future internet

Article

NDN-BDA: A Blockchain-Based Decentralized Data Authentication Mechanism for Vehicular Named Data Networking

Ahmed Benmoussa [1,*], Chaker Abdelaziz Kerrache [2], Carlos T. Calafate [3,*] and Nasreddine Lagraa [2]

[1] Department of Computer Science, University Center of Aflou, Laghouat 03000, Algeria
[2] Laboratoire d'Informatique et de Mathématiques, Université Amar Telidji de Laghouat, Laghouat 03000, Algeria; ch.kerrache@lagh-univ.dz (C.A.K.); n.lagraa@lagh-univ.dz (N.L.)
[3] Computer Engineering Department (DISCA), Universitat Politècnica de València, 46022 Valencia, Spain
* Correspondence: a.benmoussa@cu-aflou.edu.dz (A.B.); calafate@disca.upv.es (C.T.C.)

Abstract: Named Data Networking (NDN) is an implementation of Information-Centric Networking (ICN) that has emerged as a promising candidate for the Future Internet Architecture (FIA). In contrast to traditional networking protocols, NDN's focus is on content, rather than the source of the content. NDN enables name-based routing and location-independent data retrieval, which gives NDN the ability to support the highly dynamic nature of mobile networks. Among other important features, NDN integrates security mechanisms and prioritizes protecting content over communication channels through cryptographic signatures. However, the data verification process that NDN employs may cause significant delays, especially in mobile networks and vehicular networks. This aspect makes it unsuitable for time-critical and sensitive applications such as the sharing of safety messages. Therefore, in this work, we propose NDN-BDA, a blockchain-based decentralized mechanism that provides a faster and more efficient data authenticity mechanism for NDN-based vehicular networks.

Keywords: blockchain; data authentication; Named Data Networking (NDN); Future Internet Architecture (FIA); vehicular networks

Citation: Benmoussa, A.; Kerrache, C.A.; Calafate, C.T.; Lagraa, N. NDN-BDA: A Blockchain-Based Decentralized Data Authentication Mechanism for Vehicular Named Data Networking. *Future Internet* **2023**, *15*, 167. https://doi.org/10.3390/fi15050167

Academic Editor: Paolo Bellavista

Received: 31 March 2023
Revised: 21 April 2023
Accepted: 28 April 2023
Published: 29 April 2023

1. Introduction

Vehicular networks encounter many challenges, such as high mobility, heterogeneous communication, and low latency, which need to be addressed to ensure long-distance and reliable connections. However, the TCP/IP model that the actual Internet relies on was not designed for such networks as it does not adequately address these prerequisites. Furthermore, the current Internet architecture was not designed to handle the massive number of connected devices, which is expected to reach almost 30 billion in 2023 [1]. Additionally, people's use of the Internet has changed, with their focus shifting from the origin of the content to the content they want to access. To tackle these limitations, the research community has explored alternatives to support these requirements. One such alternative is Information-Centric Networking (ICN) [2], which is considered one of the most-attractive Future Internet Architectures (FIAs) [3]. Among the different ICN architectures proposed, such as DONA [4] and PUSUIT [5], Named Data Networking (NDN) [6] seems to be the most-promising contender for the upcoming Internet architecture. NDN finds its roots in the Content-Centric Networking (CCN) project [7], which emphasizes a content-oriented communication approach and prioritizes the data over their owner. Instead of relying on IP addresses, NDN uses data names for packet forwarding. In addition, NDN offers in-network caching capabilities, built-in multicast forwarding, mobility support, and security mechanisms. Unlike traditional networking protocols, NDN concentrates on securing content rather than communication channels.

Data signatures are required in NDN, which enables users to access and retrieve any content as long as its signature can be verified, regardless of its origin. NDN employs a

mechanism that involves signing content by its producer and the authority who provided the certificate to the producer. To validate content, a user checks the authenticity of all the certificates involved within the certificate chain till it encounters a trusted or a self-signed network entity (usually a trusted authority). This process helps to enhance the trustworthiness of the data by verifying their authenticity. However, vehicular networks are prone to issues related to their highly dynamic nature. The data verification process employed by NDN may result in significant delays, which is unsuitable for sensitive applications, such as sharing safety messages. In traditional centralized authentication systems, data authentication is often associated with high overheads due to the need to communicate with a central authority. To this end, we propose an approach that enables local data verification while providing a secure authentication mechanism for producer vehicles. Our approach involves leveraging blockchain technology to decentralize the authentication process, making it more efficient for vehicular environments. By using blockchain, we can avoid the need for centralized authorities, thereby reducing the associated overheads. To the best of our knowledge, this study is the first attempt to use blockchain for decentralized data authentication in vehicular NDN. The validation process that we conducted showed that our proposal is secure against attackers.

The paper is structured as follows: Section 2 provides an overview of the NDN architecture. Then, Section 3 examines the related work. The threat model and assumptions for our study are outlined in Section 4. In Section 5, we describe our proposed solution, NDN-BDA. Our work is evaluated and validated in Section 6, and we conclude the paper and discuss future work in Section 7.

2. Named Data Networking: Nuts and Bolts

NDN is an Internet architecture that prioritizes data and intends to substitute the present TCP/IP-centric architecture of the Internet. In particular, it attempts to address the escalating need for communication that is centered on content [6]. NDN identifies two entities: *producers*, who are responsible for creating and offering content, and *consumers*, who seek and retrieve data. In order to obtain content, consumers transmit the name of the desired data via a specific packet. NDN relies on a hierarchical namespace to differentiate content, for example the present paper could be named: ‘‘`com/mdpi/future-Internet/2023/ndn-bda`’’.

NDN relies on two packet types: *Interest* packets and *Data* packets. Consumers use *Interest* packets to request content that producers include within *Data* packets. The most-recent NDN packet specifications [8] mandate that every *Interest* packet must comprise a combination of mandatory and optional parameters. A `Name` must be included in each *Interest* packet, representing the content being sought by the consumer. The `Nonce` field, consisting of a randomly generated string of four octets, is leveraged to uniquely identify *Interest* packets, and thus prevent looping in the network. *Interest* packets may include optional parameters: `CanBePrefix` for the *Interest* packet's name, `MustBeFresh` for the requested content, `InterestLifeTime` representing how long an NDN router will maintain the state for this *Interest*, being `HopLimit` and `ForwardingHint` utilized in forwarding. In addition, *Interest* packets may also include application-specific parameters in the `ApplicationParameters` field. Furthermore, *Interest* packets can be signed if required [9].

Once a consumer sends an *Interest* packet to request particular data, the corresponding response is transmitted through a *Data* packet. In NDN, the producer should sign every *Data* packet. Fundamentally, a *Data* packet comprises three components: the `Name`, the `Content`, which denotes the payload of the *Data* packet, and a `Signature`. Supplementary details such as `ContentType`, `FreshnessPeriod`, and `FinalBlockId` are also encompassed within the *Data* packet.

Each node with the NDN stack maintains three data structures [10]. The Pending Interest Table (PIT) stores all the not-yet-satisfied *Interest* packets, which remain in the PIT until a *Data* packet returns or times out. The PIT entry contains fields such as the name of the requested *Data*, incoming and outgoing interfaces, and an expiry timer. The Content Store (CS) enables NDN to offer in-network caching, where each NDN node can store

passing *Data* packets in a local cache. To maintain its size, each CS needs to implement a caching policy, such as First In First Out (FIFO), Least Recently Used (LRU), and Least Frequently Used (LFU) [11]. The Forwarding Information Base (FIB) is used to forward incoming *Interest* packets to upstream nodes. FIB entries are indexed with name prefixes instead of IP addresses, and each FIB entry consists of a name prefix and a list of next hops. Routers can forward *Interest* packets to one or multiple hops, enabling multi-path forwarding based on their strategy.

2.1. Security Principles in NDN

In a named data network, the primary objective is to locate and retrieve data objects within a specified context by their names. The security features of the NDN design are intrinsic and require signatures to be used in *Data* packets [12]. Producers are responsible for digitally signing all *Data* they create, which enables consumers to validate the authenticity of the received information. Additionally, routers and repositories can store *Data*, and consumers can receive *Data* packets from any source. The NDN architecture relies on several security components to ensure data security [13].

2.1.1. Digital Signatures

Digital signatures in NDN represent the basic building blocks of object security. They enable securing data regardless of their location or transmission channel. The signature field is present in every *Data* packet and serves to bind the content of the packet to its name [14]. It consists of two components, namely `SignatureInfo` and `SignatureValue`. The former contains the producer's public key name and the cryptographic algorithm used to sign the *Data*, while the latter represents the signature's generated bits. The `SignatureInfo` field may also include `KeyLocator`, which includes information regarding the location of parent certificate(s). The signed hash of a *Data* packet covers all its fields except the `SignatureValue`, as illustrated in Figure 1.

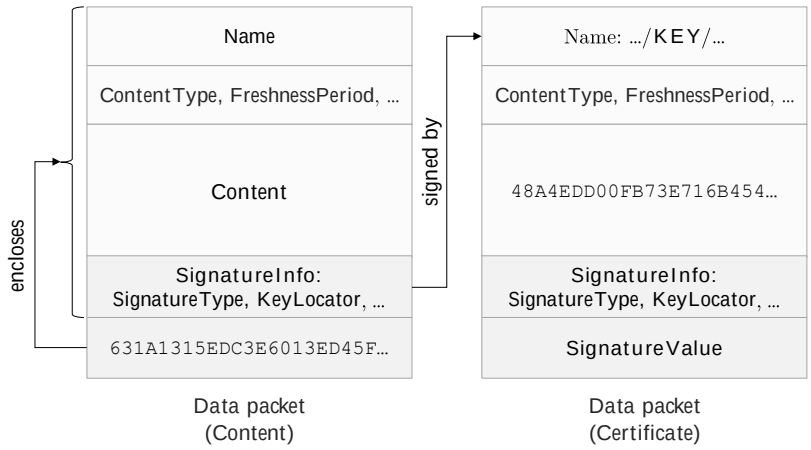

Figure 1. Example of a *Data* packet and its signing certificate.

While NDN does not mandate the use of signatures in *Interest* packets, they may be required in some scenarios where their authenticity is necessary, such as sending a new route announcement or a command packet to an IoT device. Compared to the signature field in *Data* packets, the *Interest* packet's signature field includes additional components, such as `SignatureNonce`, `SignatureTime`, and `SignatureSeqNum`, to add uniqueness to the signature.

2.1.2. NDN Certificates

To ensure secure communication in NDN, data producers need to possess at least one cryptographic key pair. Producers use their private keys to sign *Data* packets, and consumers verify them using public keys. Each public key is encoded in an X509 format digital

certificate signed by an issuer [15]. The NDN certificate's name follows a specific naming convention: `/<IdentityName>/KEY/<KeyId>/<IssuerId>/<Version>`. `IdentityName` represents the namespace in which the key can operate, followed by the keyword `KEY`. The `KeyId` identifies an instance of the public key. Its value can be an 8-byte-long random value, an SHA-256 digest of the public key, a timestamp, or a numerical identifier. The `IssuerId` part identifies the issuer of the certificate. Its value is identical to the `KeyId`. Finally, the version of the certificate is included. In addition to its name, a certificate *Data* packet contains other fields: the `FreshnessPeriod` and the `Content`, which embeds the bits of the DER-encoded public key. Like any NDN Data packet, certificates contain a signature field, which is the signature of the issuer.

2.1.3. Trust Model

The *Data* packet's signature validates the authenticity of the received content and its origin. However, it does not indicate whether the signer has the authorization to produce the received content [16]. To verify whether a producer is authorized to generate data under a given namespace, consumers require a mechanism to check if the producer's key has the right to sign a *Data* packet under that namespace [17]. Application-based trust policies define which keys have the authorization to sign which *Data*, as illustrated in Figure 2. Furthermore, applications can use access control policies to protect the *Data* packet's content through encryption and permit only authorized nodes to decrypt it [18].

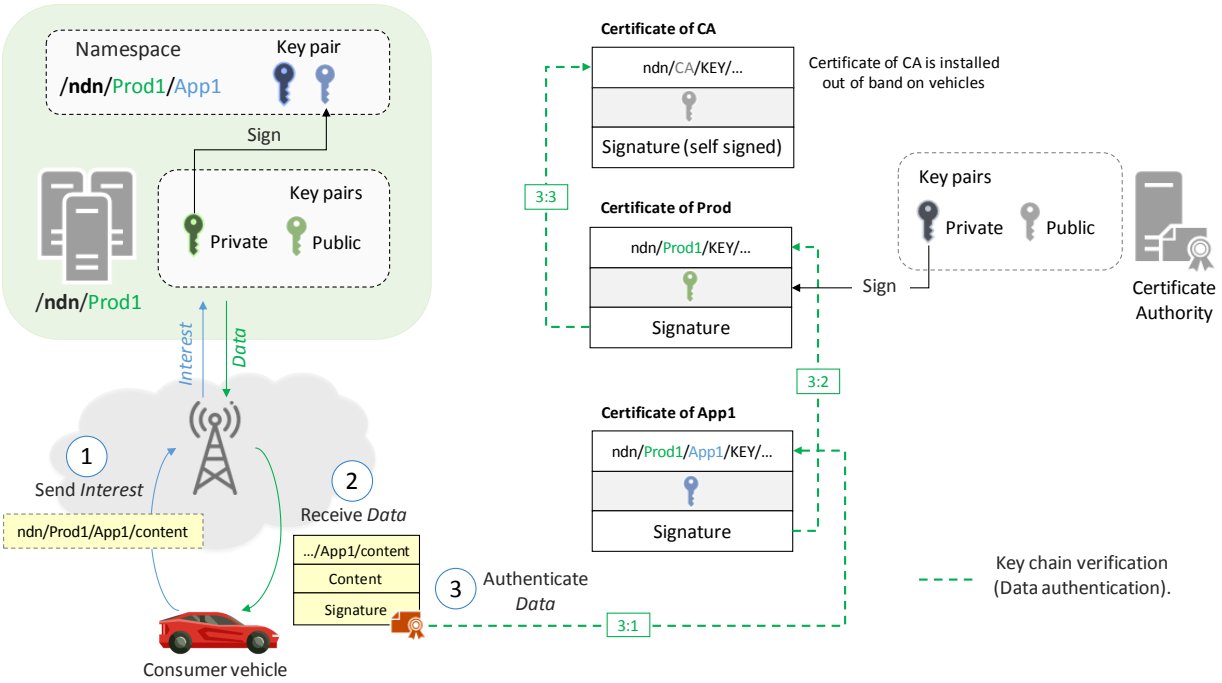

Figure 2. Example of the data authentication process.

2.2. Packet Forwarding

Instead of using IP addresses to forward packets, NDN routers utilize application namespaces [19]. NDN routers employ stateful forwarding, which implies that routers store details of received requests until they are satisfied or timed out. Upon the arrival of an *Interest* packet, the router checks whether it holds a copy of the *Data* in the CS. The NDN node forwards the *Data* downstream if it finds a copy of it. Otherwise, it checks the PIT for similar requests. If a pending request already exists, the router updates the incoming interfaces field. If no pending *Interest* exists, a new entry is created. The NDN node then forwards the *Interest* packet to the upstream neighbor(s) using a forwarding strategy and

the FIB table [20]. The NDN node may respond with a NACK or drop the *Interest* packet if no route is found.

Once a *Data* packet is received by a router from a producer or in-network cache, the router checks for a matching pending *Interest* by examining the name of the received *Data* packet. If there is no match in the PIT, the router drops the received *Data* packet. However, if a match is found, the router caches (or not, depending on its caching strategy) the received *Data* and forwards them downstream to all the interfaces associated with the pending *Interest*. As a result, NDN routers provide a built-in multicast mechanism. Figure 3 illustrates a scenario of packet forwarding in NDN.

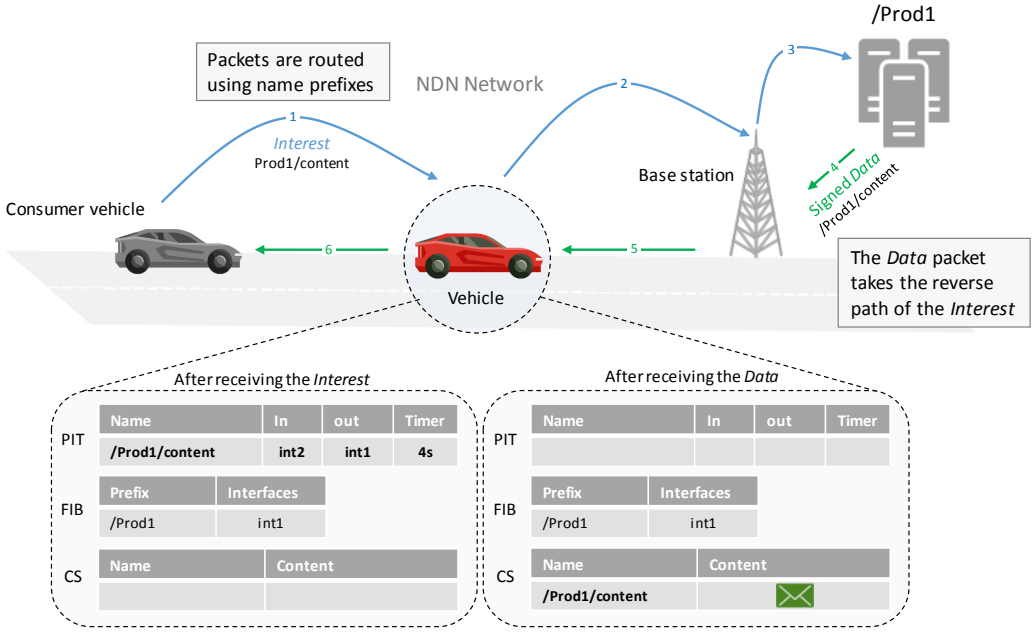

Figure 3. Example of packet forwarding in vehicular NDN.

2.3. Mobility Support in NDN

The packet forwarding process in mobile networks can be challenging due to the constantly changing network topology [21]. However, the NDN paradigm overcomes this issue by breaking the point-to-point nature of IP networks and enabling ubiquitous data retrieval. Unlike IP networks, NDN allows mobile nodes to retrieve data from anywhere in the network, instead of delivering data to a specific mobile node [22]. Furthermore, NDN's implementation allows it to operate over any transport protocol, which is a significant advantage. Additionally, the location-independent data retrieval enabled by NDN's data-centric security is another key advantage to mobility support.

In NDN, mobile consumers are not identified by addresses, which avoids the overhead associated with address management and connection sessions that TCP/IP requires. Instead, when a consumer requests content, the *Interest* packet creates reverse path entries in the PIT as it travels to the data producer. This enables a mobility-friendly forwarding process, as *Data* packets can follow the reverse path to reach the consumer. If the consumer moves to a new location, resending the *Interest* packet is sufficient to update the reverse route to the new location [23].

The simplest way of retrieving the content of a mobile producer consists of broadcasting *Interest* packets throughout the network until they reach the intended mobile producer or the network cache holding the content. However, this approach consumes significant network resources, making it unsuitable for large networks. Another approach involves using a DNS-like node to map produced data names to the actual location of a mobile producer [24]. A consumer sends a query to the *rendezvous* point to ask for the position of the producer. The *rendezvous* point can also act as a relay, which forwards requests to the mobile producer's position and sends back data to the requesting consumer. Data depots

are another approach that leverages the built-in caching capabilities of NDN, allowing consumers to send *Interest* packets to a data depot instead of directly retrieving content from mobile producers [25]. The data depot can then send the requested data if it holds them or fetch them using one of the other approaches. To retrieve data associated with a specific region, a geographic-based forwarding strategy is used where any mobile node in that region can act as a producer. For example, a consumer can request traffic information for a particular road by sending an *Interest* packet to the relevant region using geographic-based routing. When a node in that region receives the request, it either sends the requested data (if it already has them in its cache) or generates them before sending them back to the consumer.

3. Related Work

Several authentication mechanisms have been proposed in recent years. The work in [26] presented a privacy mechanism for NDN-based vehicular networks. The proposal uses pseudonyms to identify vehicles instead of real names. The mechanism involves a vehicle broadcasting an *Interest* to retrieve a key from a nearby proxy. The vehicle then generates a pseudonym and a new key. Following that, it encrypts the new information and its actual key using the manufacturer's public key. It also encrypts the manufacturer's name using the proxy's key. The receiving proxy sends an *Interest* to the manufacturer for key issuance, which creates a certificate for the new key, and encrypts it using the actual vehicle's key. The vehicle decrypts and stores the new certificate and the manufacturer's pseudonym to use for future pseudonyms' generation. Similarly, the work in [27] proposed an anonymity-enabled consumer authentication mechanism. The proposed scheme aims at reducing the computational overhead associated with data authentication by verifying multiple signatures in a batch. When a consumer attempts to retrieve content for the first time, its edge node acts as a temporary content provider. It checks that the consumer has not been corrupted or impersonated before sending the requested content. The edge node and the consumer agree and create a shared key based on the randomly chosen pseudo-identity of each consumer.

The authors in [28] provided a solution for the long-living Data, which outlasts the lifetime of their signatures. The authentication framework, named DeLorean, uses a publicly auditable bookkeeping service that logs the fingerprints of signatures in a Merkle tree. By providing DeLorean proofs for data packets and certificates, the signature validity at the time of data creation can be checked, allowing authentication regardless of the signatures' expiration. The framework includes a data "chronicle", which consists of a sequence of volumes, each containing fingerprints of the witnessed signatures within a fixed timeslot. DeLorean requires auditors to continuously check the consistency of the chronicle to ensure correct and truthful operations. If an auditor detects any modification, it can share it publicly, serving as a deterrent for the service's misbehavior. To overcome the problem related to multiparty authentication, the authors of [29] presented a security support tool for applications named NDN-MPS. It consists of three components, a multi-signature trust schema to enable multiparty authentication policies, an extended key locator data object to keep data consistent across signers, and a signature collection protocol for multi-signature verification. The producer starts the signing process with the multi-signature trust schema, identifies the list of signers, and coordinates the delivery of the data object to be signed. Then, it collects signature values from signers. The producer uses another protocol such as NDN sync [30] to collect signatures from signers. The producer ensures consistency among signers by filling the key locator field with a placeholder name called the late-binding key locator. In the case of failure or the unavailability of signers, the producer can try reaching alternative signers permitted by the trust schema to provide a degree of resiliency.

Another authentication mechanism was proposed in [31]. The authors presented a secure authentication scheme for big-data-enabled ICN to prevent fake data caching. The proposal aimed at establishing trust among unpredictable entities during data acquisition. The mechanism involves three phases: initial trust establishment, data-centric certificate

management, and forwarding-integrated authenticable data retrieval. In the first phase, certificates are issued to a group of highly trusted nodes called Intermediate Physical Entities (IPEs), which are used to cache big data. In the second phase, IPEs store the certificates of their neighboring nodes and other highly trusted entities in a local repository. The third phase consists of constructing a hop-by-hop trust chain to authenticate *Interest* and *Data* packets during forwarding. It utilizes highly trusted IPEs to replace highly trusted certificates of the chain. Additionally, it shortens the certificate chain into one certificate. With the objective of reducing the overhead associated with signature verification, the work in [32] presented a lightweight communication strategy for ICN-based UAV networks. The mechanism utilizes the trust score of the node to determine whether data authenticity needs to be checked. In the case that the source UAV is considered trustworthy, the process of *Data* validation is postponed for a certain period, which is proportional to the UAV's trust score.

Blockchain technology is extensively employed in conjunction with the NDN architecture. Some authors have used it to provide solutions against security attacks, such as [33,34], while others have used it for securing content dissemination [35–37]. The authors of [38] proposed a blockchain-based mechanism to authenticate mobile producers in ICN. The protocol aims to authenticate producers' prefixes and ensure genuine routing updates. The architecture identifies two domains: the core network and the clusters. The clusters consist of both access gateways and a cluster head. Each cluster uses a private ledger managed by the cluster head. The core network includes a subset of core routers that manage the global blockchain, which stores the transactions generated to and from various clusters. The mechanism involves utilizing the mobile device's SIM card to securely distribute and register key pairs. When a producer connects to the network for the first time, the authorization server verifies the prefix announced by the producer by validating its digital signature. Then, it records the producer's public key and digital signature into the blockchain.

In a different context, the work in [39] presented a lightweight consensus mechanism for the IoT. The authors proposed a model named Proof of Evolutionary Model (PoEM) to enhance the consensus efficiency of Blockchain-as-a-Service (BaaS) in the IoT. The model relies on machine learning to achieve consensus. It also includes a mechanism to handle the dynamic nature of IoT environments by managing nodes joining and exiting in real-time.

4. Threat Model and Assumptions

The present paper aimed to address the challenges associated with the certificate chain verification process, which is a critical aspect of ensuring data authenticity and integrity. The proposed solution focuses on enabling consumers to authenticate and verify the content they receive, regardless of their location, while avoiding the overhead associated with validating each certificate involved in the received content. Unlike existing solutions, our approach does not require consumers to have prior knowledge of the trust schema used for data authentication. Instead, we assumed that consumers must possess the blockchain before requesting and verifying data, and each consumer in the network can request and receive the blockchain.

To account for potential adversarial nodes, we made several assumptions. Firstly, we assumed that attackers have the capability to initiate multiple protocol sessions simultaneously. Secondly, we assumed that attackers can access all public data related to the protocol, which gives them the ability to read, store, and block any sent messages. Furthermore, attackers can intercept, construct, and resend messages to their desired destinations. We also assumed that no key breaches occur. To ensure the effectiveness of our proposed mechanism, we needed to develop robust strategies to mitigate the impact of such attacks and enhance the overall security of the system. To this end, we investigated several potential security threats to the system, including impersonation attacks, blockchain tampering attacks, man-in-the-middle attacks, and (D)DoS attacks. In an impersonation attack, the adversary tries to masquerade as a legitimate producer. The blockchain tampering attack, on the other hand, involves the insertion of false or modified data into the blockchain by

a malicious node. The man-in-the-middle attack, as the name implies, occurs when an attacker intercepts and alters the communication between two parties, often for the purpose of stealing or manipulating sensitive information. Finally, the (D)DoS attack involves flooding the system with *Interest* packets in an attempt to overwhelm it, and thereby prevent legitimate requests from being satisfied.

5. NDN-BDA

This section outlines the design of our proposed mechanism, NDN-BDA. It begins with an overview of the mechanism, followed by a detailed description of each step involved in its functioning. Table 1 lists the notations that are used in this paper.

Table 1. List of notations used in this paper.

Notation	Meaning
VRE	Vehicle Registration Entity
MS	Management Server
SCA	System Certification Authority
PV	Producer Vehicle
CV	Consumer Vehicle
si_i	The ith signed *Interest* packet
i_i	The ith *Interest* packet
d_i	The ith *Data* packet
K	Cryptographic Key
$\{m\}K$	Encrypted message m with K
K_{PV}	Public key of PV
K_{PV}^{-1}	Private key of PV
$decrypt(mK_{PV}^{-1}, K_{PV})$	Decrypts a message m encrypted with K_{PV}^{-1} using K_{PV}
BC	Blockchain
adv	Adversary node
$adv(PV) \hookrightarrow MS : m$	adv impersonates PV to send a message m to MS

5.1. Protocol Overview

The significant entities involved in the NDN-BDA protocol are: *Vehicle Registration Entity (VRE)*, *System Certificate Authority (SCA)*, *System Management Server (MS)*, and the blockchain. Figure 4 illustrates the different components of our proposed mechanism. The Vehicle Registration Entity represents the governmental body responsible for providing vehicle registration documents to a vehicle owner. It also acts as a certification authority, generating a personal certificate for the vehicle's owner. Our proposed mechanism involves the System Management Server as a central component responsible for receiving requests from producer vehicles and generating certificates for them using the System Certificate Authority. The Management Server is also tasked with maintaining and distributing the blockchain. It may also employ distribution servers that assist with blockchain distribution. The SCA is a local certification authority maintained by the MS. Finally, the blockchain contains the certificates of vehicle producers.

5.2. Blockchain

Blockchain technology is based on a decentralized and distributed database that is managed by a network of computers. Each block in the chain contains a record of transactions, and once a block is added to the chain, it cannot be altered or deleted. This immutability and transparency of the blockchain make it ideal for use cases where security and trust are critical. The use of blockchain involves utilizing a consensus algorithm, such as Proof of Work (PoW) and Proof of Stake (PoS). It consists of a mechanism that allows nodes to have an identical copy of the blockchain and to add transactions to the blockchain consistently and securely.

NDN-BDA relies on a centralized blockchain maintained by the Management Server. Each block header consists of several components, such as a unique identifier for the block,

which is represented by the name of the certificate that it contains, a timestamp, a nonce, a hash pointer that directs to the previous block in the chain, and the hash of the block signed by the MS. The body of each block includes a producer's certificate. Figure 5 shows the structure of a block.

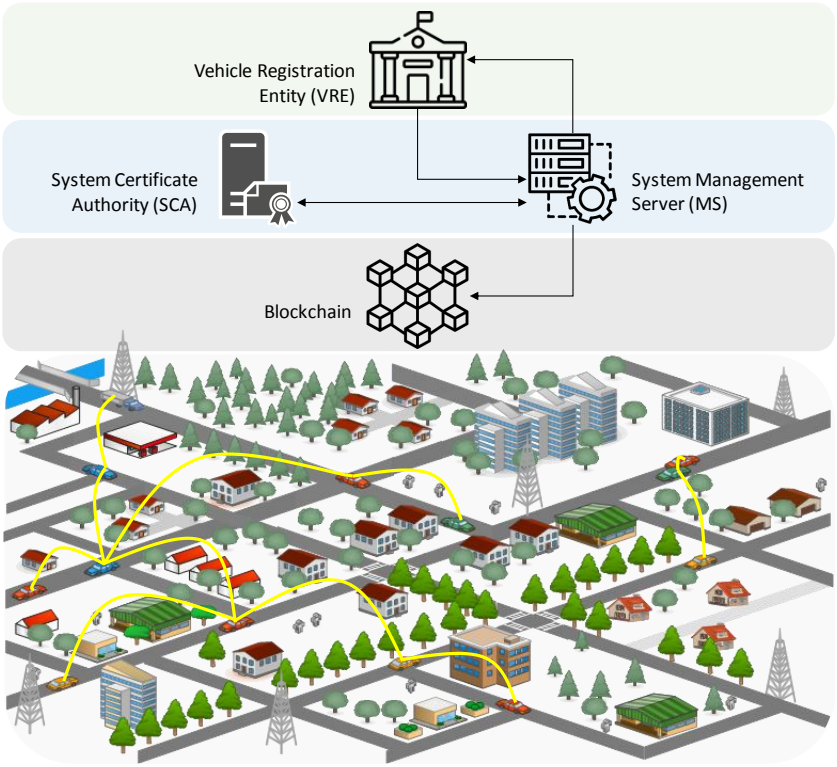

Figure 4. Overview of NDN-BDA.

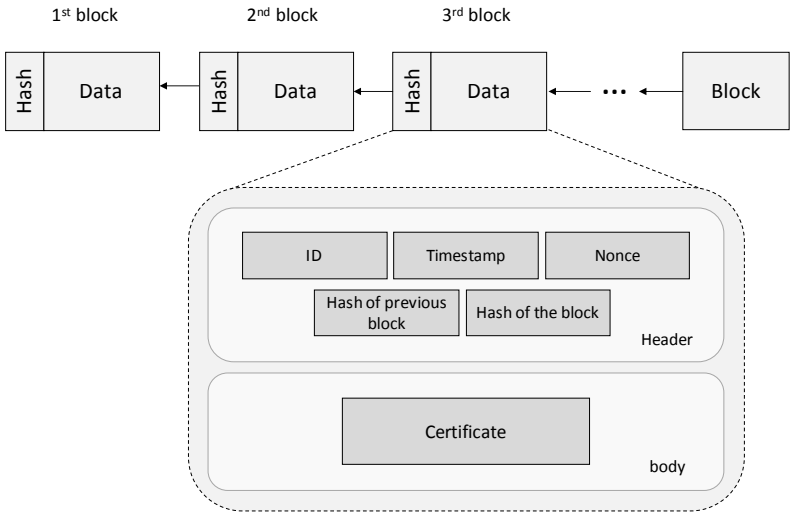

Figure 5. Overview of the structure of the blockchain used by NDN-BDA.

5.3. Trust Schema

The trust schema that NDN-BDA relies on is the Vehicle Registration Entity as the root node. The VRE grants authorization to the System Certification Authority and the Management Server on the name prefixes: ''`/NDN-BDA/`'' and ''`/NDN-DBA/MS/`'', respectively, enabling them to produce *Data* within these name prefixes. Additionally, the Vehicle Registration Entity generates a certificate for each citizen and assigns them a namespace in the format of ''`/NDN-BDA/CID/`'', where `CID` is a unique identifier for each citizen. Each regis-

tered citizen in the Vehicle Registration Entity possesses a car identified by its registration ID. Additionally, each producer vehicle in the network needs to have a certificate issued by the System Certification Authority. The namespace that each vehicle uses to produce *Data* is: ''/NDN-BDA/VID/'', where VID is the registration ID of the car. Figure 6 illustrates the trust schema that NDN-BDA uses.

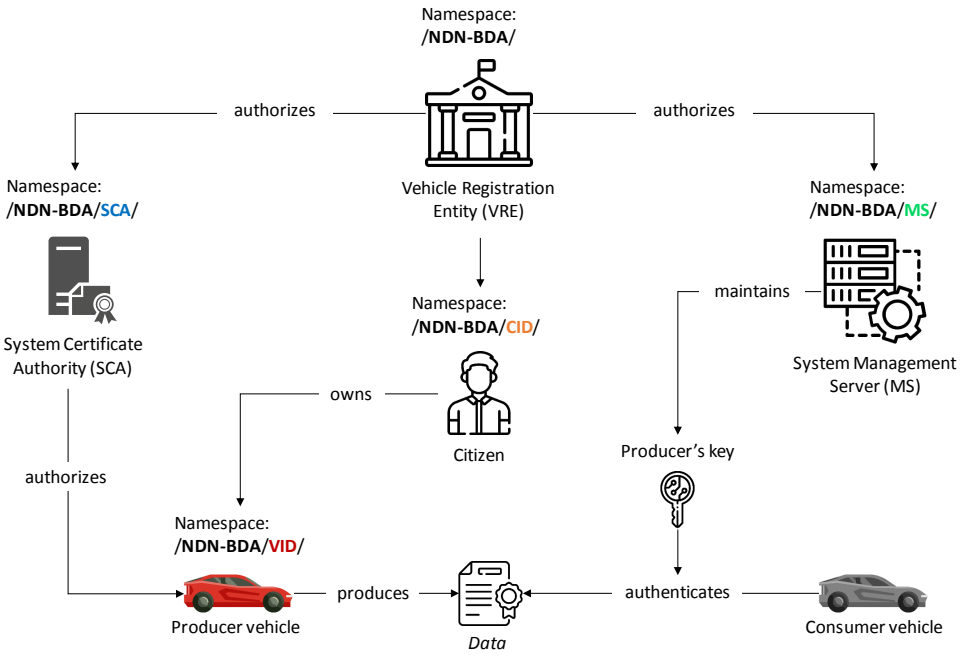

Figure 6. Overview of the trust schema used by NDN-BDA.

5.4. Producer Registration

The producer vehicle registration process in NDN-BDA involves multiple steps, which we discuss in detail below. Figure 7 provides a visual representation of the steps that are involved in the producer registration process.

Step 1: The producer vehicle PV sends an *Interest* packet si_1 with a name similar to: ''/NDN-BDA/MS/join/{CID/VID}K_{MS}'' and signs it with the vehicle owner's private key K_{CID}^{-1}. The name of the signed *Interest* starts with /NDN-BDA/MS to indicate the destination, which is, in this case, the MS. Keyword join indicates to the MS that this message is a request to join the network of producers. The Interest's name also includes the CID and VID information, encrypted with the MS's public key K_{MS}. This ensures that the information remains confidential during transmission, and only the MS can read it.

Step 2: Once the MS receives the signed *Interest* from the producer vehicle PV, it decrypts the associated CID and VID information using its private key: $decrypt(\{\text{CID/VID}\}K_{MS}, K_{MS}^{-1})$. Following that, the MS sends a signed *Interest* si_2 with a name ''/NDN-BDA/MS/join/CID/VID'' to the VRE to verify whether the association CID and VID exist or not in the VRE's database. It checks whether the vehicle's owner, identified by its CID, possesses the vehicle identified by the received VID. Following the reception of the signed *Interest* si_2, the VRE checks the CID/VID association in its database. If it exists, the VRE responds with a *Data* packet d_1 containing the citizen's certificate. Otherwise, it notifies the MS that the received CID/VID association does not exist.

Step 3: After the reception of *Data* packet d_1, the MS stores the received certificate, if it exists, and sends a notification to PV regarding the outcome of its request, indicating whether it was accepted or not.

Step 4: The fourth step of the process involves PV generating an asymmetric key pair (k_{VP}, K_{VP}^{-1}). Once the key pair is generated, PV sends to the MS a signed *Interest* packet si_3 with the name ''/NDN-BDA/MS/sign/{VID/<key>}K_{MS}''. The name contains the keyword sign, which indicates that the PV requests a certificate for its generated public key. The

name of si_3 embeds the value of the public key encrypted with the MS's key K_{MS} to ensure information integrity and secrecy.

Step 5*:* When the `sign` request is received, the MS first verifies the authenticity of si_3 using the certificate that it stored in *Step 3*. Then, the MS decrypts the information of the key that the PV intends to sign, which is performed by using the MS's private key: $decrypt(\{\text{VID/<key>}\}K_{MS}$, $K_{MS}^{-1})$. Following that, the MS generates a signed *Interest* packet si_4 with a name that is identical to "`/NDN-BDA/SCA/sign/CID/VID/<key>`" and sends it to the System Certification Authority (SCA) to create a certificate for the public key K_{VID}.

Step 6*:* Upon the reception of si_4, the SCA generates a certificate for the received public key value in `<key>` and sends back to the MS a *Data* packet d_3 containing the generated certificate.

Step 7*:* When it receives the generated certificate from the SCA, the MS creates a new block in the blockchain BC containing the certificate of the PV. Afterward, the MS transmits the newly generated certificate to the PV within a *Data* packet d_4. With the reception of d_4, the process of the PV registration is complete, and the PV can now generate and send *Data* into the network.

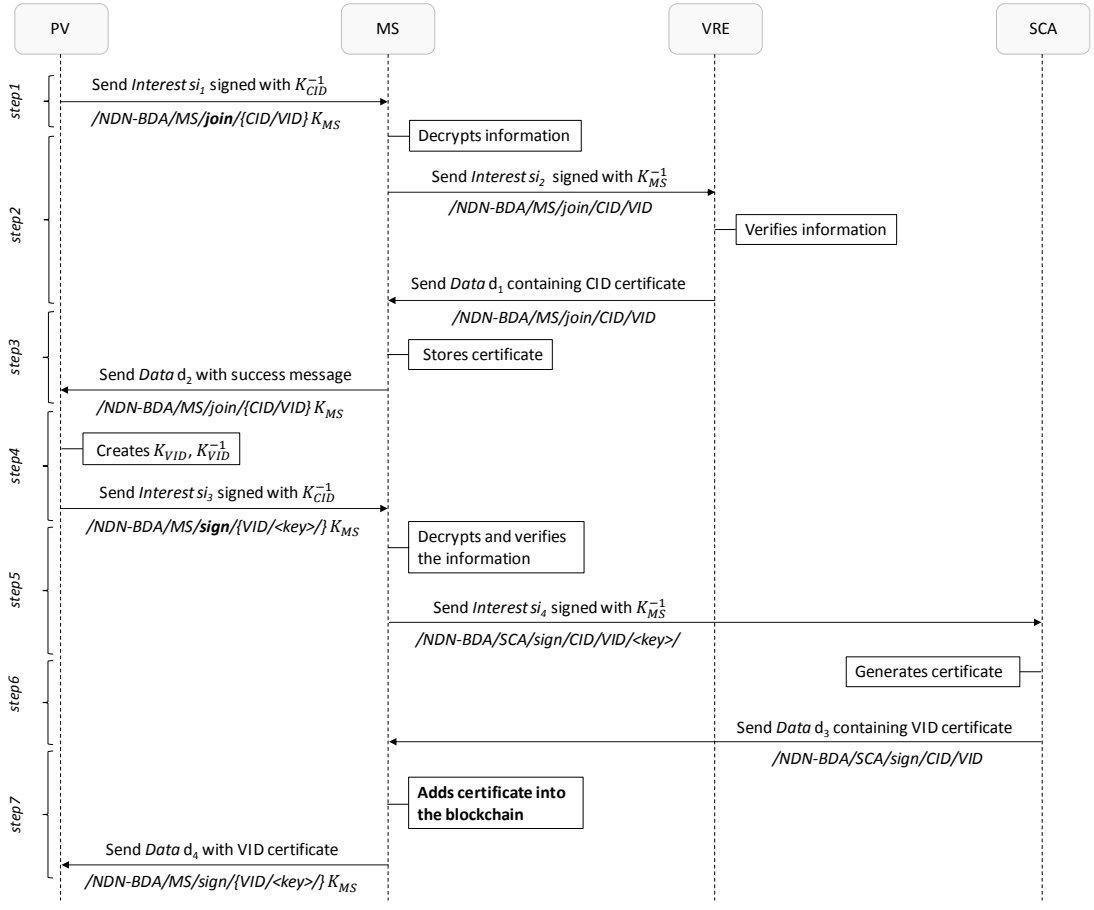

Figure 7. The steps involved in the producer vehicle registration process.

5.5. Data Authentication Process

The NDN-BDA employs a data authentication process that requires each consumer vehicle CV to send an *Interest* packet to request the associated blockchain of the network before it can request and receive the *Data*. To initiate this process, the CV sends an *Interest* i with the name "`/NDN-BDA/MS/BC/request`" towards the MS. Upon receiving the *Interest*, the MS responds with *Data* containing the requested blockchain. The CV will then use the received blockchain as an authentication tool for all the *Data* it receives from producer

vehicles. In order to do so, the consumer vehicle needs to check the blockchain to find the certificate associated with the received *Data*. It checks if it can find the name included in the *Data's* `KeyLocator`. If the certificate is found, the *Data* are validated and accepted. If not, the *Data* are discarded. This process ensures the authenticity of the *Data* being transmitted within the network and provides a secure means of authentication for the CV.

6. Evaluation and Discussion

This section aims to validate our proposal, NDN-BDA. We began by performing a security analysis to test its resilience against the attacks mentioned in Section 4. Following that, we evaluated the effectiveness of NDN-BDA using the validation tool AVISPA [40]. Automated Validation of Internet Security Protocols and Applications (AVISPA) is a tool that analyses and validates cryptographic protocols. It also provides analysis techniques to detect possible vulnerabilities and flaws in protocols.

6.1. Security Analysis

The effectiveness of NDN-BDA against adversary nodes was tested in various scenarios in this section. It was assumed that the adversary *adv* can be any vehicle functioning as a producer or consumer in the network.

6.1.1. Impersonation Attack

In this attack scenario, an adversary *adv* attempts to impersonate another producer by sending an *Interest* packet with a fake or stolen producer CID and VID: $adv(PV) \hookrightarrow MS$: si(`/NDN-BDA/MS/sign/{VID/<key>}`K_{MS}). If *adv* sends an *Interest* with the keyword `join` (*step1*), it will be discarded by the VRE as the `CID` included in the name does not match the key K_{adv}^{-1} that signed the *Interest*. Suppose that the *adv* fakes an *Interest* with the keyword `sign` (*step4*), willing to request a fraudulent certificate. This attempt will also fail, as the *Interest* packet will be rejected by the MS since it does not have the certificate that signed the *Interest*.

6.1.2. Blockchain Tempering Attack

In this attack scenario, an adversary *adv* attempts to insert fake blocks into the blockchain. Suppose that a legitimate consumer vehicle requests the blockchain so it can start verifying the received *Data*. It sends an *Interest* packet to MS asking for the blockchain: $CV \hookrightarrow MS$: i(`/NDN-BDA/MS/BC/request`). Since NDN supports in-network caching, a network cache may satisfy the consumer's request, providing the *adv* with the ability to fulfill the request with a falsified blockchain. The Consumer Vehicle (CV) can detect a fake blockchain using the *Data*'s certificate by relying on the MS's key K_{MS}. The CV can also detect a manipulated block when it receives *Data*. For instance, when the CV receives a *Data* packet from the PV with a particular VID, it searches for the associated certificate in the blockchain. Upon finding it, it verifies the validity of the block by authenticating its hash, checking if it was signed by the MS's key K_{MS}^{-1} or not. If the verification fails, the CV will consider the blockchain as invalid and then request another version from the MS.

6.1.3. Man-in-the-Middle Attack

In this attack scenario, the adversary (*adv*) tries to substitute the key value of a legitimate producer with its own in order to obtain a certificate. For instance, the *adv* intercepts the message: $PV \hookrightarrow MS$: si(`/NDN-BDA/MS/sign/{VID/<key>}`K_{MS}) and then changes the values of `/VID/<key>/` with its own. When it receives the request, the MS will detect that the *Interest*'s information has been altered since the signature value does not match the tampered *Interest*'s hash. The adversary (*adv*) may also try to inject fake information into the message: $MS \hookrightarrow PV$: si(`/NDN-BDA/MS/sign/{VID/<key>}`K_{MS}) containing the generated certificate for the PV. However, the PV will still be able to detect it when it authenticates

the received *Data* with the *MS*'s key K_{MS}, thereby making NDN-BDA resilient against such attacks.

6.1.4. Denial of Service Attack

In NDN, Denial of Service (DoS) attacks are mainly associated with an *Interest* Flooding Attack (IFA) [41]. Such an attack involves sending a large number of *Interest* packets to a target to overwhelm it, making it unable to accept legitimate requests. NDN-BDA, and specifically the Management Server, is also susceptible to DoS attacks. The attack is characterized by the adversary sending an excessive number of `join` and `sign` *Interest* packets with random CID and VID values to overwhelm the MS. The aim is to fill the MS's PIT with these fake requests, making it impossible to add legitimate requests. However, the MS can limit the impact of the attack by discarding successive *Interest* packets with the same signature. The adversary, *adv*, is also capable of launching an IFA using *BC* `request` *Interest* packets. Nevertheless, the impact of this attack is considered less severe since the blockchain could be obtained from network caches, thereby satisfying the *Interest* packets before they reach the *MS*. Despite this, the design of NDN-BDA does not prevent DoS attacks, and the MS should use an IFA mechanism such as the one proposed in [42] to mitigate the attack.

6.2. Protocol Validation

In this section, we evaluated our proposal using AVISPA. AVISPA uses a formal language named High-level Protocol Specification Language (HLPSL) to verify the correctness and security of communication protocols [43]. To ensure the security of our proposed protocol, we identified four roles, namely `role_PV`, `role_MS`, `role_VRE`, and `role_SCA`, which represent the *PV*, *MS*, *VRE*, and *SCA*, respectively. We then established sessions between these entities to simulate the exchange of protocol-based messages. In accordance with the diagram presented in Figure 7, these sessions are as follows: (1) the `PV` and `MS` establish two sessions, one for the `join` message and another one for the `sign` message; (2) the *MS* and *VRE* establish a session to verify the information of the `join` message received from the *PV*; (3) the *MS* establishes a session with the *SCA* to send a certificate request message. The validation process aims to ensure the secrecy of the information sent within two messages:

- $PV \hookrightarrow MS$: si(`/NDN-BDA/MS/join/{CID/VID}`K_{MS}).
- $PV \hookrightarrow MS$: si(`/NDN-BDA/MS/sign/VID/<key>`$/K_{MS}$).

Additionally, it aims to authenticate several messages:

- $PV \hookrightarrow MS$: si(`/NDN-BDA/MS/join/{CID/VID}`K_{MS}).
- $MS \hookrightarrow VRE$: si(`/NDN-BDA/MS/join/CID/VID`).
- $VRE \hookrightarrow MS$: d(`/NDN-BDA/MS/join/CID/VID`).
- $MS \hookrightarrow PV$: d(`/NDN-BDA/MS/join/{CID/VID}`K_{MS}).
- $PV \hookrightarrow MS$: si(`/NDN-BDA/MS/sign/VID/<key>`$/K_{MS}$.
- $MS \hookrightarrow SCA$: si(`/NDN-BDA/SCA/sign/CID/VID/<key>`).
- $SCA \hookrightarrow MS$: d(`/NDN-BDA/SCA/sign/CID/VID`).
- $MS \hookrightarrow PV$: d(`/NDN-BDA/MS/sign/VID/<key>`$/K_{MS}$.

The HLPSL role specification of the system nodes, namely the *VP*, *MS*, *VRE*, and *SCA*, is presented in Figure 8. The definition code of the *MS* node is represented from *Line 3* to *Line 36*. The *MS* interacts with all the other entities in the system. Therefore, we need to add all the system nodes as agents within the definition code of *MS*, as shown in *Line 3*. The role specification of the *MS* utilizes the public keys belonging to the *MS*, *VP*, *VRE*, and *SCA*, denoted as `Kms`, `Kpv`, `Kvre`, and `Ksca`, respectively. It also employs two communication channels, `SND`, used to send messages, and `RCV`, to receive messages. It also uses local variables to store/create incoming/outgoing messages (*Line 7*). The *MS* role specification includes a set of transitions, which outline all the actions that the *MS* performs during communication sessions (*Lines 11–35*). The HLPSL specification defines a set of actions. Along with `SND` and `RCV`, an action may also be `secret`, which verifies

message encryption, `witness` to enable message authentication, and `wrequest` to verify the message's authenticity. For example, in *Line 16*, the *MS* requests the message that it sends to the *VRE* to be authenticated. Similarly, in *Line 13*, the *MS* verifies the authenticity of the message that it received from *PV*.

The *PV* role specification starts from *Line 39* and ends in *Line 59*. *PV* employs only two agents: the `VP` and `MS`, as the *PV* communicates only with the *MS* (*Line 39*). The `role_PV` definition also defines the actions performed by the *PV*, which includes sending secret messages (*Lines 48 and 53*).

Similarly, the *VRE* specification ranges from *Line 62* to *Line 77*. It also defines two agents: the `VRE` and `MS`, with whom it communicates. The *VRE* performs two actions. It verifies the received message from the *MS* (*Lines 70–72*) and sends a data packet to *MS* (*Lines 75–77*). The *SCA* also defines two agents, and it performs two actions: receiving a message from the *MS* (*Line s89–90*) and sending the data to the *MS* (*Lines 92–94*).

```
1. %% Definition of the MS role
2. %% ========================
3. Role
   role_MS(MS:agent,PV:agent,VRE:agent,SCA:agent,S2,D4:text,m
   s,Kpv,Kvre,Ksca:public_key,SND,RCV:channel(dy))
4. played_by MS
5. def=
6. local
7.   State:nat,S,D,D3:text
8. init
9.   State := 0
10. transition
11. 1. State=0 /\ RCV({PV.{S'}_inv(Kpv)}_Kms)  =|>  State':=1
12. %% MS checks the autheticity of PV->MS:si1
13. /\ wrequest(MS,PV,auth_1,S')
14. %% Sending the message MS->VRE:si2
15. /\ SND({MS.{S2}_inv(Kms)}_Kvre)
16. /\ witness(MS,VRE,auth_2,S2)
17. %% Receiving the message VRE->MS:d1
18. /\ D':=new()
19. /\ RCV({VRE.{D'}_inv(Kvre)}_Kms)
20. /\ wrequest(MS,VRE,auth_3,D')
21. 2. State=1 /\ RCV({PV.{S'}_inv(Kpv)}_Kms)  =|>  State':=2
22. %% MS checks the autheticity of PV->MS:si1
23. /\ wrequest(MS,PV,auth_4,S')
24. %% Sending the message MS->SCA:si4
25. /\ S2':=new()
26. /\ SND({MS.{S2'}_inv(Kms)}_Ksca)
27. /\ witness(MS,SCA,auth_5,S2')
28. 3. State=2
29. %% Receiving the message SCA->MS:d3
30. /\ RCV({SCA.{D3'}_inv(Ksca)}_Kms)  =|>  State':=3
31. /\ wrequest(MS,SCA,auth_6,D3')
32. %% Sending the message MS->PV:d4
33. %/\ S2':=new()
34. /\ SND({MS.{D4}_inv(Kms)}_Kpv)
35. /\ witness(MS,PV,auth_7,D4)
36. end role

37. %% Definition of the PV role
38. %% ========================
39. role
   role_PV(PV:agent,MS:agent,S:text,Kms,Kpv:public_key,SND,RC
   V:channel(dy))
40. played_by PV
41. def=
42. local
43.   State:nat,D4:text
44. init
45.   State := 0
46. transition
47. 1. State=0 /\ RCV(start) =|> State':=1 /\
   SND({PV.{S}_inv(Kpv)}_Kms)
48. /\ secret(S,sec_1,{PV,MS})
49. /\ witness(PV,MS,auth_1,S)
50. %% Sending the message PV->MS:si3
51. /\ S':=new()
52. /\ SND({PV.{S'}_inv(Kpv)}_Kms)
53. /\ secret(S,sec_2,{PV,MS})
54. /\ witness(PV,MS,auth_4,S')
55. 2. State=1
56. %% Receiving the message MS->PV:d4
57. /\ RCV({MS.{D4'}_inv(Kms)}_Kpv)  =|> State':=2
58. /\ wrequest(PV,MS,auth_7,D4')
59. end role

60. %% Definition of the VRE role
61. %% ========================
62. role
   role_VRE(VRE:agent,MS:agent,D1:text,Kms,Kvre:public_key,SND,
   RCV:channel(dy))
63. played_by VRE
64. def=
65. local
66.   State:nat,S:text
67. init
68.   State := 0
69. transition
70. 1. State=0 /\ RCV({MS.{S'}_inv(Kms)}_Kvre)  =|>  State':=1
71. %% VRE checks the autheticity of MS->VRE:si2
72. /\ wrequest(VRE,MS,auth_2,S')
73. 2. State=1
74. %% VRE sends the message VRE->MS:d1
75. /\ SND({VRE.{D1}_inv(Kvre)}_Kms) =|>  State':=2
76. /\ witness(VRE,MS,auth_3,D1)
77. end role

78. %% Definition of the SCA role
79. %% ========================
80. role
   role_SCA(SCA:agent,MS:agent,D3:text,Kms,Ksca:public_key,SND,
   RCV:channel(dy))
81. played_by SCA
82. def=
83. local
84.   State:nat,S:text
85. init
86.   State := 0
87. transition
88. %% Receiving the message MS->SCA:si4
89. 1. State=0 /\ RCV({MS.{S'}_inv(Kms)}_Ksca)  =|>  State':=1
90. /\ wrequest(SCA,MS,auth_5,S')
91. %% Sending the message SCA->MS:d3
92. /\ SND({SCA.{D3}_inv(Ksca)}_Kms)
93. /\ witness(SCA,MS,auth_6,D3)
94. end role
```

Figure 8. HLPSL specification code used for the definition of the NDN-BDA roles.

Figure 9 shows the HLPSL specification code for the communication sessions (*Lines 97–104*), the simulation environment (*Lines 107–120*), and the security goals of the system (*Lines 123–127*). A session consists of the roles `PV`, `MS`, `VRE`, and `SCA`, respec-

tively. The knowledge assumptions regarding the adversary node are specified using `intruder_knowledge` inside the environment role (*Line 115*). It assumes that the adversary knows all the agents in addition to their public keys. Finally, the security objectives are defined in the goal specification (*Lines 123–127*). The `secrecy_of` function in *Line 124* verifies the secrecy of messages `sec_1` and `sec_2` that *PV* sends, while the `weak_authentication` function was used to validate the authenticity of all exchanged messages.

```
95.%% Definition of the session
96.%% ========================
97.role
   session(MS:agent,PV:agent,VRE:agent,SCA:agent,S:text,Kms,K
   pv,Kvre,Ksca:public_key)
98.def=
99.local
100.   SND1,RCV1,SND2,RCV2,SND3,RCV3,SND4,RCV4:channel(dy)
101.composition
102.role_PV(MS,PV,S,Kms,Kpv,SND2,RCV2)
   /\role_MS(MS,PV,VRE,SCA,S,S,Kms,Kpv,Kvre,Ksca,SND1,RCV1)
103./\role_VRE(VRE,MS,S,Kms,Kvre,SND3,RCV3)
   /\role_SCA(SCA,MS,S,Kms,Ksca,SND4,RCV4)
104.end role

105.%% Definition of the environment
106.%% ============================
107.role environment()
108.def=
109.const
110.   pv,ms,vre,sca:agent,
111.   s1,s2,s3:text,
112.   kms,kpv,kvre,ksca:public_key,
113.   sec_1, sec_2, auth_1, auth_2, auth_3, auth_4, auth_5,
   auth_6, auth_7:protocol_id
114.%% Knowledge of the adversary
115.intruder_knowledge = {pv,ms,vre,kms,kpv,kvre,ksca}
116.composition
117.   session(pv,ms,vre,sca,s1,kms,kpv,kvre,ksca)
118.   /\session(ms,vre,i,i,s2,kms,kpv,kvre,ksca)
119.   /\session(ms,pv,i,i,s3,kms,kpv,kvre,ksca)
120.end role

121.%% Definition of the objectives
122.%% ============================
123.goal
124.   secrecy_of sec_1, sec_2
125.   weak_authentication_on auth_1, auth_2, auth_3, auth_4,
   auth_5, auth_6, auth_7
126.end goal

127.environment()
```

Figure 9. HLPSL specification code used to define the simulation environment and communication sessions.

The specified scenario was executed in AVISPA using the ATtack SEarcher (ATSE) backend, and it was animated using a tool called Security Protocol Animator (SPAN). Our proposed mechanism was deemed safe against intruders and met the security requirements of the threat model implemented in AVISPA.

7. Conclusions and Future Works

In the context of vehicular networks, the process of verifying data in NDN can lead to significant delays due to their highly dynamic nature. In this paper, we introduced NDN-BDA, a decentralized data authentication mechanism based on blockchain technology, which is both efficient and straightforward. Our proposal was thoroughly evaluated for efficiency against various attacks, and its functionality was validated using the widely used AVISPA tool.

As future work, we aim to enhance our solution by including a certificate revocation process and expanding it to a distributed mechanism that utilizes multiple Management Servers maintaining a global blockchain.

Author Contributions: The authors equally contributed to this work in all aspects with more implementation-related efforts from A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was derived from the R&D project PID2021-122580NB-I00, funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

Data Availability Statement: All implementation details, sources, and data will be delivered upon request to the corresponding author Carlos T. Calafate.

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

References

1. Cisco Annual Internet Report (2018–2023) White Paper. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-Internet-report/white-paper-c11-741490.html (accessed on 7 March 2023).
2. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 1024–1049. [CrossRef]
3. Pan, J.; Paul, S.; Jain, R. A survey of the research on future Internet architectures. *IEEE Commun. Mag.* **2011**, *49*, 26–36. [CrossRef]
4. Koponen, T.; Chawla, M.; Chun, B.G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007; pp. 181–192.
5. Fotiou, N.; Nikander, P.; Trossen, D.; Polyzos, G.C. Developing information networking further: From PSIRP to PURSUIT. In *Proceedings of the Broadband Communications, Networks, and Systems, 7th International ICST Conference, BROADNETS 2010, Athens, Greece, 25–27 October 2010*; Revised Selected Papers 7; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–13.
6. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.C.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
7. Jacobson, V.; Mosko, M.; Smetters, D.; Garcia-Luna-Aceves, J. *Content-Centric Networking*; Whitepaper; Palo Alto Research Center: Palo Alto, CA, USA , 2007; pp. 2–4.
8. NDN Packet Format Specification Version 0.3. Available online: https://named-data.net/doc/NDN-packet-spec/current (accessed on 3 March 2023).
9. Signed Interest. NDN Packet Format Specification Version 0.3. Available online: https://named-data.net/doc/NDN-packet-spec/current/signed-interest.html (accessed on 3 March 2023).
10. Zhang, H.; Li, Y.; Zhang, Z.; Afanasyev, A.; Zhang, L. Ndn host model. *ACM SIGCOMM Comput. Commun. Rev.* **2018**, *48*, 35–41. [CrossRef]
11. Zhang, M.; Luo, H.; Zhang, H. A survey of caching mechanisms in information-centric networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1473–1499. [CrossRef]
12. Zhang, Z.; Yu, Y.; Zhang, H.; Newberry, E.; Mastorakis, S.; Li, Y.; Afanasyev, A.; Zhang, L. An overview of security support in named data networking. *IEEE Commun. Mag.* **2018**, *56*, 62–68. [CrossRef]
13. Tehrani, P.F.; Osterweil, E.; Schmidt, T.C.; Wählisch, M. SoK: Public key and namespace management in NDN. In Proceedings of the 9th ACM Conference on Information-Centric Networking, Osaka, Japan, 19–21 September 2022; pp. 67–79.
14. Li, Y.; Zhang, Z.; Wang, X.; Lu, E.; Zhang, D.; Zhang, L. A secure sign-on protocol for smart homes over named data networking. *IEEE Commun. Mag.* **2019**, *57*, 62–68. [CrossRef]
15. Zhang, Z.; Yu, Y.; Afanasyev, A.; Zhang, L. *NDN Certificate Management Protocol (NDNCERT)*; Technical Report NDN-0050; NDN: Helensvale, Australia, 2017.
16. Zhang, Z.; Yu, Y.; Ramani, S.K.; Afanasyev, A.; Zhang, L. Nac: Automating access control via named data. In Proceedings of the MILCOM 2018—2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; pp. 626–633.
17. Nour, B.; Khelifi, H.; Hussain, R.; Mastorakis, S.; Moungla, H. Access control mechanisms in named data networks: A comprehensive survey. *ACM Comput. Surv.* **2021**, *54*, 61. [CrossRef]
18. Lee, C.A.; Zhang, Z.; Tu, Y.; Afanasyev, A.; Zhang, L. Supporting virtual organizations using attribute-based encryption in named data networking. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 188–196.
19. Li, Z.; Xu, Y.; Zhang, B.; Yan, L.; Liu, K. Packet forwarding in named data networking requirements and survey of solutions. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1950–1987. [CrossRef]
20. Voitalov, I.; Aldecoa, R.; Wang, L.; Krioukov, D. Geohyperbolic routing and addressing schemes. *ACM SIGCOMM Comput. Commun. Rev.* **2017**, *47*, 11–18. [CrossRef]
21. Zhang, Y.; Afanasyev, A.; Burke, J.; Zhang, L. A survey of mobility support in named data networking. In Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, USA, 10–14 April 2016; pp. 83–88.
22. Khelifi, H.; Luo, S.; Nour, B.; Moungla, H.; Faheem, Y.; Hussain, R.; Ksentini, A. Named data networking in vehicular ad hoc networks: State-of-the-art and challenges. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 320–351. [CrossRef]
23. Zhu, Z.; Afanasyev, A.; Zhang, L. A New Perspective on Mobility Support. NDN, Technical Report NDN-0013. 2013. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a6fc488646c27d13191691f7145745ba828cff9b (accessed on 30 March 2023) .
24. Afanasyev, A.; Jiang, X.; Yu, Y.; Tan, J.; Xia, Y.; Mankin, A.; Zhang, L. NDNS: A DNS-like name service for NDN. In Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–9.
25. Zhang, Y.; Xia, Z.; Mastorakis, S.; Zhang, L. Kite: Producer mobility support in named data networking. In Proceedings of the 5th ACM Conference on Information-Centric Networking, Boston, MA, USA, 21–23 September 2018; pp. 125–136.
26. Chowdhury, M.; Gawande, A.; Wang, L. Anonymous authentication and pseudonym-renewal for VANET in NDN. In Proceedings of the 4th ACM Conference on Information-Centric Networking, Berlin, Germany, 26–28 September 2017; pp. 222–223.

27. Lu, Y.; Wang, C.; Yue, M.; Wu, Z. Consumer-source authentication with conditional anonymity in information-centric networking. *Inf. Sci.* **2023**, *624*, 378–394. [CrossRef]
28. Yu, Y.; Afanasyev, A.; Seedorf, J.; Zhang, Z.; Zhang, L. NDN DeLorean: An authentication system for data archives in named data networking. In Proceedings of the 4th ACM Conference on Information-Centric Networking, Berlin, Germany, 26–28 September 2017; pp. 11–21.
29. Zhang, Z.; Liu, S.; King, R.; Zhang, L. Ndn-mps: Supporting multiparty authentication over named data networking. In Proceedings of the 8th ACM Conference on Information-Centric Networking, Paris, France, 22–24 September 2021; pp. 83–94.
30. Li, T.; Shang, W.; Afanasyev, A.; Wang, L.; Zhang, L. A brief introduction to ndn dataset synchronization (ndn sync). In Proceedings of the MILCOM 2018—2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; pp. 612–618.
31. Li, R.; Asaeda, H.; Wu, J. DCAuth: Data-centric authentication for secure in-network big-data retrieval. *IEEE Trans. Netw. Sci. Eng.* **2018**, *7*, 15–27. [CrossRef]
32. Barka, E.; Kerrache, C.A.; Hussain, R.; Lagraa, N.; Lakas, A.; Bouk, S.H. A trusted lightweight communication strategy for flying named data networking. *Sensors* **2018**, *18*, 2683. [CrossRef]
33. Zhu, K.; Chen, Z.; Yan, W.; Zhang, L. Security attacks in named data networking of things and a blockchain solution. *IEEE Internet Things J.* **2018**, *6*, 4733–4741. [CrossRef]
34. Goyal, S.; Sharma, N.; Kaushik, I.; Bhushan, B. Blockchain as a solution for security attacks in named data networking of things. In *Security and Privacy Issues in IoT Devices and Sensor Networks*; Academic Press: Cambridge, MA, USA, 2021; pp. 211–243.
35. Chen, C.; Wang, C.; Qiu, T.; Lv, N.; Pei, Q. A secure content sharing scheme based on blockchain in vehicular named data networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3278–3289. [CrossRef]
36. Rawat, D.B.; Doku, R.; Adebayo, A.; Bajracharya, C.; Kamhoua, C. Blockchain enabled named data networking for secure vehicle-to-everything communications. *IEEE Netw.* **2020**, *34*, 185–189. [CrossRef]
37. Khelifi, H.; Luo, S.; Nour, B.; Moungla, H.; Ahmed, S.H.; Guizani, M. A blockchain-based architecture for secure vehicular Named Data Networks. *Comput. Electr. Eng.* **2020**, *86*, 106715. [CrossRef]
38. Conti, M.; Hassan, M.; Lal, C. BlockAuth: BlockChain based distributed producer authentication in ICN. *Comput. Netw.* **2019**, *164*, 106888. [CrossRef]
39. Zhao, Y.; Qu, Y.; Xiang, Y.; Zhang, Y.; Gao, L. A Lightweight Model-Based Evolutionary Consensus Protocol in Blockchain as a Service for IoT. *IEEE Trans. Serv. Comput.* **2023**, *Early Access*.
40. Vigano, L. Automated security protocol analysis with the AVISPA tool. *Electron. Notes Theor. Comput. Sci.* **2006**, *155*, 61–86. [CrossRef]
41. Benmoussa, A.; Kerrache, C.A.; Lagraa, N.; Mastorakis, S.; Lakas, A.; Tahari, A.E.K. Interest Flooding Attacks in Named Data Networking: Survey of Existing Solutions, Open Issues, Requirements and Future Directions. *ACM Comput. Surv.* **2021**, *55*, 139. [CrossRef]
42. Benmoussa, A.; El Karim Tahari, A.; Kerrache, C.A.; Lagraa, N.; Lakas, A.; Hussain, R.; Ahmad, F. MSIDN: Mitigation of sophisticated interest flooding-based DDoS attacks in named data networking. *Future Gener. Comput. Syst.* **2020**, *107*, 293–306. [CrossRef]
43. Chevalier, Y.; Compagna, L.; Cuellar, J.; Drielsma, P.H.; Mantovani, J.; Mödersheim, S.; Vigneron, L. A high level protocol specification language for industrial security-sensitive protocols. In Proceedings of the Workshop on Specification and Automated Processing of Security Requirements-SAPS'2004, Linz, Austria, 20–25 September 2004; Austrian Computer Society: Vienna, Austria, 2004; p. 13.