

# Facilitating Data Usage Control Through IPv6 Extension Headers

Haydar Qarawlus<sup>1</sup><sup>a</sup>, Malte Hellmeier<sup>1</sup><sup>b</sup> and Falk Howar<sup>1,2</sup><sup>c</sup>

<sup>1</sup>*Fraunhofer ISST, Dortmund, Germany*

<sup>2</sup>*TU Dortmund University, Dortmund, Germany*

**Keywords:** Data Usage Control, Data Privacy, Data Sovereignty, IPv6, Extension Headers, Hop-by-Hop Options, Destination Options.

**Abstract:** Data ownership and privacy control are gaining increasing attention and relevance. More and more aspects of enforcement methods for data ownership, data usage control, and data access control are being researched to find suitable solutions and practical applications. This includes technical methods, standards, and reference architectures for controlling data after it has been shared, which are frequently discussed under the term data sovereignty. However, enforcing data sovereignty is still challenging, and most solutions only work in trusted environments or are focused on the application layer. In this paper, we focus on policy enforcement mechanisms to facilitate data sovereignty on a lower network layer. We explore the use of steganography concepts and IPv6 extension headers in data usage control as an additional usage control measure. We propose the use of IPv6 Hop-by-Hop Options and Destination Options Extension Headers, including possible implementation methods and a prototype setup. We tested our work in multiple scenarios to examine performance and applicability. The results highlight the numerous benefits of our proposal with minimal drawbacks.


## 1 INTRODUCTION


The value of data across various industries is continuously increasing, with more industries depending on data for their operations (Zrenner et al., 2019). However, this value increase raises the risks of data misuse (Jung and Dörr, 2022). The increased risks reduce trust among the parties sharing data, leading to slowdowns in data sharing to increase caution. To counter this, much research is being done in this area, including topics such as data ownership and usage (Otto et al., 2022). Much of the research focuses on data sovereignty and its effects on increasing trust. An increase in trust can be achieved through, for example, the design of specific high-level protocols and frameworks for data sharing. Frameworks like the International Data Spaces (IDS)<sup>1</sup> create environments for data sharing under conditions controlled by the data owners, enforced through technical applications such as IDS Connectors operating in user-space at each end. Although user-space applications, such as IDS


Connectors, are essential to increase trust, a significant research question remains open with respect to facilitating transparency, third-party verification, and possible enforcement. At the time of writing, the enforcement of the data usage policies defined by the owners occurs on the end systems and, in some cases, within secure environments (Wagner et al., 2018). To our knowledge, there is no current research that explores the embedding and enforcement of data usage policies at different stages and locations other than the end systems. We address this through the following research questions (RQs):

1. Can data usage control concepts be applied at the network protocol level?
2. What are possible implementation methods for network-based data usage policy enforcement?
3. Which security, technical, and performance aspects must be considered?

In this work, we attempt to answer the research questions through the use of a known and well-defined medium, namely Internet Protocol Version 6 (IPv6) Extension Headers. We focus on examining the compatibility of the IPv6 Hop-by-Hop Options and Destination Options extension headers for carrying relevant data usage policy data. We define a format for the

<sup>a</sup> <https://orcid.org/0000-0003-3248-1163>

<sup>b</sup> <https://orcid.org/0000-0002-2095-662X>

<sup>c</sup> <https://orcid.org/0000-0002-9524-4459>

<sup>1</sup><https://internationaldataspaces.org>, last accessed: March 2nd, 2025

carried policies and examine how it can be enforced during transport.

Our paper is organized as follows: In Section 2, we provide background information and related work to the topics addressed in this paper. Next, we highlight our proposed approach for using extension headers in Section 3. We provide a proof-of-concept prototype setup and implementation possibilities in Section 4. In Section 5, we evaluate our prototype across multiple test environments. We identify the known limitations and discuss our findings in Section 6 and highlight possible future work in Section 6.2. Finally, we conclude our work in Section 7.

## 2 BACKGROUND AND RELATED WORK

In this section, we review related research and introduce a brief background to the key concepts used in this work.

### 2.1 Data Sovereignty and Control

Securing data while maintaining control in a self-determined way characterizes the term data sovereignty (Hellmeier and von Scherenberg, 2023; Jarke et al., 2019). While a single definition would not cover the term's complexity, current conceptualization describes it as the ability of a data provider to negotiate specific contract agreements on how a data asset is used by an external data consumer (von Scherenberg et al., 2024). This allows data owners to set specific conditions on how their data may be requested and consumed.

Current solutions to address data sovereignty are using connectors for each data sharing participant in order to satisfy access control requirements (Otto et al., 2022). Through the use of additional frameworks and mechanisms, connectors are able to enforce usage limitations on data within their runtime environments (Jung and Dörr, 2022).

Research was done to explore data usage control enforcement in different contexts. For example, related work explored the use of both trusted platform modules and secure execution environments, such as Intel SGX, to create isolated environments ensuring the application of certain restrictions on data usage (Wagner et al., 2018). Moreover, work was done to explore the basic use of certain transmission protocols, such as the Transmission Control Protocol (TCP), to pre-exchange data usage policy information and create enforcement conditions once the data is transmitted (Kelbert and Pretschner, 2013).

In other contexts, work was done on the topic of data usage control and data privacy at the operating system level. For example, mechanisms were created to bind policies defined by data owners to datasets being exchanged as a complete bundle. (Wang et al., 2013). This bundle was then analyzed by the operating system's kernel, and the conditions set in the bundle were enforced. More recent work summarized a wide range of usage control mechanisms across various use cases (Akaichi and Kirrane, 2025).

In this work, we explore facilitating the enforcement of certain data usage control aspects through the use of network protocols and Internet Protocol (IP) Version 6 packet headers.

### 2.2 Steganography

Under the umbrella of the research domain for information hiding, the possibility of hiding secret information inside a cover medium for secure communication is often referred to as steganography (Ahvanooey et al., 2022; Kahn, 1996; Krishnan et al., 2017). The hidden information and the cover medium can range from pure text to images and videos to specific network packets. Especially in digital communication, *network steganography* has established itself as an independent term with increased research interest, with various implementations (Seo et al., 2016).

Implementations that use internet protocols such as IPv4 and IPv6 to covertly send information have been researched for various use cases. Methods include the use of fragmentation bits within an IPv4 header to covertly transmit data (Kundur and Ahsan, 2003). Further, specific fields within the IPv4 packet headers, such as the overflow field of the timestamp options were used to transmit covert information across networks (Bedi and Dua, 2020b). In IPv6, extension headers have been used to transmit data (Mazurczyk et al., 2019; Bedi and Dua, 2020a). We base our approach on similar principles to embed data usage policies within IPv6 packets.

### 2.3 Networking Concepts

In this subsection, we introduce the network concepts related to specific headers, which are essential for understanding our work.

#### 2.3.1 IP Packet Headers

Data is transmitted through the internet mainly via the IPv4 and IPv6 network-layer protocols, which contain metadata such as the sender and receiver IP addresses. The header structure varies based on the protocol version being used, as shown in Figure 1a.

IPv4 Header	Payload
-------------	---------

(a) IPv4 packet structure.

IPv6 Header	TCP Header	Payload
-------------	------------	---------

(b) Example of an IPv6 packet structure.

Figure 1: Packet structure for IPv4 and IPv6 based on Deering and Hinden (1998).

Next Header	Header Length	Options
-------------	---------------	---------

Figure 2: Structure of IPv6 Hop-by-Hop Options extension header following Deering and Hinden (1998).

The IPv4 header contains the source and destination addresses, total packet length in bytes, IP header length, data protocol information, and checksum. The total length of the IPv4 header is limited to 60 bytes (Postel, 1981). IPv6 was introduced to address the shortcomings of IPv4, including address space and extensibility. In contrast to the fixed-header size of IPv4, IPv6 offers a more modular structures. This allows for variable size information to be transmitted if needed, as illustrated in Figure 1b. Furthermore, the variable size enable Extension Headers, addressed in more detail in Section 2.3.2. Each IPv6 packet starts with the IPv6 header for routing, containing the source and destination addresses and a *next header* field, specifying the type of the next header for processing (Hagen, 2014).

### 2.3.2 IPv6 Extension Headers

Extension headers were introduced in order to make IPv6 extensible, allowing the possibility to add functionality in the future beyond the basic definitions (Hagen, 2014), but are purposefully vaguely defined. They require only three fields: *Next header*, defining the type of subsequent header, *length*, defining the length of the extension header, and the actual *data* field. Typically, extension headers are not to be processed by intermediary nodes, except the Hop-by-Hop Options extension header (Deering and Hinden, 1998). Figure 2 illustrates the Hop-by-Hop Options header structure, where the data field in the header was renamed to *options* to represent the contents of the header.

The Hop-by-hop Options header contains information to be processed by the nodes along the network path. As shown in Figure 3, each option must contain an identifying option type field, defining data padding, processing behavior, and whether the content can be changed by the nodes. It also specifies the length of its data field in bytes. Finally, the data field contains the data to be processed at the nodes (Car-

Opt Type	Opt Length	Opt Data
----------	------------	----------

Figure 3: Structure of an IPv6 Hop-by-Hop Option as defined by Deering and Hinden (1998).

penter, B and Jiang, S, 2024). The Destination Option header retains the same data structure but is only processed at the destination (Deering and Hinden, 1998).

In this work, we use Hop-by-Hop Options and Destination Options headers as the means to transmit usage policies within network packets.

## 3 PROPOSED APPROACH

We propose the use of IPv6 extension headers to enrich network packets with metadata, addressing RQ 1 by facilitating control mechanisms that enable data usage control at the network protocol level. Our proposal serves as an addition to the application and transport-level enforcement mechanisms currently available through connectors applications, or within secure environments as examined by Wagner et al. (2018).

Based on data sovereignty concepts, our proposal allows data owners to define a set of rules to be attached to packets leaving their system. Unlike the Dataspace Protocol (DSP)<sup>2</sup>, which uses the full Open Digital Rights Language (ODRL) standard to define the policies, we present a simpler, smaller policy format due to size considerations and processing overheads. To enable compatibility with the DSP, ODRL can be mapped to our proposed format, enabling the use within existing dataspace. Code 1 shows a sample policy following our proposed format.

```
{
  "t": "PLC",
  "d": "2001:0db8:85a3::8a2e:0370:7334",
  "r": "DE",
  "o": "a1b2c3"
}
```

Code 1: Example of a network-specific policy with a geofenced restriction.

To minimize space usage, each field within the policy uses a single letter to identify its purpose. The policy definition contains metadata information to identify the type of policy ("t"). Next, it defines the packet's destination ("d"). Its value is an IPv6 address, defining a single user or a complete network. This provides data owners with the ability to define

<sup>2</sup><https://docs.internationaldataspaces.org/ids-knowledgebase/v/dataspace-protocol>, last accessed: March 2nd, 2025

the parties that can gain access to the data and prevent possible misuse through malicious requests.

Next, the policy sets the data's regional constraints through the field ("r"). This restriction allows data owners to define the region or country where the data can be accessed. This gives data owners additional assurance to protect their data and limit data transport to locations where data privacy restrictions are not strict. For example, the data owners may define that their data may not leave Germany (defined using the ISO 3166 code "DE" in Code 1), where strict General Data Protection Regulations (GDPR) are applied.

Finally, the policy defines a flexible options field ("o"). The use of this field is purposefully left open to allow for different implementations. For example, the option field may be interpreted as the action that can be taken by the node inspecting this policy, e.g., terminating the transmission and notifying the sender.

As each packet can have one Hop-by-Hop Options and one Destination Options extension header, the policy is embedded as a single option within that header. As detailed in Section 2.3.2, each option must contain three fields: The option type, the length of the data, and the data itself. The policy content is embedded within the data field, with its length directly calculated and set within the option length field. Within the option type field, we define that the processing node should ignore the option header if it is not understood and that the remaining headers should be processed typically. This allows nodes to continue processing or forwarding the packet without dropping it unnecessarily. This approach follows the standard definition of the Hop-by-Hop Options and Destination Options extension header (Deering and Hinden, 1998).

Using a plain JSON string directly within the extension header facilitates easier identification and processing by responsible parties, but it also increases the risk of malicious removal of policies. To mitigate this, obfuscation methods can prevent the detection of the policy. Simple algorithms, such as ROT47, based on the Caesar Cipher (Wobst, 2007), can be applied without severe overhead, as they shift the characters by a specified value.

## 4 IMPLEMENTATION POSSIBILITIES

To answer RQ 2, we examine various possible ways to implement our proposed solution. The two most realistic and implementable approaches, in-node processing and the use of middle boxes, are introduced in the following.

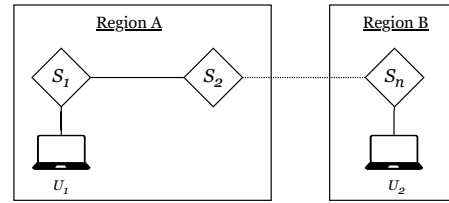


Figure 4: An example implementation based on the usage of routers for enforcement.

### 4.1 In-Node Processing

The first approach involves processing the headers directly at the network nodes. It requires a correctly configured network infrastructure (i.e., routers, layer-3 switches, etc.) that supports IPv6 Extension Header processing within each node. Figure 4 illustrates a scenario implementing this approach. The scenario involves two users located in different regions, interconnected through various network nodes.

In the example shown in Figure 4, a user  $U_1$  is directly connected to a regional network with network provider  $S_1$  in "Region A". Through this connection,  $U_1$  sends data to  $U_2$ , which is located in "Region B". Upon sending data,  $U_1$  appends all outgoing IPv6 packets with a Hop-by-Hop Options Header containing the policy as option data. Upon arrival at  $S_1$ , the option contents are processed to ensure that the constraints for forwarding of the data are met. Upon arrival at  $S_2$ , the last node in region A, along the selected path, the option data are processed again to ensure that the route matches with the destination region. At  $S_n$  in the destination's region B, the policy is processed further to ensure that the IPv6 address matches the one set in the policy constraint. If any node along this path decides not to forward the request, the transmission is interrupted. The node can optionally notify the user through an Internet Control Message Protocol Version 6 (ICMPv6) that the transmission has failed.

The main benefit of this approach is utilising existing infrastructure. As the processing is done in-node, the delay overheads incurred are minimal. Further adjustments to the infrastructure are also minimal since the method used is based on the standard IPv6 protocol, and it should be supported by network devices adhering to it.

To allow the network infrastructure to process the added extension headers within the packet, the network operator must apply a new configuration at various network locations, or at least along several pre-selected paths designated for these packets. Depending on the size of the network and the infrastructure setup, the difficulty level of this option can vary from easily achievable to economically infeasible.



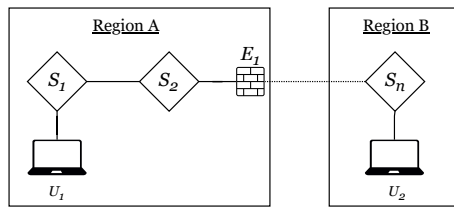


Figure 5: An example implementation based on the usage of transparent middle boxes for enforcement.

## 4.2 Use of Middle Boxes

To address possible implementation difficulties of the first approach, we propose a second method based on the use of network middleboxes that reduces realization complexity. A middlebox is defined formally in RFC 3234 as “any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host” (Carpenter and Brim, 2002, p. 3). This definition allows middleboxes to cover a wide variety of use cases and applications, such as firewalls, Intrusion Detection Systems, VPN Gateways, etc. (Benhabbour and Dacier, 2025).

To implement our proposal, the network operator must route the traffic between two networks through a suitable middlebox. Figure 5 illustrates this deployment. As shown, traffic flowing from  $S_2$  to  $S_n$  is routed through a middlebox  $E_1$ , either physically by rewiring and re-routing connections, or virtually through Software Defined Networking (SDN) and Network Function Virtualization (NFV) concepts.

The main advantage of using middleboxes is the speed and ease of deployment. This allows our proposal to be implemented for initial tests within a network without significant changes to the network infrastructure, as it’s attached to existing network components. Furthermore, depending on the middlebox performance and optimization level, this approach can be used as a permanent implementation. The middlebox can be set up in a “transparent” mode, in which the transmission’s source and destination are unaware of its existence. This could allow for further use cases where the embedding of policies in headers does not occur directly at the source of the transmission, for example, with the use of a corporate Virtual Private Network (VPN).

An additional advantage of this approach is the ability to use different or custom Extension Header formats. Since middleboxes examine each packet for extension headers, operators are not restricted to Hop-by-Hop Options extension headers; They can instead choose to use the Destination Options extension headers or create their own specialized header format per

the IPv6 standard to address specific use cases.

Generally, the use of middleboxes can incur additional delays due to the additional routing of a new network component. Further delays can be incurred during the processing at the middlebox itself. However, these delays depend on the middlebox’s specific implementation and optimization level.

## 5 PROTOTYPICAL EVALUATION

To address the technical aspects of RQ 3, we build a prototype of our proposal and evaluate its validity and applicability with various parameters across different deployment scenarios.

### 5.1 Evaluation Setup

We evaluate our proposal in varying conditions designed to validate two main test cases. The first case is the implementability using middleboxes. The second case is aimed to test the applicability of the proposed solution outside of controlled environments.

#### 5.1.1 Validation in Controlled Environments

For the first test case, we use the GNS3<sup>3</sup> network emulator to create a virtual environment closely representing real network operating environments and topologies. Similar to the topology illustrated in Figure 5, we created a topology that includes the two geopolitical regions, EU and UK. Within the EU, we set up two routers representing the top-level networks of Germany and France, with the latter’s network connected to the UK network. The link between these networks is adapted to use a middlebox that functions as the policy enforcer component. The task of the middlebox is to scan each packet traversing the link from the EU side to the UK side. The middlebox then blocks any transmission containing a policy that restricts the data transmitted to the region “EU”. Further, the links between all networks within the emulator are configured without artificial delays.

To inject the policies at the server side in Germany, we used a specific IP table configuration and created an application that uses the NetfilterQueue<sup>4</sup> and the Python Scapy<sup>5</sup> Libraries to intercept and add the extension headers to outgoing IPv6 packets.

The enforcer ( $E_1$ ) component is implemented similarly, using the Ubuntu distribution of the Linux op-

<sup>3</sup><https://gns3.com/>, Last accessed: February 17th, 2024

<sup>4</sup><https://github.com/oremanj/python-netfilterqueue>, last accessed: March 2nd, 2025

<sup>5</sup><https://scapy.net/>, last accessed: March 2nd, 2025

erating system and a specific IP tables ruleset. The enforcer application was written in Python using the aforementioned libraries. The client in the UK used the simple wget<sup>6</sup> command-line interface (CLI) to request the main webpage from the server in Germany.

The experiments were run in batches. We use the policy format shown in Code 1 in every batch. To measure the possible effects of the total size of the policy contents on the performance, we systematically adjust the size of the options field "o". The field is initially populated with five characters, with systematic increments in character length in steps of two characters for each experiment batch. The increments continue until 55 characters are reached. Each experiment batch consists of 10 requests sent from the client in the UK to the server in Germany. Each request includes the policy within an extension header in each packet header. To evaluate the cases, we collect the mean roundtrip time and success rate metrics.

Additionally, two experiment batches are used to evaluate the baseline roundtrip metrics without extension headers. One experiment evaluates the mean roundtrip time without a middle box, and the other evaluates the mean roundtrip time with a middlebox added that only forwards the packets without additional processing. These experiments help to identify the basic delays incurred by a packet when traversing the emulated network setup.

### 5.1.2 Applicability Outside Controlled Environments

In addition to the experiments in a controlled laboratory setup, we test the applicability and limits of our proposed solution in other environments. For this, we test the survivability of our proposed approach across the Internet. The main driving factor behind this case is the work done by Léas et al. (2022). They determine that packets embedded with large Hop-by-Hop Options extension headers are mostly blocked by Internet Service Providers (ISPs). As such, we present this test case to additionally test the content limits of our proposed solution.

For this case, we designed various experiments that are similar to the previous case. The experiments include using the cloud provider Amazon Web Services (AWS) to deploy IPv6 capable services across multiple continents. For each test scenario, batches of multiple HTTP requests are made to each deployed service. Table 1 summarizes the tested AWS locations.

Similar to the experiment setup defined in 5.1.1,

Table 1: AWS locations used to test this work.

Location Code	City, Country
ap-south-1	Mumbai, India
ap-northeast-3	Osaka, Japan
ap-northeast-2	Seoul, South Korea
ap-southeast-1	Singapore, Singapore
ap-southeast-2	Sydney, Australia
ap-northeast-1	Tokyo, Japan
ca-central-1	Montreal, Canada
eu-central-1	Frankfurt, Germany
eu-west-1	Ireland, Ireland
eu-west-2	London, United Kingdom
eu-west-3	Paris, France
eu-north-1	Stockholm, Sweden

the requests were run in batches of 10 experiments for each extension header size configuration. Similarly, each request is modified to include an extension header with varying options "o" field.

Unlike the first experiment case, this experiment used a different technical basis for the execution. Instead of virtual machines running within a GNS3 emulator environment, these requests were run on a dedicated Linux machine connected to the Internet through a Coaxial connection with 1000 mbps download speed and a 50 mbps upload speed. An HTTP request is sent per experiment to the AWS service, which in turn only returns an empty response, indicating the request was successfully received.

To generate a baseline for each experiment batch with an embedded policy, a baseline experiment batch was run for each AWS location. The baseline consisted of a pure IPv6 HTTP request and did not include any extension headers. These experiments enable us to collect baseline metrics to evaluate the effects of extension header insertion and processing along various routes.

In our initial experiments, we used the Hop-by-Hop Options extension header to embed the policy information. However, the requests were mostly rejected, validating the conclusions made by Léas et al. (2022). As such, we adjusted the packet insertion method to use the similarly structured Destination Options extensions header to embed the policy information, still allowing for the use of middleboxes for enforcement.

## 5.2 Results

We examine experiment results in order to answer the performance aspect of RQ 3, structuring the presentation into two parts based on the environment used.

<sup>6</sup><https://www.gnu.org/software/wget/>, last accessed: March 2nd, 2025

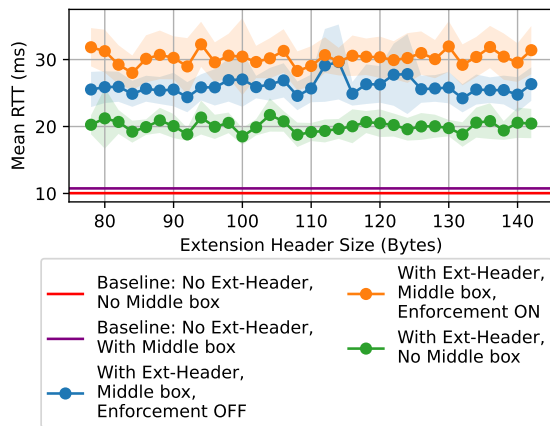


Figure 6: Mean roundtrip times in controlled environments.

### 5.2.1 Controlled Environments

In the first result set, we present the roundtrip times of the packets obtained from the emulated test environment. The results from the experiments are shown in Figure 6.

The results show that a baseline mean roundtrip time for a request without any extension headers traversing a middle-box-free link from the client to the server is 10ms. The addition of the middlebox to the setup incurs an additional 1ms delay. This additional delay is negligible in non-real-time applications.

The first main added delay starts when inserting the extension headers without a middlebox instance along the link, as seen when observing the green line in Figure 6. Once an extension header is inserted, the mean roundtrip time is shifted to a range between 18-22ms. The process of header insertion at the server is therefore incurring an additional 7-11ms delay.

Once a middlebox is added to the topology, the delays are further increased, as seen from the blue line. In this case, a middlebox operates on the link between EU and the UK. The middlebox software intercepts packets; However, no application logic is applied. As such, all packets are automatically accepted for forwarding. This interception process of the middlebox incurs an additional 4-10ms, resulting in a total mean roundtrip time of 23-29ms. During this time, the middlebox intercepts the packet at its inbound interface, transfers it from the kernel level of the operating system to the user-level, and then forwards it to the middlebox software for a final forwarding decision.

The final delay increase occurs when the middlebox software is operating in "Enforcement" mode. In this mode, the middlebox actively parses the packet headers and extracts the extension header content. The extracted content is scanned for a policy accord-

ing to the format defined in Section 3. After parsing the policy contents, the middlebox compares the policy elements with the metadata of the packet headers. If the packet is allowed to leave the EU, the decision is to forward it towards its destination. In total, this process incurs an additional 5-10ms, resulting in a total mean roundtrip time of 27-33ms.

Throughout the experiment batches with various extension header sizes, the mean delays across all cases remain stable without major fluctuations. As such, the added delay costs are easily determined at each stage of the experiments, allowing for specific optimizations to be conducted.

### 5.2.2 Outside Controlled Environments

In the second result set, we present our experiments' mean roundtrip times and success rates against the various AWS instances deployed worldwide. As explained in Section 5.1.2, we replace the use of the Hop-by-Hop Options extension header with the Destination Options extension header. This change does not affect the functionality of the middlebox, as it processes all packets and extracts header information as configured.

The first result subset covers the mean roundtrip time and is shown in Figure 7. The figure includes the baseline value for each experiment without any extension header. The baseline is distinguished with the extension header size 0 for each plotted data series. Further, the subset results are grouped based on their geographic location to maintain a similar result scale for each group.

Generally, the roundtrip times remain stable for each region compared to its baseline until reaching a total extension header size of 108 bytes. Afterwards, the mean roundtrip time starts to vary significantly based on the region. In the case of the Asia-Pacific region, the roundtrip times increase from below 1000ms to over 12000ms in the worst case if the total size of the extension header is larger than 108 bytes. In the case of Europe, the roundtrip times remained stable throughout the various sizes, except for the region "eu-west-2 (London, UK)". In this particular location, the roundtrip time increased from below 400ms to over 6000ms after the extension header size increased to more than 108 bytes.

The final result subset covers the mean request success rate and is shown in Figure 8. Similar to the previous subset, the baseline values are included at the data point 0 for each experiment.

Similar to the behavior observed in the roundtrip time results, the mean success rate of the packet is measured at 100% for the majority of the extension header sizes. Moreover, the decrease in request suc-

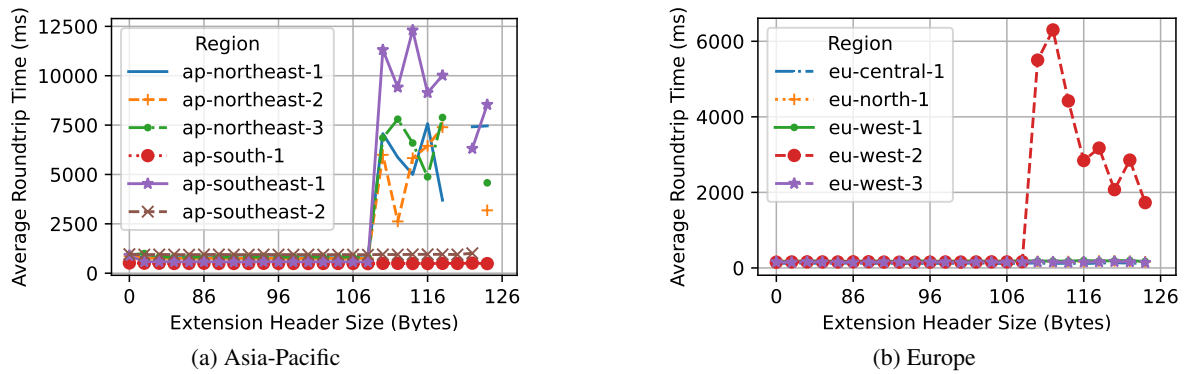


Figure 7: Average roundtrip times for packets tested against the various AWS regions.

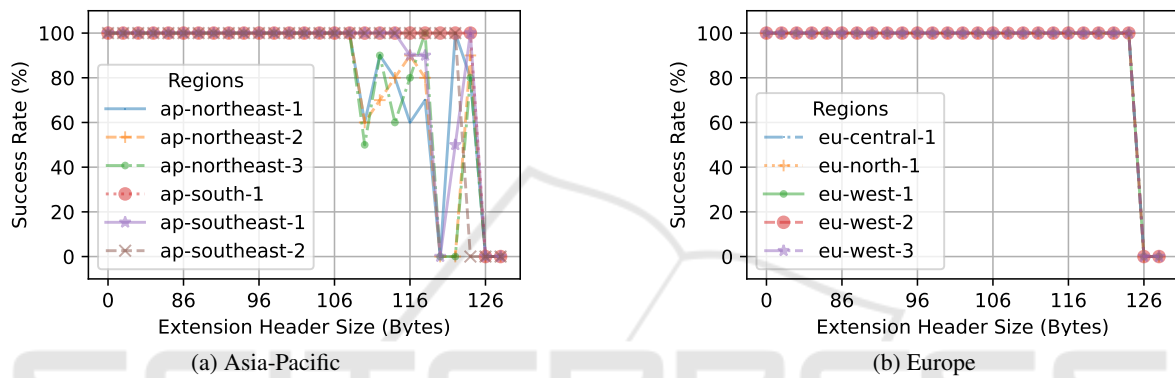


Figure 8: Average success rate for packets tested against the various AWS regions.

cess rate begins to occur after reaching 108 bytes of total extension header size. The most detrimental results are observed in the Asia-Pacific region. Here, the request success rate starts to decrease after 108 bytes and continues to remain unstable afterward. This continues until reaching 126 bytes, after which all requests begin to have a 0% success rate.

Comparably, the results observed in Europe do not experience a similar drop for the vast majority of extension header sizes. A rate drop from 100% to 0% is only observed upon reaching an extension header size of 126 bytes. We attribute this drop in success rate to various factors, such as active firewalls blocking the packets with unusually large headers, or to limitation to packet sizes enforced by multiple network operators along the routes.

We further examined the observed varying roundtrip times. We repeated the experiments for those cases and closely studied the captured and transferred packets. We deduced that the requests were experiencing drops along certain high-bandwidth or high-speed routes to their destination. As such, the packets were retransmitted and rerouted through other and often longer routes.

## 6 DISCUSSION

Our proposal provides an answer to our main research question RQ 1 by enabling aspects of usage control at the network protocol level. Its main benefit is the addition of a protection layer without exposing the payload. Owners can add policies to their data that can be verified and enforced by authorized third parties, such as network providers, without affecting any encryption and without accessing the contents of the data. Our proposed implementation methods and performance evaluation provide answers to RQ 2 and RQ 3.

Our proposal can be a significant factor in increasing trust among parties sharing data, as a data owner will not need to solely trust the services running at the data recipient's side to enforce or uphold any possible data usage policies. The enforcement takes place by preventing data transfer at the network level. Furthermore, our proposal acts to strengthen existing data privacy regulations, like the GDPR in the European Union, by providing an enforcement mechanism to identify and prevent data from leaving certain geographical regions, ensuring data privacy and data



sovereignty is preserved.

The approach has implications for data ownership and data sovereignty, namely regarding the scope of explored technologies. To our knowledge, no recent work has been done on implementing aspects of data sovereignty at the network protocol level. As explored in Section 2.1, the available research focuses on the two main parties (sender and receiver) and not the medium or protocols through which the data is being transferred.

## 6.1 Limitations

As with any new research area, we are aware of various limitations discussed in the following.

**Delay Increase.** Regular IP packets are usually processed and forwarded by nodes at the hardware level, scanning for specific data fields, which minimizes any possible delays. However, processing extension headers involves increased resource requirements, as shown in our prototypical evaluation. This can cause significant delays in comparison to standard packets. To address this point, future work can examine the possibilities of performance optimization and designing frameworks to enable hardware acceleration to process the policy header. Furthermore, methods to efficiently embed the extension headers on the kernel level, such as the work by Iurman et al. (2023), can be applied to significantly reduce initial delays seen in Figure 6.

**Survivability of Packets with Hop-by-Hop Options Extension Headers.** In addition to their benefits, using IPv6 Extension Header in networks adds some security concerns. Gamilla et al. (Gamilla and Naagas, 2022) concluded that extension headers can be used to evade Intrusion Detection Systems, allowing for various other attacks. These security implications and increased resource use cause internet providers to drop packets with extension headers. As discussed in Section 5.1.2, we observed the drop of packets with the Hop-by-Hop Options extension header. However, packets including the Destination Options extension header are not as strictly blocked until certain size limits are reached. To address this point, awareness could be raised among network providers about data usage control and the use of extension headers for this purpose so that network operators can handle it as a *known option*.

**Security Considerations.** To answer the security aspect of RQ 3, malicious actors can use man-in-the-middle attacks to manipulate IP packets along their

route to their destination (Bhushan et al., 2017). In this case, the extension header can be modified to alter the intended behavior of nodes processing the policy extension header. Using obfuscation methods such as ROT47 can provide an additional deterrent. To address further considerations, future work can examine the use of verification methods to detect unauthorized modifications and drop the packets where such instances have been found. Another possibility would be to use encryption mechanisms, where authorized nodes encrypt/decrypt the content of the policy header and perform the required actions.

## 6.2 Future Work

There are various points pertaining to possible future work in this field, both theoretically and practically. More work can be done on the theoretical aspect, examining new use cases and potential policies. From the practical angle, further work can be done to address the points defined in Section 6.1 so that the known limitations can be well addressed.

In terms of the total size of the policies being embedded within an IPv6 extension header, a problem may arise if the policy used is extensive. Large extension headers might cause problems due to fragmentation applied at the intermediary nodes. To address this possible issue, research can be done on the use of compression algorithms and encoding mechanisms on the content of the policy options. Here, research can be done to examine both, the applicability of such algorithms in this context, and the implication of their use on the total performance.

## 7 CONCLUSION

In this work, we presented a novel approach to data usage control that functions at the network protocol level based on the use of IPv6 extension headers. We offered multiple implementation possibilities for our proposed approach and how it can help facilitate new mechanisms for usage control. Overall, the evaluation results demonstrate the viability of utilizing network protocols and IPv6 extension headers as additional mechanisms to enforce certain usage control conditions. Moreover, using our approach enhances data sovereignty by having additional parties act as enforcers of the policies without gaining access to the data being exchanged. Our work can be further examined and expanded to explore its implementation and adjustment requirements across various use cases.

## ACKNOWLEDGEMENTS

This research was supported by the Cluster of Excellence Cognitive Internet Technologies CCIT and the Center of Excellence Logistics and IT, funded by the Fraunhofer-Gesellschaft.

## REFERENCES

- Ahvanooey, M. T., Zhu, M. X., Mazurczyk, W., and Ben-dechache, M. (2022). Information hiding in digital textual contents: Techniques and current challenges. *Computer*, 55(6):56–65.
- Akaichi, I. and Kirrane, S. (2025). A comprehensive review of usage control frameworks. *Computer Science Review*, 56:100698.
- Bedi, P. and Dua, A. (2020a). Network steganography using extension headers in ipv6. In *International Conference on Information, Communication and Computing Technology*, pages 98–110. Springer, Singapore.
- Bedi, P. and Dua, A. (2020b). Network steganography using the overflow field of timestamp option in an ipv4 packet. *Procedia Computer Science*, 171:1810–1818.
- Benhabbour, I. and Dacier, M. (2025). Endemic: End-to-end network disruptions - examining middleboxes, issues, and countermeasures - a survey. *ACM Computing Surveys*.
- Bhushan, B., Sahoo, G., and Rai, A. K. (2017). Man-in-the-middle attack in wireless and computer networking — a review. In *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA) (Fall)*, pages 1–6, Piscataway, NJ. IEEE.
- Carpenter, B. and Brim, S. (2002). RFC 3234 - middleboxes: Taxonomy and issues.
- Carpenter, B. and Jiang, S. (2024). RFC 7045: Transmission and processing of ipv6 extension headers.
- Deering, S. and Hinden, R. (1998). RFC 2460 - internet protocol, version 6 (IPv6) specification.
- Gamilla, A. P. and Naagas, M. A. (2022). Header of death: security implications of IPv6 extension headers to the open-source firewall. *Bulletin of Electrical Engineering and Informatics*, 11(1):319–326.
- Hagen, S. (2014). *IPv6 essentials: Integrating IPv6 into Your IPv4 Network*. O'Reilly, Beijing, 3rd ed. edition.
- Hellmeier, M. and von Scherenberg, F. (2023). A delimitation of data sovereignty from digital and technological sovereignty. *ECIS 2023 Research Papers*, 306.
- Iurman, J., Vyncke, E., and Donnet, B. (2023). Using eBPF to inject IPv6 extension headers. In *Netdev 0x17*. Netdev.
- Jarke, M., Otto, B., and Ram, S. (2019). Data sovereignty and data space ecosystems. *Business & Information Systems Engineering*, 61(5):549–550.
- Jung, C. and Dörr, J. (2022). Data usage control. In Otto, B., ten Hompel, M., and Wrobel, S., editors, *Designing Data Spaces*, pages 129–146. Springer Cham.
- Kahn, D. (1996). The history of steganography. *International Workshop on Information Hiding*.
- Kelbert, F. and Pretschner, A. (2013). Data usage control enforcement in distributed systems. In Bertino, E., editor, *Proceedings of the third ACM conference on Data and application security and privacy*, ACM Digital Library, pages 71–82, New York, NY. ACM.
- Krishnan, R. B., Thandra, P. K., and Baba, M. S. (2017). An overview of text steganography. In *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–6. IEEE.
- Kundur, D. and Ahsan, K. (2003). Practical internet steganography: Data hiding in IP. *Proc. Texas wksp. security of information systems*.
- Léas, R., Iurman, J., Vyncke, É., and Donnet, B. (2022). Measuring IPv6 extension headers survivability with james. In *Proceedings of the 22nd ACM Internet Measurement Conference*, New York, NY, USA. ACM.
- Mazurczyk, W., Powójski, K., and Caviglione, L. (2019). IPv6 covert channels in the wild. In *Proceedings of the Third Central European Cybersecurity Conference*, ACM Digital Library, pages 1–6, New York, NY, United States. Association for Computing Machinery.
- Otto, B., ten Hompel, M., and Wrobel, S., editors (2022). *Designing Data Spaces*. Springer Cham, 1 edition.
- Postel, J. (1981). RFC 791: Internet protocol.
- Seo, J. O., Manoharan, S., and Mahanti, A. (2016). Network steganography and steganalysis - a concise review. In *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pages 368–371. IEEE.
- von Scherenberg, F., Hellmeier, M., and Otto, B. (2024). Data sovereignty in information systems. *Electronic Markets*, 34(1).
- Wagner, P. G., Birnstill, P., and Beyerer, J. (2018). Distributed usage control enforcement through trusted platform modules and sgx enclaves. In Bertino, E., Lin, D., and Lobo, J., editors, *Proceedings of the 23rd ACM on Symposium on Access Control Models and Technologies*, pages 85–91. ACM.
- Wang, X., Yong, Q., Dai, Y., Ren, J., and Hang, Z. (2013). Protecting outsourced data privacy with lifelong policy carrying. In *HPCC and EUC 2013*, pages 896–905, Los Alamitos, California and Washington and Tokyo. Conference Publishing Services, IEEE Computer Society.
- Wobst, R. (2007). *Cryptology unlocked*. John Wiley & Sons, Chichester and Hoboken, NJ.
- Zrenner, J., Möller, F. O., Jung, C., Eitel, A., and Otto, B. (2019). Usage control architecture options for data sovereignty in business ecosystems. *Journal of Enterprise Information Management*, 32(3):477–495.