript*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 147–167.
41. Mohanty, D.; Mohanty, D. Frameworks: Truffle and embark. In *Ethereum for Architects and Developers: With Case Studies and Code Samples in Solidity*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 181–195.
42. Demir, M.; Turetken, O.; Ferworn, A. Blockchain based transparent vehicle insurance management. In Proceedings of the 2019 Sixth International Conference on Software Defined Systems (SDS), Rome, Italy, 10–13 June 2019; pp. 213–220.
43. Roriz, R.; Pereira, J.L. Avoiding insurance fraud: A blockchain-based solution for the vehicle sector. *Procedia Comput. Sci.* **2019**, *164*, 211–218. [CrossRef]
44. Liu, X.; Yang, H.; Li, G.; Dong, H.; Wang, Z. A blockchain-based auto insurance data sharing scheme. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 3707906. [CrossRef]

45. Bhadra, O.; Sahoo, S.; Kumar, C.M.; Halder, R. Decentralized Insurance Subrogation Using Blockchain. In Proceedings of the 2022 5th International Conference on Blockchain Technology and Applications, Xi'an, China, 16–18 December 2022; pp. 1–9.
46. Loukil, F.; Boukadi, K.; Hussain, R.; Abed, M. Ciosy: A collaborative blockchain-based insurance system. *Electronics* **2021**, *10*, 1343. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.