future internet

MDPI

Article

Reliable Application Layer Routing Using Decentralized Identifiers

Khalid Alsubhi [1], Bander Alzahrani [1], Nikos Fotiou [2,*], Aiiad Albeshri [1] and Mohammed Alreshoodi [3]

[1] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 37848, Saudi Arabia
[2] Department of Informatics, School of Information Sciences and Technology, Athens University of Economics and Business, 10434 Athens, Greece
[3] Unit of Scientific Research, Applied College, Qassim University, Buraydah 52211, Saudi Arabia
* Correspondence: fotiou@aueb.gr

Abstract: Modern internet of things (IoT) applications can benefit from advanced communication paradigms, including multicast and anycast. Next-generation internet architectures, such as information-centric networking (ICN), promise to support these paradigms, but at the same time they introduce new security challenges. This paper presents a solution that extends an ICN-like architecture based on software-defined networking (SDN) that supports those communication paradigms. Using the proposed solution, the underlying architecture is enhanced with a novel security mechanism that allows content "advertisements" only from authorized endpoints. This mechanism prevents "content pollution", which is a significant security threat in ICN architectures. The proposed solution is lightweight, and it enables identity sharing as well as secured and controlled identity delegation.

Keywords: group communication; internet of things; software-defined networking; source routing

Citation: Alsubhi, K.; Alzahrani, B.; Fotiou, N.; Albeshri, A.; Alreshoodi, M. Reliable Application Layer Routing Using Decentralized Identifiers. *Future Internet* **2022**, *14*, 322. https://doi.org/10.3390/fi14110322

Academic Editor: Sachin Sharma

Received: 7 October 2022
Accepted: 2 November 2022
Published: 6 November 2022

1. Introduction

The predominant client–server communication paradigm of the current internet, although it works, impedes the exploitation of the full potentials of emerging architectures, such as the internet of things.

The internet of things (IoT) promises to interconnect users, software, and devices, enabling novel, innovating applications. Nevertheless, many of the envisioned IoT systems require advanced communication paradigms, such as multicast and anycast. Many IoT scenarios can benefit from group communication toward devices that share some properties (e.g., all smart lamps of a building). Most legacy IoT systems provide this functionality by relying on a centralized entity, which acts as an indirection layer that maps application-layer semantic-rich identifiers (e.g., building1.floor5.lamps) to the corresponding network addresses (e.g., CoAP and CoAP group communication relies on DNS, and MQTT relies on a MQTT "broker"). Nevertheless, these systems cannot efficiently handle dynamic networks (i.e., network with IoT devices that experience churn or move around), neither can they easily support multiple administrative domains.

Information-centric networking (ICN) [1] in a next-generation internet architecture that can overcome many of the limitations of IP-based networks and provide the desired functionality without requiring a centralized indirection point. ICN architectures allow endpoints to "advertise" content identifiers, extending, at the same time, network routers to operate using content identifiers (rather than network identifiers). ICN facilitates caching, and it supports advanced communication paradigms (such as multicast). For these reasons, ICN architectures are considered by many research efforts to be underlays for IoT systems (see, for example, the survey of Nour et al. [2]). However, ICN-based systems not only require modifications to the networking infrastructure, but they also introduce new security threats, such as "pollution" attacks that require new security solutions. Content pollution attacks are a kind of denial-of-service attack, where an attacker injects fake content into the network [3]. Fake content can be cached, amplifying, in this way, the impact of the

attack [4]. One root cause of a content pollution attack is that any entity is allowed to advertise and provide any content item.

This paper considers an architecture that abides by the principles of ICN; it has the same advantages as ICN but, at the same time, it does not require modifications to the networking architecture. Then, it proposes an efficient solution to prevent content pollution attacks. The considered architecture leverages software-defined networking (SDN), and it was originally proposed in [5]. That architecture allows stateless broadcast and multicast by using content identifier advertisements and Bloom-filters-based network communication. In particular, in this architecture, Bloom filters are used for "encoding" the network path that a packet must follow [6]; additionally, edge nodes advertise content identifiers, and these advertisements are used for creating lookup tables that map a content identifier to the Bloom filter that should be used in order to reach the nodes that have advertised it. In order to solve the problem of content pollution, a solution is proposed that enables edge nodes to validate that an identifier is advertised by an *authorized* entity: this is achieved using decentralized identifiers (DIDs) [7].

A DID is a URI-like identifier which resolves to a "DID document" that includes "information" about the DID owner. DIDs are a standardized self-sovereign identity (SSI) mechanism. SSIs have received increased attention in the context of the IoT [8]. The proposed system allows DIDs to be used as content identifiers and a DID document includes information that can be used for validating that an identifier advertisement has originated from an *authorized* entity. Our paper makes the following contributions:

- It proposes a solution that integrates DIDs with a registration system allowing ordinary, human readable URLs to be used as DIDs. These URLs can then be used for identifying and/or grouping IoT devices.
- It defines methods that allow DID owners to securely bind a DID to IoT devices, enabling, in this way, IoT devices to generate "proofs of ownership", which are used as the main building block of our security solution.
- It defines protocols that allow DID owners to perform identifier *delegation* to a third party in a secure and controlled way.
- It designs the solution to be lightweight since the only operation that an IoT device should perform is the generation of a digital signature and the only operation that an edge node should perform is the validation of that signature.

The remainder of this paper is organized as follows. Section 2 presents background information, and it introduces DIDs and the SDN-based underlay architecture. Section 3 discusses related work in this area. Section 4 details the design of the proposed solution. The implementation and evaluation of the proposed solution are presented in Section 5. Section 6 discusses alternative DID methods and ICN underlays as well as the impact of the proposed solution in lightweight devices. Finally, we present our conclusions in Section 7.

2. Background

2.1. Decentralized Identifiers

Decentralized identifiers (DIDs) are a new type of globally unique identifier, recently standardized by W3C. DIDs are designed to enable individuals and organizations to generate their own identifiers using systems they trust [9]. Due to their intriguing security and privacy properties, DIDs have been investigated as a security solution for many types of systems, including IoT-based ones (see, for example, [10–12]).

A DID systems is akin to a key–value lookup architecture, where the key is the decentralized identifier (DID) and the value is a DID *document*. The actual contents of a DID document, as well as how it is resolved, depend on the implementation of the specific DID *method*.

A DID is a string consisting of three parts: (1) the "did" URI scheme identifier, (2) the identifier for the DID method, and (3) the DID method-specific identifier [9]. An example of a DID for the "example" DID method follows.

$$did : example : fddc3256$$

A DID document may include, among other things, public keys (or "pointers" to public keys) that can be used as *verification methods*, e.g., an "authentication" key used for authenticating the DID "owner", or an "assertion" key used for verifying digital signatures generated by the DID owner. A DID document is usually maintained by a DID *registry*. A registry implements proper security and access control mechanisms. Registries allow the DID document *owner* to manage their corresponding DID documents, as well as third parties to look up DID documents. Additionally, registries provide *proofs* of correctness (e.g., a proof can be a digital signature generated by the registry). In a typical DID system, any entity can verify that a digital signature has been generated by the owner of a DID by following a two-step process: (a) retrieve the corresponding DID document, and (b) check if the digital signature can be verified with (one of) the assertion key(s) included in the DID document.

In this work, a new DID method is defined, and it is used to protect routing advertisements. The proposed DID method relies on ICN for managing the corresponding DID documents.

2.2. SDN and Bloom-Filter-Based Forwarding

Software-defined networking (SDN) [13] is a technology that enables the use of programmable network switches, which can be configured by a centralized entity, known as the "network controller" (or simply controller). A controller can use a protocol, such as OpenFlow [14], to configure switches with "rules" that are used for making switching decisions.

Reed et al. [6] leverage this property of SDN and implement source-routing using Bloom filters [15]. From a high-level perspective, this solution is implemented as follows. Firstly, it assumes that each outgoing interface of an SDN switch is identified by a bitstring identifier. Then, each sender can create a "forwarding identifier" using a Bloom filter constructed by ORing, the identifier of all interfaces through which a packet should be forwarded; this forwarding identifier is stored in the IPv6 address field of the transmitted packet. SDN switches are pre-configured with rules that allow them to forward these packets through the interfaces whose identifier is included in the forwarding identifier. An interesting property of this solution is that it can implement stateless multicast since a forwarding identifier may include a complete multicast delivery tree. Another useful property of this system is that given a forwarding identifier X for a path from a node A to a node B, and another forwarding identifier Y from the same node A to a node C, by ORing X and Y, we obtain a forwarding identifier that corresponds to a multicast delivery tree from A to B and C.

3. Related Work

Many related systems are trying to secure (inter-)networks by using cryptographic constructions (such as public keys) at the network layer (see, for example, [16–18]). These systems have broader goals than our approach; therefore, even though they can be used to achieve similar properties to our solution, they require significant changes to the networking infrastructure. Our solution is an overlay approach that can be used on top of any networking technology. Similarly, our constructions are applied in the edge devices, and they are transparent to the core network.

DIDs as a mechanism for securing routing messages have been investigated in the context of next-generation internet architectures. For example, the solution in [19] uses DIDs to protect the routing layer of information-centric architectures from poison attacks. These approaches use public keys as the corresponding DID, whereas our solution uses a registrar service to support URLs as DIDs. Additionally, our solution leverages an ICN-based DID registry to implement DID document management. Similarly, recent solutions investigate the use of DIDs for providing access control in the IoT (e.g., [20–22]). Although the goal of these solutions is different compared to our system, they follow the same principle: they

use DIDs to derive a key, which is then used for signing and verifying a digital signature. Nevertheless, these solutions use blockchain to manage trust. Our system uses as a root of trust a centralized entity. Although our approach introduces a single point of failure, it is more lightweight, and more realistic to deploy. Furthermore, given that IoT devices do not have the processing power to interact with the blockchain directly, all these solutions rely on a "proxy" that mediates the communication between the IoT devices and the blockchain: this proxy is also a single point of failure. The solution of Enge et al. [23] uses DIDs for establishing secure communication channels between IoT devices. This solution uses DIDs only for deriving public keys, whereas our system includes "constraints" that control how a key can be used. Furthermore, our solutions supports secure and controlled delegation.

Related technologies, such as Verifiable Credentials [24] Macaroons [25], Authorization capabilities for linked data (ZCAP-LD) [26] and capabilities as defined by the WAVE framework [27] can also be used for expressing "assignments" and "capabilities". The system proposed in this paper uses DIDs because are simpler to implement and use, and easier to verify. Nevertheless, if more complex trust relationships are required, these technologies can be used instead.

In the context of the IoT, related efforts try to provide security properties at the application layer, supporting, at the same time, advanced communication paradigms, such as publish–subscribe, and group communication (see, for example, group OSCORE [28]).These efforts are complementary to our approach: with our solution, we are securing the routing layer, but the used cryptographic material can be provided as an input to application-layer-security solutions.

4. Design

4.1. Entities and Interactions

The proposed system allows end-user *client* applications to interact with IoT devices using an application layer protocol, such as CoAP group communication [29]. IoT devices can be real devices, virtual devices (e.g., generated by a web of things IoT gateway), and cloud-based systems: the nature of an IoT device does affect our design. IoT devices are associated with one or more URLs, referred to as $Resource_{URL}$. Similarly, a $Resource_{URL}$ can be associated with one more IoT devices. IoT devices *advertise* their $Resource_{URL}$, and these advertisements are digitally signed. IoT devices are attached directly or indirectly to an *edge* node, which is identified by an *identifier*.

Our system assumes that $Resource_{URL}$ is of the form

$$schema : //domain/path$$

and considers a network of trusted *registrars* which are responsible for managing the *domain* part of $Resource_{URL}$. Domain *owners* register their domains with a *registrar*; this process is assumed to be secure in the sense that domains cannot be "hijacked" or "stolen", i.e., an attacker cannot impersonate a legitimate user to a registrar. We assume that there is a mechanism that allows registrars to verify domain ownership. Similarly, we assume that the public keys of the trusted registrars are well-known. Then, each *owner* can create an arbitrary number of $Resource_{URL}$ by appending paths or by creating sub-domains. An *owner* can *assign* a $Resource_{URL}$ to an IoT device, or even *delegate* it to a third party *controller*.

Figure 1 illustrates an example of $Resource_{URL}$ management. In this example, there is a domain called "/smart-city.iot". The domain owner generates three $Resource_{URL}$ identifiers and delegates them to three different *controllers*. Each controller uses the delegated $Resource_{URL}$ to create new URLs and assign them to the corresponding IoT device.

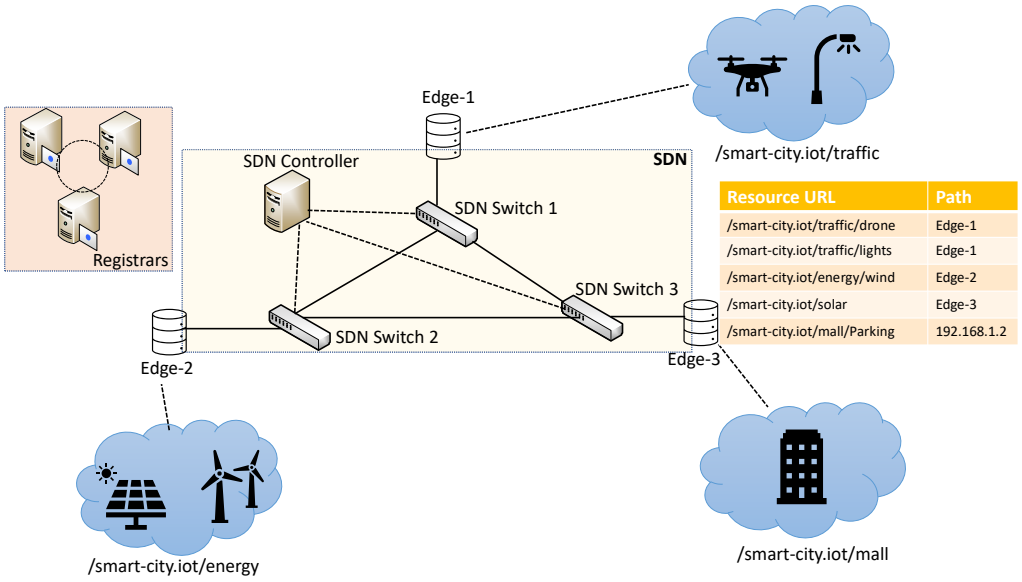

Figure 1. System overview.

4.2. Underlay Network Architecture

Our system builds on the Edge-ICN architecture defined in [5]. In particular, our system considers *edge* nodes interconnected using the SDN-based infrastructure described in Section 2.2. Each edge node is identified by an *identifier*, and each IoT device is attached to one of these edge nodes. Moreover, our system assumes that each edge node knows a "forwarding identifier" toward any other edge node. Finally, SDN switches are configured with a "special" forwarding identifier used for broadcasting messages; hence, we will refer to this forwarding identifier as the *broadcast forwarding identifier*. In our system, the core network is oblivious to the application-layer identifiers used by endpoints; therefore, the SDN controllers does not have to be aware of the $Resource_{URL}$s used by IoT endpoints and client applications.

4.3. The Use of Decentralized Identifiers

The proposed system leverages decentralized identifiers (DIDs) to implement secure URL management. In particular, a new DID method is defined that uses the ICN registry specified in [30]. This registry allows DID owners to store a DID document in a *storage node* in the network. Any entity can *read* that document by sending an ICN *subscription* message. The subscription message, which includes in its header the requested DID, is rooted by the ICN network to the appropriate node, which in return *publishes* the requested document alongside the corresponding document and authorization proofs.

As it is discussed in the following section, a DID in our system is associated with a $Resource_{URL}$ domain, and *domain* is used as our DID method identifier, e.g.,

$$did : domain : building1.iot$$

A DID in the proposed system is associated with a DID document, which is a JSON-encoded data structure that includes two claims:

- *id*: The DID which the DID document concerns.
- *assertion*: A list of public keys that can be used for verifying the digital signatures included in the corresponding *advertisements*. Each entry in the assertion list includes the following claims:
  - *id*: An identifier which is unique for the assertion list entries.
  - *prefix*: A URL prefix for which the key can be used for singing advertisements.
  - *expires*: A timestamp after which the defined public key cannot be used for signing advertisements.

- *edge*: The identifier of the edge node in which the IoT device is attached.
- *type*: The type of the public key.
- *publicKey*: The public key whose encoding depends on the type.

The type of an element of the assertion list can be either *url* or *JsonWebKey*. In the former case, the *publicKey* is a pointer to a key defined in another document, and the *publicKey* is a JSON web key (JWK) [31].

Listing 1 is an example of a DID document used in our system for the DID "did:domain:smart-city.iot"

Listing 1. The DID document for the DID "did:domain:smart-city.iot".

```
{
  ''id'': ''did:domain:smart−city.iot'',
  ''assertion'': [{
    ''id'': ''#key1'',
    ''expires'': ''1651072705'',
    ''prefix'': ''smart−city.iot/'',
    ''edge'':''edge−1'',
    ''type'': ''JsonWebKey'',
    ''publicKey'': {
      ''crv'': ''Ed25519'',
      ''x'': ''7a345vc...'',
      ''kty'': ''OKP''
    },
    {
    ''id'': ''#key2'',
    ''expires'': ''1651072710'',
    ''prefix'': ''smart−city.iot/building1'',
    ''edge'':''edge−2'',
    ''type'': ''url'',
    ''publicKey'':''did:domain:building1.iot''
    }
  ]
}
```

A DID document is protected by a *document proof* and an *authorization proof*. The *document proof* is used for verifying the DID document's integrity. The *authorization proof* is used for learning the public key of the DID *owner*. A *document proof* is a compact encoded JSON web signature (JWS) [32], and its payload includes the following claims:

- `created`: A timestamp indicating the creation time of the proof.
- `expires`: A timestamp indicating the expiration time of the proof.
- `sha-256`: The base64url encoded SHA-256 hash of the DID document.

The signature of the proof is generated using EdDSA and the private key of the DID owner.

The *authorization proof* is also a compact encoded JWS, and its payload includes the following claims:

- `id`: The DID.
- `created`: A timestamp indicating the creation time of the proof.
- `expires`: A timestamp indicating the expiration time of the proof.
- `controller`: The public key of the DID owner.

The signature of an authorization proof is generated by the registrar.

Given a DID and the corresponding DID document, any entity can validate its correctness by executing the following steps:

1. Verify that the DID is included in the `id` claim of the DID document.

2. Verify that the document proof has not expired, it includes DID in the `id` claim, and it includes the correct value in the `sha-256` claim.
3. Verify that the authorization proof has not expired and includes DID in the `id` claim.
4. Verify that the signature of the authorization proof has been generated by a trusted registrar and validate it.
5. Validate the signature of the document proof using the public key located in the `controller` claim of the the authorization proof.

4.3.1. Domain Registration

Each domain owner should interact with a registrar in order to register a domain that they own. The registration request includes also a public key, owned by the domain owner. The outcome of the registration process is the authorization proof, which includes the `id` claim, an id of the form $did :< domain >$, and in the `controller` claim, the public key sent by the owner. The proof is signed by the registrar. Since (as we already discussed) the public keys of the trusted registrars are well-known, anybody can verify the validity of the latter signature.

4.3.2. Resource URL Assignment

A domain owner can create a $Resource_{URL}$ and assign it to an IoT device. The IoT device will "advertise" this $Resource_{URL}$ and will sign these advertisements using its private key. $Resource_{URL}$ assignment is simply implemented by adding a new entry in the assertion list that includes the public key of the IoT device.

As discussed in Section 4.2, the only cryptographic operation that IoT devices have to perform is to generate a digital signature every time they send an advertisement message. There can be cases where an IoT device is not powerful enough to perform this operation. In these cases, the advertisement process can be delegated to another entity or even to the edge node. This can be trivially achieved by including the public key of that node in the corresponding entry of the assertion list.

4.3.3. Resource URL Delegation

There can be cases where a domain owner would like to delegate the administration of a $Resource_{URL}$ to a third-party *controller*. Compared to the assignment process, discussed in the previous section, with the delegation process, the public key used by the controller is defined in a DID document owned by the controller. Therefore, a controller can generate new $Resource_{URL}$ identifiers (prefixed with the delegated $Resource_{URL}$), and assign them to IoT devices that it controls. The delegation process is implemented by generating a new item in the assertion list of type "url" and by setting the `publicKey` claim of that element equal to the DID of the controller.

4.4. Resource URL Advertisement

Resource URL advertisement extends the advertisement process of the underlying architecture (defined in [5]) to include a verification process. Resource URL advertisement in [5] is implemented as follows. Each IoT device *advertises* its $Resource_{URL}$ to the edge node where it is attached, and each edge node forwards this advertisement to the SDN network using the *broadcast forwarding identifier*. Therefore, all advertisements are eventually received by all edge nodes. Using these advertisements, edge nodes construct *routing tables* that map a $Resource_{URL}$ to a set of (local) IoT devices, as well as to a set of (remote) edge nodes. Therefore, upon receiving a request for a $Resource_{URL}$, an edge node may forward it to one or more locally attached IoT devices and/or to one or more edge nodes (using the appropriate forwarding identifier). Advertisement messages include the identifier of the edge node that can be used for reaching the corresponding IoT device as well as a nonce used for preventing replay attacks.

The additional verification step is implemented as follows. Upon receiving an advertisement, an edge node needs to validate it. In order to do this, it uses the registry operations

defined in [30] to receive the appropriate DID document(s). An edge node may cache these documents. An example of this process is illustrated in Figure 2. In this example, an edge node receives an advertisement for "smart-city.iot/mall", then it requests the ICN network to retrieve the DID document for "did:domain:smart-city.iot"; in the retrieved document, it performs a closer prefix match with the prefixes included in the assertion list, and selects the appropriate entry. In this example, the entry for "smart-city.iot/mall" is of the type URL, so the edge node requests the DID document for that URL and locates the desired key. Finally, the edge node verifies the digital signature of the advertisement using the key extracted from the DID document.If all validations succeed, the edge node updates its routing table accordingly. Therefore, advertisements are handled by edge nodes, and they do not reach IoT devices; hence, IoT devices do not have to perform any DID validation.

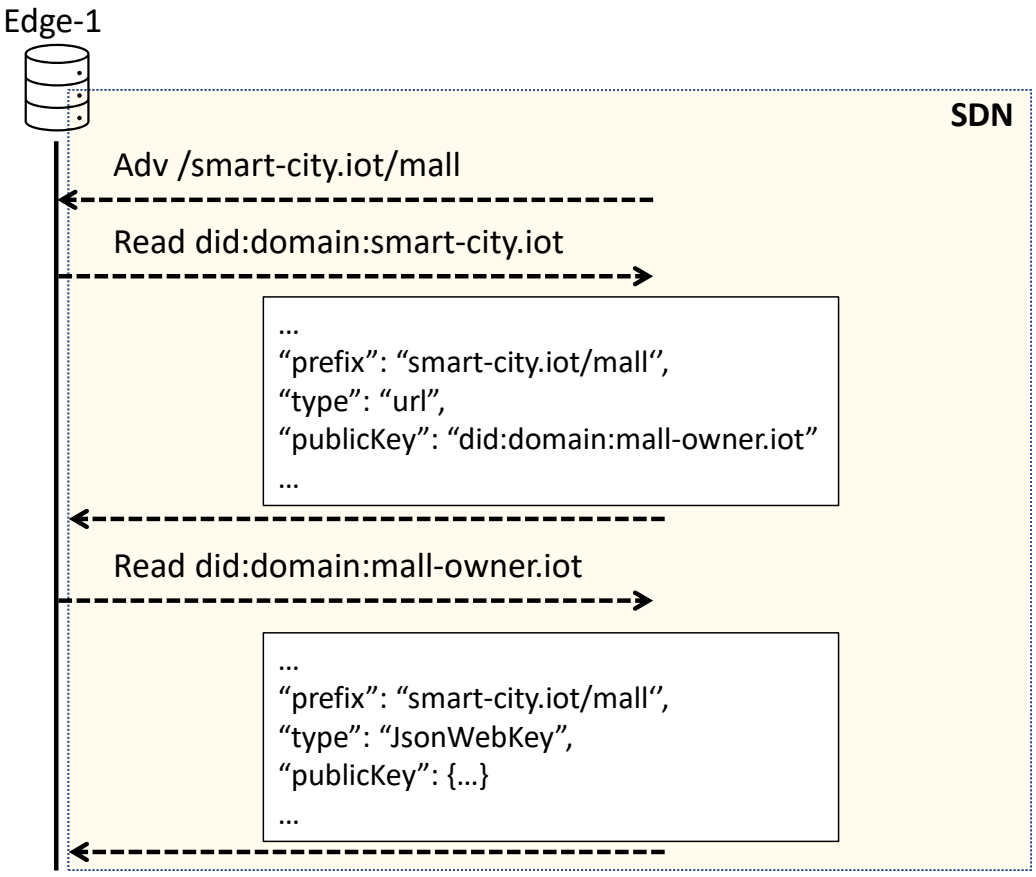

Figure 2. Assertion key resolution.

5. Implementation and Evaluation

In order to validate the proposed solution, we emulated an SDN network using the mininet network emulator [33]. We used open vSwitch programmable switches [34], which we configured with the appropriate rules for performing Bloom-filter-based switching using the POX network controller [35]. We implemented *did:domain* using the jwcrypto (https://jwcrypto.readthedocs.io/en/latest/) (accessed on 6 October 2022) to generate the required JSON objects as well as to generate and verify the required JWS signatures.

5.1. Evaluation Scenario

The proposed solution was evaluated through a smart city emulation scenario. In this scenario, the owner of the domain "smart-city.iot" delegates to 100 building owners the $Resource_{URL}$ "smart-city.iot/building-X", where X is a number between 1 and 100. Then, each building owner can configure the IoT devices of each smart building to advertise resources using the delegated URL, e.g., the owner of "building-1" could configure a light controller to advertise the URL "smart-city.iot/building-1/lights".

In our evaluation scenario, we consider the topology depicted in Figure 3. In particular, each building is attached to a different edge node, and each edge node is attached to a different SDN switch. We group SDN switches in two groups, and we connect all SDN switches of the same group to another SDN switch. Finally, the latter two SDN switches are connected using a *backbone* link. In our evaluation scenario, each building advertises 50 different $Resource_{URL}$s every 10 s and a DID document every 1 min. Advertisements take place almost simultaneously. Table 1 summarizes the parameters used in our evaluation scenario.

Table 1. Evaluation scenario parameters.

Parameter	Value
Number of buildings	100
Number of edge nodes	100
$Resource_{URL}$ per building	50
Interval between content advertisements	10 s
Interval between DID document advertisements	1 min

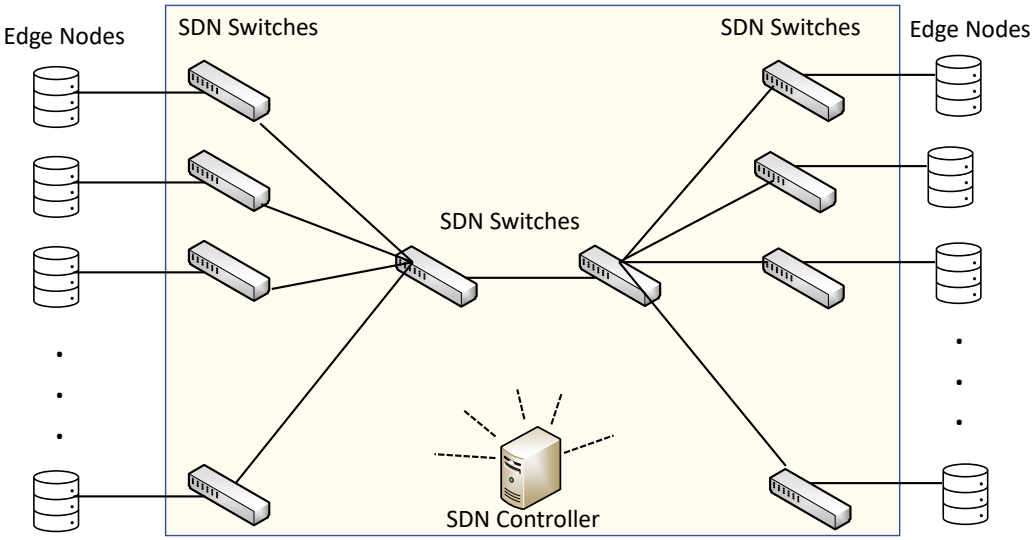

Figure 3. Evaluation topology. The links between the SDN switches and the controller are not shown for clarity reasons.

As a baseline for comparison of our system, we consider the trust schema defined in [36]. This trust schema is used by a popular ICN architecture known as named data networking (NDN) [37], and it resembles WebPKI. In particular, this schema defines "trust anchors", which are public keys used to sign digital certificates. Additionally, each (edge) router is configured with rules that map prefixes of $Resource_{URL}$ to identifiers which are resolved to a digital certificate: this certificate includes a public key that can be used for validating the digital signature included in the corresponding $Resource_{URL}$ advertisement. Therefore, compared to our solution, a trust anchor has the same role as the registrar, the identifier of a digital certificate is equivalent to a DID, and a digital certificate is equivalent to a DID document. Similarly to our solution, the solution of [36] can be used for creating "trust chains". In the considered use case, a trust anchor would sign a digital certificate for the domain "smart-city.iot", and the private key that corresponds to the key included in the latter certificate would be used to sign the digital certificates of each building.

5.2. Routing State Storage and Computational Overhead

Each edge node maintains a routing table that maps a $Resource_{URL}$ to either an edge node identifier or to an IP address: in the former case, the IoT device that has advertised the corresponding $Resource_{URL}$ is attached to another edge node, whereas in the latter case, the

device that has advertised the $Resource_{URL}$ is attached to the local node. In order to decrease the storage overhead, entries that share the same prefix and have the same destination edge node are aggregated. For example, the entries ["smart-city.iot/building-1/lights", "edge-10"] and ["smart-city.iot/building-1/power", "edge-10"] will be aggregated to ["smart-city.iot/building-1/", "edge-10"]. Using this approach, each edge node in the considered evaluation scenario maintains 149 entries: 99 entries that correspond to the smart buildings attached to other edge nodes, and an additional 50 entries for the URLs of the smart building attached to that edge node. These entries are due to the ICN functionality of our system. In addition, the edge router should maintain routing entries used for locating DID documents. In particular, each building is associated with a different *did:domain* DID, and hence, each router should maintain another 100 records. This state is due to the ICN registry functionality of our system.

In a system built using the solution defined in [36] a routing entry per digital certificate should be used. Therefore, such a system requires the same routing entries as in our solution. However, in addition to the routing entries, the solution of [36] requires each router to be configured with rules that map prefixes to digital certificate identifiers. Since in the considered use case there are 100 buildings and each building uses its own key to sign advertisements, each router should be configured with 100 additional rules. It should be highlighted that in addition to the extra routing state, the solution defined in [36] creates additional administrative overhead since every time a digital certificate changes (which means its identifier also changes), the corresponding rules in the routers must be updated.

Routing in our system is implemented using *longest prefix match* at the edge routers and Bloom-filter-based forwarding in the core network. Using a solution such as the one proposed in [38], an ordinary PC can perform a few millions lookups per second in a routing table that contains 3,000,000 entries. Similarly, the solution presented in [6], which is used in the core of our architecture for implementing Bloom-filter-based forwarding, requires switches from SDN to perform just an OR operation per outgoing link.

5.3. Communication Overhead

Our solution introduces the communication overhead for retrieving DID documents. In the considered scenario, every time an edge node receives an advertisement, it requests the network to retrieve two DID documents: the DID document that corresponds to "smart-city.iot", and the DID document that corresponds to the building owner. It is reminded, however, that the underlying ICN architecture allows edge nodes to construct multicast delivery trees; hence, multiple requests concerning the same item can be satisfied using a single multicast delivery. However, this requires that the requests arrive almost simultaneously to the edge node. Since we know that an advertisement will trigger requests for DID documents, we evaluate a strategy based on which an edge node adds some delay before responding to a DID document request, hoping that during that period, more requests concerning the same DID will arrive. In order to evaluate the impact of this strategy, we consider that only two buildings, one in each side of the backbone link, advertise $Resource_{URL}$s (every 10 s), and we measure the number of DID documents that are transferred through the backbone link in one minute. We consider three strategies: edge nodes add no delay, edge nodes add 2 ms delay, and edge nodes add 4 ms delay. The obtained results can be seen in Figure 4. As it can be observed, by adding 4 ms delay, all requests for DID documents arrive at edge nodes; therefore, they can all be satisfied by a single multicast transmission. As a result, only two DID documents are transferred over the backbone link.

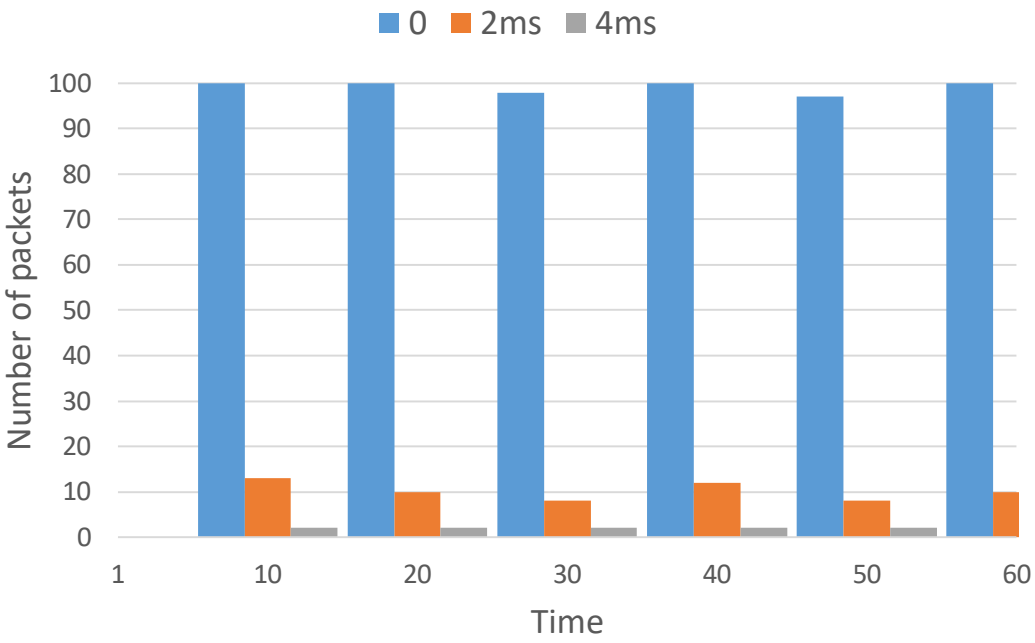

Figure 4. Number of DID documents transferred over the backbone link if 0, 2, or 4 ms delay is added before responding to DID document request.

Similarly, in a system that uses the solution of [36], an advertisement triggers two requests: a request for retrieving the certificate that corresponds to "smart-city.iot" and a request for retrieving the certificate of the building. Therefore, both solutions result in the same communication overhead. Additionally, the delay-based mechanism can also be applied for the solution of [36].

5.4. DID Document Storage Overhead

The DID document that corresponds to "smart-city.iot" includes 100 entries in the assertion list, one for each building, and each entry is of the URL type. An example of such an entry can be seen in Listing 2. The size of this document is 17,452 bytes.

Listing 2. An assertion entry for the DID "did:domain:smart-city.iot".

```
1
2  {
3     ''id'': ''#key2'',
4     ''expires'': ''1651072710'',
5     ''prefix'': ''smart−city.iot/building1'',
6     ''edge'':''edge−2'',
7     ''type'': ''url'',
8     ''publicKey'':''did:domain:building1.iot''
9  }
```

Each building owner maintains also a DID document that includes 50 entries in the assertion list, one for each advertised URL. Each entry is of the JsonWebKey type. The size of each such document is 13,900 bytes.

A digital certificate in a system that uses the solution of [36] includes only a public key; hence, its size is smaller compared to a DID document (in our use case $\approx$ 500 bytes). This happens because the information included in a DID document in our system is distributed as rules configured in routers in the solution of [36]. On the other hand, an entity authorized for multiple prefixes should store a certificate per prefix in the solution of [36], whereas in our solution, the same DID document may include a key that can be used for many prefixes.

5.5. Computational Overhead

The only operation that each IoT device has to perform is signing an advertisement message. In a RaspberryPi 2 Model B Rev 1.1 with a 900 MHz quad-core ARM Cortex-A7 CPU and 1 GB RAM, the generation of a signature using EdDSA requires 10.1 ms. Similarly, signature generation in an Espressif ESP32 WROOM-32 IoT device (240 MHz dual-core Xtensa LX7 CPU) requires 160 ms. EdDSA signatures are used in our system for providing authentication of the advertisement messages. In any case, our solution is not tightly bound to a particular authentication mechanism: other authentication mechanisms that may be more suitable for constrained devices may be considered and integrated in our approach. For example, many systems rely on MACs and pre-shared keys for providing authentication: such an approach could also be used in our system, e.g., by including a 'hint' of the used pre-shared key in the DID document. However, such approaches are left as future work.

Similarly, an edge node in our system has only to verify a digital signature. Using an Ubuntu 22.04 machine equipped with an intel i7-3770 CPU, 3.40 GHz and 8 GB of RAM, digital signature verification requires less than 1 ms.

The solution of [36] uses the same signing procedure; hence, its computational overhead is the same as that in our system.

5.6. Security Evaluation

Providing that the registrar service is secure, our solution has the following security properties:

The integrity and the authenticity of advertisements are protected. $Resource_{URL}$ advertisements are digitally signed by the IoT devices. The public key of the IoT device is included in the DID document, whose integrity is protected by the document proof generated by the DID controller. Therefore, any entity can verify the advertisement signatures. Additionally, through the authorization proof, any entity can verify that a particular IoT device is *authorized* to advertise this specific $Resource_{URL}$.

Our solution is resilient to IoT devices' key breaches. A breached IoT device key can be used for generating fake advertisements for the $Resource_{URL}$ for which the IoT device has been authorized, until the corresponding DID document expires. The impact of this attack is further decreased by including in the document the identifier of the edge node in which the IoT device is attached (it is reminded that only the DID owner can modify a DID document). Then, each edge device can check whether the included identifier matches its own identifier. With this approach, fake advertisements can only be sent from the same location where the legitimate IoT device is located, limiting, in this way, the impact of this attack to a few more valid advertisement messages.

Dependence on the trusted registrar service is minimal. Domain owners rely on the registrar service in order to receive the authorization proof. When this step is completed, domain owners can manage, assign and delegate $Resource_{URL}$ without relying on the registrars. For example, a domain owner can freely modify the assertion key of a DID document, as opposed, for example, to a solution based on WebPKI, where an owner would require a new "digital certificate".

5.7. Performance–Security Trade-Offs

As discussed in Section 4.4, whenever an edge node retrieves an advertisement, it has to request the ICN network to retrieve the corresponding DID documents. In order to decrease the number of such requests, as well decreasing the advertisement verification time, these documents can be cached. Nevertheless, this means that an edge node may not have the current version of a DID document. A DID document is updated whenever a new entry is added in the assertion list or whenever an existing entry is removed. In the former case, the edge node will not be able to locate the key used for signing the advertisement in the cached documents; hence, it may refresh its cache. However, in the latter case, the edge node will consider a signature generated with a *revoked* key to be valid. Therefore, there is

a time window between the moment that a key is revoked and the moment that a cache entry expires during which a revoked key can be used. A solution to this problem that can be explored in future work is to leverage the ICN functionality and each edge node to "subscribe" for all future versions of a DID document; this way, every time a DID document is created, it will be pushed to the edge nodes that have subscribed to its updates.

6. Discussion

6.1. Alternative ICN Underlays

Our solution is applied to the ICN-based architecture defined in [5] but it can also be applied in other ICN-based solutions. We now examine how our solution can be used with two other ICN systems.

Named-data networking (NDN) [37] is probably the most popular ICN architecture. The main difference between NDN and our underlying architecture is that in NDN, all routers participate in the ICN network as opposed to our underlay architecture, where only edge routers are aware of the ICN functionality. This means that in NDN, all routers handle advertisements and maintain routing state. NDN uses the solution defined in [36] to protect against fake advertisements. Since this solution requires the definition of "rules" in each router, it is applied only in edge routers (in order to decrease administrative overhead). Our solution does not require any configuration in routers; hence, if our solution was used in NDN, all routers could potentially verify advertisements.

The publish–subscribe internet architecture (PSI) [39] is an alternative ICN-based system. PSI assumes an overlay "rendezvous" system which is responsible for handling advertisements, as well as for responding to content-lookup requests (hence, this rendezvous system is similar to DNS). PSI considers that each rendezvous node is responsible for handling a particular portion of the name space as well as "controlling" which entities can advertise that portion. In other words, each rendezvous node can hold the role of a registrar as defined in our system. Hence, our solution fits naturally to PSI.

6.2. Alternative DID Methods

In our system, we defined our own DID method. In this section, we discuss how our DID method compares to other similar methods that could have been considered in a solution similar to ours.

did:web [40] is a DID method that stores DID documents in web servers accessed over HTTPS. The location of the web server is defined in the DID itself; for example, the DID document that corresponds to "did:web:example.com" can be located at "https://example.com" Our DID method uses a similar DID document resolution and it relies on the same security primitives (did:web requires a valid HTTPS certificate, our method requires a valid authorization proof). The main difference between the two methods is that the did:web method does not include the "prefix" property for assertion keys, and therefore, it is not straightforward as to how to delegate $Resource_{URL}$ prefixes to other entities.

did:key [41] is a simple DID method where DID documents are implicit. In particular, a DID in did:key is the encoding of a public key (e.g., using an encoding such as base-58). Then, this key is used for all verification methods. Using did:key implies that content name prefixes are also public keys (hence, they are not human-readable). Additionally, because DID documents are implied, delegation cannot be implemented. On the other hand, did:key does not require a trusted registrar.

did:self [19] is a DID method that also uses public keys as DIDs. The private key that corresponds to a DID is used for signing the corresponding DID document. Therefore, did:self can achieve delegation. Moreover, similar to did:key, it does not require a registrar. However, did:self does not support human-readable names. Moreover, in case the private key that corresponds to a DID is breached or lost, the corresponding DID cannot be used any more; therefore, if that DID is used as a prefix, all content items must be updated to use a new prefix.

6.3. Use of Constrained Devices

Although our considered use case is IoT-based, our solution is not designed with constrained devices in mind. Nevertheless, we believe that it should be possible to apply our solution even in these devices. EdDSA signatures can be implemented even in devices with limited capabilities. As Kortesniemi et al. [11] reported, the EdDSA signature can be performed in 14 million cycles on an 8-bit device, while on a 32-bit low-cost ARM Cortex-M0 core, the same operation undergoes 3.6 million cycles. Therefore, Cortex-M0 devices, which are available for less than half a dollar in large quantities and run at up to 48 MHz, can perform up to 13 EdDSA signatures per second. Similarly, the power required to generate an EdDSA signature in a Cortex-M0 processor is 20–34 μJ [11].

7. Conclusions

In this paper, a solution that leverages decentralized identifiers for secure application-layer routing is proposed. The proposed solution can be easily integrated into existing approaches since it does not require any modification to the SDN infrastructure. The proposed solution allows URLs to be used as DIDs by adding a trusted name registration service; the use of URLs as DIDs enables more user-friendly application-layer solutions, as well as greater interoperability with legacy systems. The proposed solution is lightweight since it requires only few ms to perform the necessary cryptographic operations even in IoT devices; it adds only a few bytes to the routing advertisement messages; it creates minimal additional routing states; it does not need auxiliary information to be stored per verifying entity; and it can be easily managed. Finally, the proposed solution has intriguing security properties: it protects the integrity and authenticity of advertisements, it is resilient to key breaches, and its dependence on a trust registrar is minimal.

Future work in this area includes solutions for fast revocation so that key breaches can be handled in a secure and fast manner, as well as tools for automating the process of domain registration. Furthermore, the use of verifiable credentials as well as the application of different DID methods and ICN underlays will be further investigated.

Author Contributions: Writing—original draft preparation, K.A. and N.F.; writing—review and editing, B.A. and A.A.; supervision, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia, under grant no. (RG-13-611-42). The authors, therefore, acknowledge with thanks DSR technical and financial support.

Data Availability Statement: Not applicable, the study does not report any data.

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

References

1. 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.* **2014**, *16*, 1024–1049.
2. Nour, B.; Sharif, K.; Li, F.; Biswas, S.; Moungla, H.; Guizani, M.; Wang, Y. A survey of Internet of Things communication using ICN: A use case perspective. *Comput. Commun.* **2019**, *142*, 95–123.
3. Ghali, C.; Tsudik, G.; Uzun, E. Needle in a haystack: Mitigating content poisoning in named-data networking. In Proceedings of the NDSS Workshop on Security of Emerging Networking Technologies (SENT), San Diego, CA, USA, 23 February 2014. Available online: https://www.ndss-symposium.org/ndss2014/workshop-security-emerging-networking-technologies-sent-2014-programme/ (accessed on 6 October 2022).
4. Wang, J.; Wei, X.; Fan, J.; Duan, Q.; Liu, J.; Wang, Y. Request pattern change-based cache pollution attack detection and defense in edge computing. *Digit. Commun. Netw.* 2022, *in Press*.
5. Fotiou, N.; Siris, V.A.; Xylomenos, G.; Polyzos, G.C.; Katsaros, K.V.; Petropoulos, G. Edge-ICN and its application to the Internet of Things. In Proceedings of the 2017 IFIP Networking Conference (IFIP Networking) and Workshops, Stockholm, Sweden, 12–16 June 2017; pp. 1–6. https://doi.org/10.23919/IFIPNetworking.2017.8264880.
6. Reed, M.J.; Al-Naday, M.; Thomos, N.; Trossen, D.; Petropoulos, G.; Spirou, S. Stateless multicast switching in software defined networks. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–7. https://doi.org/10.1109/ICC.2016.7511036.

7. Hughes, A.; Sporny, M.; Reed, D. Decentralized Identifiers (DIDs) v1.0. Draft Community Group Report, W3C. 2021. Available online: https://w3c-ccg.github.io/did-primer/ (accessed on 6 October 2022).
8. Fedrecheski, G.; Rabaey, J.M.; Costa, L.C.P.; Calcina Ccori, P.C.; Pereira, W.T.; Zuffo, M.K. Self-Sovereign Identity for IoT environments: A Perspective. In Proceedings of the 2020 Global Internet of Things Summit (GIoTS), Dublin, Ireland, 3 June 2020; pp. 1–6.
9. Sporny, M.; Longley, D.; Sabadello, M.; Reed, D.; Steele, O.; Allen, C. Decentralized Identifiers (DIDs) v1.0. W3C Proposed Recommendation, W3C. 2021. Available online: https://www.w3.org/TR/did-core/ (accessed on 6 October 2022).
10. Ansey, R.; Kempf, J.; Berzin, O.; Xi, C.; Sheikh, I. Gnomon: Decentralized Identifiers for Securing 5G IoT Device Registration and Software Update. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
11. Kortesniemi, Y.; Lagutin, D.; Elo, T.; Fotiou, N. Improving the privacy of iot with decentralised identifiers (dids). *J. Comput. Netw. Commun.* **2019**, *2019*, 8706760.
12. Terzi, S.; Savvaidis, C.; Votis, K.; Tzovaras, D.; Stamelos, I. Securing Emission Data of Smart Vehicles with Blockchain and Self-Sovereign Identities. In Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, 2–6 November 2020; pp. 462–469.
13. Xia, W.; Wen, Y.; Foh, C.H.; Niyato, D.; Xie, H. A Survey on Software-Defined Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 27–51.
14. Lara, A.; Kolasani, A.; Ramamurthy, B. Network Innovation using OpenFlow: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 493–512. https://doi.org/10.1109/SURV.2013.081313.00105.
15. Bloom, B.H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **1970**, *13*, 422–426.
16. Aura, T. Cryptographically Generated Addresses (CGA). RFC 3972, IETF. 2005. Available online: https://www.ietf.org/rfc/rfc3972.txt (accessed on 6 October 2022).
17. Andersen, D.G.; Balakrishnan, H.; Feamster, N.; Koponen, T.; Moon, D.; Shenker, S. Accountable Internet Protocol (Aip). *Sigcomm Comput. Commun. Rev.* **2008**, *38*, 339–350.
18. Raychaudhuri, D.; Nagaraja, K.; Venkataramani, A. Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet. *ACM Sigmobile Mob. Comput. Commun. Rev.* **2012**, *16*, 2–13.
19. Fotiou, N.; Thomas, Y.; Siris, V.A.; Xylomenos, G.; Polyzos, G.C. Securing Named Data Networking routing using Decentralized Identifiers. In Proceedings of the 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), Paris, France, 7–10 June 2021; pp. 1–6.
20. Figueroa-Lorenzo, S.; Añorga Benito, J.; Arrizabalaga, S. Modbus access control system based on SSI over hyperledger fabric blockchain. *Sensors* **2021**, *21*, 5438.
21. Saidi, H.; Labraoui, N.; Ari, A.A.A.; Maglaras, L.A.; Emati, J.H.M. DSMAC: Privacy-aware Decentralized Self-Management of data Access Control based on blockchain for health data. *IEEE Access* **2022**, *10*, 101011–101028.
22. Luecking, M.; Fries, C.; Lamberti, R.; Stork, W. Decentralized identity and trust management framework for Internet of Things. In Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Toronto, ON, Canada, 2–6 May 2020; pp. 1–9.
23. Enge, A.H.; Satybaldy, A.; Nowostawski, M. An architectural framework for enabling secure decentralized P2P messaging using DIDComm and Bluetooth Low Energy. In Proceedings of the 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 27 June–1 July 2022; pp. 1579–1586.
24. Sporny, M.; Longley, D.; Chadwick, D. Verifiable Credentials Data Model 1.0. W3C Recommendation, W3C. 2019. Available online: https://www.w3.org/TR/verifiable-claims-data-model/ (accessed on 6 October 2022).
25. Birgisson, A.; Politz, J.G.; Erlingsson, Ú.; Taly, A.; Vrable, M.; Lentczner, M. Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud. Network and Distributed System Security Symposium. 2014. Available online: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41892.pdf (accessed on 6 October 2022).
26. Webber, C.L.; Sporny, M.; Eds. Authorization Capabilities for Linked Data. Editor's Draft, W3C. 2020. Available online: https://w3c-ccg.github.io/zcap-spec/ (accessed on 6 October 2022).
27. Andersen, M.P.; Kumar, S.; AbdelBaky, M.; Fierro, G.; Kolb, J.; Kim, H.S.; Culler, D.E.; Popa, R.A. WAVE: A Decentralized Authorization Framework with Transitive Delegation. In Proceedings of the 28th USENIX Conference on Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; USENIX Association: Santa Clara, CA, USA, 2019; pp. 1375–1392.
28. Tiloca, M.; Selander, G.; Palombini, F.; Mattsson, J.; Park, J. Group OSCORE—Secure Group Communication for CoAP. RFC-Draft, IETF. 2021. Available online: https://www.ietf.org/id/draft-ietf-core-oscore-groupcomm-16.html (accessed on 6 October 2022).
29. Rahman, A.; Dijk, E. Group Communication for the Constrained Application Protocol (CoAP). RFC 7390, IETF. 2014. Available online: https://datatracker.ietf.org/doc/html/rfc7390 (accessed on 6 October 2022).
30. Alzahrani, B. An Information-Centric Networking Based Registry for Decentralized Identifiers and Verifiable Credentials. *IEEE Access* **2020**, *8*, 137198–137208.
31. Jones, M. JSON Web Key (JWK). RFC 7517, IETF. 2015. Available online: https://www.rfc-editor.org/rfc/rfc7517 (accessed on 6 October 2022).
32. Jones, M.; Bradley, J.; Sakimura, N. JSON Web Signature (JWS). RFC 7515, IETF. 2015. Available online: https://www.rfc-editor.org/rfc/rfc7515 (accessed on 6 October 2022).

33. Lantz, B.; Heller, B.; McKeown, N. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Atlanta, GA, USA, 9–10 November 2010; ACM: New York, NY, USA, 2010; pp. 19:1–19:6.
34. Pfaff, B.; Pettit, J.; Koponen, T.; Jackson, E.; Zhou, A.; Rajahalme, J.; Gross, J.; Wang, A.; Stringer, J.; Shelar, P.; et al. The Design and Implementation of Open vSwitch. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA, USA, 4–6 May 2015; USENIX Association: Oakland, CA, USA, 2015; pp. 117–130.
35. Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. NOX: Towards an Operating System for Networks. *Sigcomm Comput. Commun. Rev.* **2008**, *38*, 105–110.
36. Yu, Y.; Afanasyev, A.; Clark, D.; Claffy, K.; Jacobson, V.; Zhang, L. Schematizing Trust in Named Data Networking. In Proceedings of the 2nd ACM Conference on Information-Centric Networking, Macao, China, 24–26 September 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 177–186.
37. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named Data Networking. *Sigcomm Comput. Commun. Rev.* **2014**, *44*, 66–73.
38. Wang, Y.; He, K.; Dai, H.; Meng, W.; Jiang, J.; Liu, B.; Chen, Y. Scalable name lookup in NDN using effective name component encoding. In Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems, Macau, China, 18–21 June 2012; pp. 688–697.
39. Xylomenos, G.; Vasilakos, X.; Tsilopoulos, C.; Siris, V.A.; Polyzos, G.C. Caching and mobility support in a publish-subscribe internet architecture. *IEEE Commun. Mag.* **2012**, *50*, 52–58.
40. Zagidulin, D.; Terbu, O.; Guy, A. did:web Method Specification. Technical Report. 2020. Available online: https://w3c-ccg.github.io/did-method-web/ (accessed on 6 October 2022).
41. Longley, D.; Zagidulin, D.; Sporny, M. The did:key Method. Technical Report. 2020. Available online: https://w3c-ccg.github.io/did-method-key/ (accessed on 6 October 2022).