Internet transparency through multi-party computation

Paweł Maćkowiak and Fernando Kuipers

TU Delft, Delft, the Netherlands
{p.mackowiak@student.,F.A.Kuipers@}tudelft.nl

Abstract. The inability to check how our Internet traffic is being handled and routed poses all kinds of security and privacy risks. Yet, for the typical end-user, the Internet indeed is such a black box. In this paper, we adhere to the call for an Internet that is more transparent, and as a step forward we propose a mechanism that carefully balances the desire to share transparency information with the necessity to not expose all internal details of a network. We realize this by building on the framework of multi-party computation. Our architecture and corresponding proof-of-concept is evaluated via experiments and demonstrates the feasibility of our concept to improve Internet transparency.

1 Introduction

The Internet is of vital importance to our modern society, yet – to its end-users who generate and receive network traffic – it provides limited transparency and hardly any control regarding how user-traffic is processed.

In this paper, **transparency** relates to the level at which a user is able to see how their network traffic is being processed. Following such transparency, **control** refers to the level to which end-users are able to determine or convey how their traffic should be handled by networks.

Technologies like Virtual Reality (VR) or the Tactile Internet are extremely resource-intensive and resource-sensitive [16], which means that poor Quality-of-Service (QoS) directly affects the Quality-of-Experience (QoE). Providing QoS over multiple domains is especially challenging. In many cases, different network domains are administered by different entities, which significantly complicates the ability to collect performance measurements and data on resource allocation. As a result, supervision (and hence transparency) becomes ineffective and sometimes even impossible. In an ideal world, it would be sufficient to draw up a number of agreements between Internet Service Providers (ISPs) and domain administrators in order to ensure that sufficient resources are always available. Unfortunately, in practice and because of the complexity involved in networking, this has not been realized.

Even within a single domain, once an anomaly has been observed, further diagnostics are needed to properly understand what has happened. This means labor-intensive network administrator intervention. Such diagnostic efforts greatly

increase when multiple domains are involved, making it even harder to find the root cause of network and service problems. In some cases, this results in situations where the domains' administrators claim that other parties are responsible for a certain service disruption. Clearly, to provide end-users and domain administrators with more network-related insights poses a non-trivial challenge; a solution is needed for multi-domain supervision and transparency of network services and devices, which does not compromise the security of the involved domains nor discloses any competitive network information. This paper contributes to solving that challenge, thereby leveraging existing and novel telemetry protocols and technologies.

The structure of this paper is as follows: Section 2 presents our architecture along with a Proof-of-Concept (PoC) implementation. This PoC is experimentally evaluated in Section 3. Section 4 presents related work and we conclude in Section 5.

2 Design of a multi-domain network telemetry system

The first part of this section will present our design goals and further considerations. Subsequently, the proposed approach will be presented, followed by the description of the Proof-of-Concept (PoC) developed by us.

2.1 Design goals and considerations

The main design goal is to offer users greater transparency, i.e., to give them insight into how the network traffic generated by these users and their applications is processed. In addition to this high-level goal, the objectives further detailed in the following are to propose a modular, easy-to-adopt system that will enable users to securely share transparency data.

Goals In creating a system to provide third parties with (controlled) access to network characteristics, the question arises of how to balance the amount of information provided versus the security of the infrastructure they describe. The information provided may include details about the devices included in the infrastructure (manufacturer of the device, software version, relations with other devices in the stack), details about the processing of network traffic, such as the exact path travelled, the associated delays, and resources used, and even other types of information, such as its energy profile. However, openly disclosing such information could enable malicious actors to more easily identify vulnerable points in the network, which on its turn may significantly accelerate potential attacks on the network. While one could argue that fast-paced patching or deception technologies[1] may reduce the risk of such security incidents manifesting, this

[1] Deception technologies refer to cyber security defence mechanisms that facilitate early threat detection and enhanced incident response by means of deploying fake/misleading targets in the network.

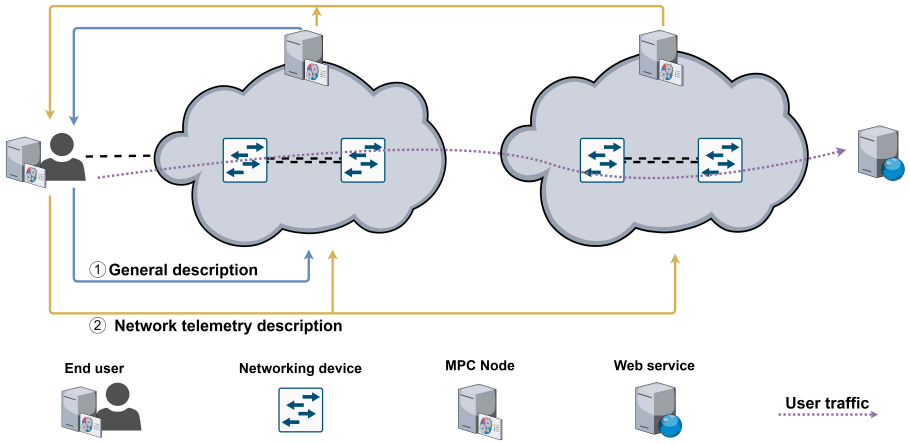


Fig. 1. High-level overview of the proposal with two types of description.

is countered by the increasing levels of automation that may significantly accelerate the occurrence and execution of potential attacks. Similarly, the provision of sensitive data describing how ISPs operate may also impact their competitiveness. It is therefore important to deploy a mechanism that allows to securely share transparency information with trusted users. It is outside the scope of this paper to indicate what information should or should not be disclosed. But we do make it an objective to allow secure information sharing as an important step towards a responsible Internet.

Furthermore, the complexity of adopting the solution has a direct impact on the desire to use it. If a better level of transparency requires operators and users to use a particular type of equipment, the pace of adoption may be too slow to create impact. An appropriate level of modularity will enable the collection of data from various types of devices, which will allow more users to benefit from such a solution.

2.2 Design proposal

This section will describe the system design process and the accompanying decisions resulting from the goals described in the previous section. The process will be described from top to bottom, i.e., first the part of the proposal responsible for analysing and processing data between domains will be described and then the elements responsible for providing data to the top part will be discussed.

In Figure 1, a high-level overview of the system is presented. As shown in that figure, the initiator of the data sharing process is the end-user, who can ask for two types of description. The first one is a general description summarizing how network traffic is processed by the domain, similar to security audits conducted within the Cloud Security Alliance STAR framework [11]. It may include information such as a description of services that process the traffic (e.g.,

DPI, Encryption, DNS), a description of infrastructure processing the traffic at the data-plane and control-plane levels (both software and hardware), the ability to perform measurements on this traffic (data-plane telemetry), support for additional security functions (e.g., DNSSec, DoH), peering relations that are directly related to the path of traffic, or under whose jurisdiction the domain operates. Such a description allows the user to verify that the service provided by the domain meets their requirements/norms/standards. The second type of description results directly from the type of measurements that domains declare they can offer, along with the results of those measurements. As a result of this description, the user can see values such as the processing time of their traffic at different levels of granularity (from a single device to a domain summary) or, for example, the use of resources.

As can be seen from the description of the functions presented above, the user, whether it is an end-user or another service provider, could retrieve a great deal of information about the internals and functioning of the domain they are requesting transparency information from. To avoid such information leaking to malicious entities, he system could be designed using a client-server architecture model, in which the user, if necessary, would obtain encrypted data directly from the server, assuming that (s)he has appropriate permissions. One other approach could be to use a peer-to-peer model, where peers would encrypt the data as part of a protocol for exchanging information between themselves. Unfortunately, in both cases, there still is a risk that a malicious actor could impersonate an entity with privileges to obtain data, consequently, accessing shared information. This may be the case when a malicious insider [12] is involved or an active persistent threat manifests and compromises the system.

We have therefore decided to base our proposed architecture on secure multi-party computation (MPC). MPC allows to carry out a computation without the actual data being shared with the entities participating in the calculation. As a result, entities involved in the exchange will only receive the result without the possibility of gaining access to sensitive information that could jeopardise the security of many. In addition, many MPC frameworks use public key infrastructure (PKI), which allows to validate the authenticity of the data provided for the calculation. In order to successfully utilize MPC, it is necessary to design the to be calculated functions in such a way that the result of those calculations does not include/leak information about the system from which the data originated. In our architecture two function types corresponding to two description types are specified.

The first type of function (general description) is the ability to check the requirements of the user against the domain. The input data for such a function would be a list of requirements that the user has. The result of such a function can help determine whether the requirements are or will not be met, which may allow the user to decide to change the domain being used to transfer traffic or renegotiate the way it is processed.

The second type of function (network telemetry description) is to provide information on the use of the infrastructure, and also to examine measurement

data that will allow to achieve the effect of end-to-end measurements without violating the iOAM standard [8]. The input data for such a function would be the measurement data, and an example of the result – as used in our Proof-of-Concept (PoC) – is the sum of all the delays or the argmax function, which may allow to determine the bottleneck.

In the context of providing data for the implementation of the MPC protocol, it is necessary to define the scope of the domain, from which descriptive and measurement data originate. In our proposal, we postulate that from the point of view of the user using the system, it is important to limit the complexity of the system by minimising communication patterns (e.g., limiting the number of involved parties). For example, if the end-user would like to obtain a description from an operator who outsources part of their operations, it would be the operator's responsibility to obtain data from the entity to which its operations are outsourced, in order to eliminate the need for the end-user to make additional contact with such entities. Therefore, we propose to define domains based on the maintenance domain hierarchy from protocol 802.1ag. The user using the system would only communicate with a domain one degree lower in the hierarchy. Furthermore, the maximum number of entities that could participate in the data exchange would be determined by the number of domains of lower level contained within the customer level.

As for the granularity of the data originating from domains, it could vary depending on the function performed, and should meet user expectations. The level of granularity could range from summaries over a whole domain to per-device information.

The next design aspect is how to transfer data to the MPC protocol. The variety of available measurement technologies and the way of obtaining data from devices significantly limits the possibility of introducing and using one standard. Therefore, in our solution, a message broker is used to resolve this difficulty, as it mediates communication among applications, minimising the mutual awareness that is necessary to exchange messages successfully. Another advantage of the message broker is that most of the open-source brokers have an API for many languages, such as Java, C/C++ or NodeJS, which allows the creation of a simple extension of the devices' functionality with the possibility of sending messages.

2.3 Proof-of-Concept details

This section will explain how we implemented our Proof-of-Concept.

Building blocks Within the framework of the proposal described in the previous section, three main elements can be distinguished. These are (1) the networking device that is extended with functionality to report telemetry data to a message broker, (2) the message broker itself, and (3) the module responsible for executing a specific function using MPC and obtaining data from the message broker.

The most important of these elements is the MPC framework, which forms the core of our solution. Due to the recent interest in this field, many frameworks

	Supported threat model	Supported data types	General support	Last major update
Frigate[7]	N/A	Fix and Arbitrary integer, Array	Documentation, Example code, Open source	8/2020
CBMC-GC[14]	N/A	Fixed integer, Float, Boolean, Array	Partial documentation, Example code, Open source	10/2018
SCALE-MAMBA[15]	Semi-honest, Malicious	Fixed/Arbitrary integer, Float, Array, Struct	Documentation, Online Support, Example code, Open source	03/2022
Wysteria[6]	Semi-honest	Fixed integer, Boolean, Struct	Partial documentation, Example code, Open source	10/2014

Table 1. Comparison of the general-purpose MPC frameworks supporting two or more parties.

have been released. Available frameworks can be divided into two main groups: specialised frameworks and general-purpose frameworks. The main distinguishing features of these groups are the functions offered and their performance. In the former case, the developers focus on the realisation of a specific function and thus try to optimise the performance of their solution for said functionality [9]. The latter group aims at providing a framework that allows for any computation that can be realised within the offered features. General-purpose frameworks tend to be more regularly maintained, which in the long run also improves their performance. In addition, the protocols performed by these general-purpose frameworks are typically described using high-level languages, which benefits its ease of adoption.

In consideration of the above, a general-purpose framework was chosen for our PoC implementation. Among the available general-purpose frameworks, many differ in terms of the number of supported parties, the security model, and the general expressiveness of the high-level language that is used in the framework. The number of parties involved in the execution of these functions could be greater or equal to two. In Table 1, a comparison of the frameworks is presented.

As shown in the table, of the available frameworks, only SCALE-MAMBA provides support against the malicious threat model. According to the documentation provided by the authors, this solution has no support for logical operations (resulting from lack of Boolean variables), which should eliminate it in view of the first of the two functions described in Section 2.2. However, during our evaluation of that framework, we found that the appropriate use of bitwise operations allows overcoming the lack of these logical operations (e.g., a maximum value is calculated using the following expression: $x-((x-y)\&-(x<y))$). Hence, we selected SCALE-MAMBA.

The next building block is the message broker. There are many open-source brokers available, which differ widely in characteristics. Important features that should be taken into account when choosing a broker are high availability, guaranteed delivery and delivery acknowledgement, and how developer-friendly the broker is. Among the brokers that meet these requirements, the following are most popular: Apache Kafka [1], RabbitMQ [2], and ZeroMQ [3]. Their popularity implies better support and ease-of-use and, consequently, matches our

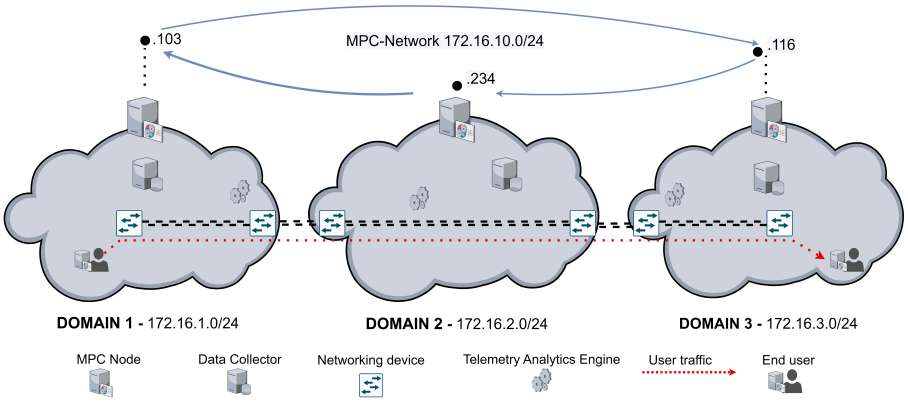


Fig. 2. Overview of the PoC implementation.

objective in terms of adoption potential. From these three, Apache Kafka stands out in one feature. Namely, it has the ability to reproduce messages. In the case of many other message brokers, once a message is consumed it might not be repeated. Yet, this is an important feature for validating the authenticity of the data presented for computation. Hence our choice for Apache Kafka. Moreover, Kafka uses a pull-based approach. This allows for on-demand data analysis, which increases the flexibility of our proposal. Finally, many programming languages have dedicated modules to communicate with the Kafka broker, which again aids its adoption potential.

The PoC development process was carried out in an OpenStack-based cloud platform. As can be seen in Figure 2, the PoC consists of three independent domains and a separate network used for MPC communication. Each domain includes several telemetry-enabled network devices, a message broker, and an MPC node extended with software for providing data and supervising protocol execution. In each of the domains, during operation, the delay between switches is monitored (the delay value includes the delay due to port queuing, packet processing, and link delay). This delay is then reported and made available for multi-domain analysis. In the PoC, three types of analysis were conducted on the gathered data. These types are further discussed in the following section. In the PoC, the edge domains (i.e., 1 and 3) are the same in terms of used telemetry technology, while the middle domain differs. The edge domains consist of a Mininet network with two switches and two hosts, where the switches in the network each are an OpenFlow-enabled Open vSwitch (OvS). A RYU SDN controller manages the network. In the beginning, a flow traversing the switches is initialised, which then triggers monitoring of the link between them. The latency of the link between the switches is varying and ranges from 15 to 50 ms, as can be seen in the later analysis with MPC. Since, in this network set-up, the controller gathers the telemetry data, it sends the delay measurement results to the broker.

The middle domain is based on the FD.io VPP telemetry solution [13]. In this domain, the telemetry data is encapsulated in the packet using an iOAM hop-by-hop header extension as it enters the network name-space. This action is executed by an iOAM encapsulation node. As the packet traverses the network the iOAM transit nodes add additional telemetry information. The telemetry data is removed as it is leaving the name-space, via an iOAM decapsulation node. The data is then polled by a telemetry collector allocated in the network and reported to the broker. The delay is being deducted from the Timestamp Trace type, which is a 32-bit value that represents the timestamp with ms accuracy. Moreover, if the MPC function provides for such an analysis, it is possible not only to monitor network conditions, but also to monitor the use of resources utilised to generate network traffic (e.g., a VR application). Figures 3 and 4 display two console screenshots illustrating the MPC execution.

Figure 3 presents the initialisation of domain 1. In Figure 4, a console view of the MPC execution is presented. The red boxes highlight the first iteration of the MPC execution. It can also be seen in the figure that only one of the three domains participating in the calculations gets the results of it. This is intended behaviour, as described in the code of the protocol execution available in Appendix A.

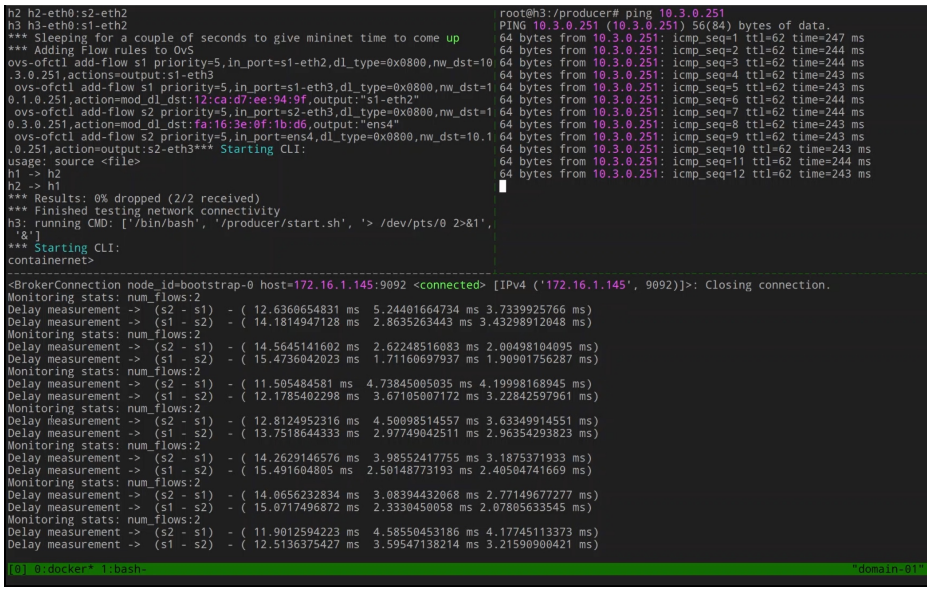


Fig. 3. Console view of domain 1 initialisation. The view includes three windows. Top-left represents the Mininet output, top-right the ping initialised by the end-user, while the bottom view presents the results from network telemetry.

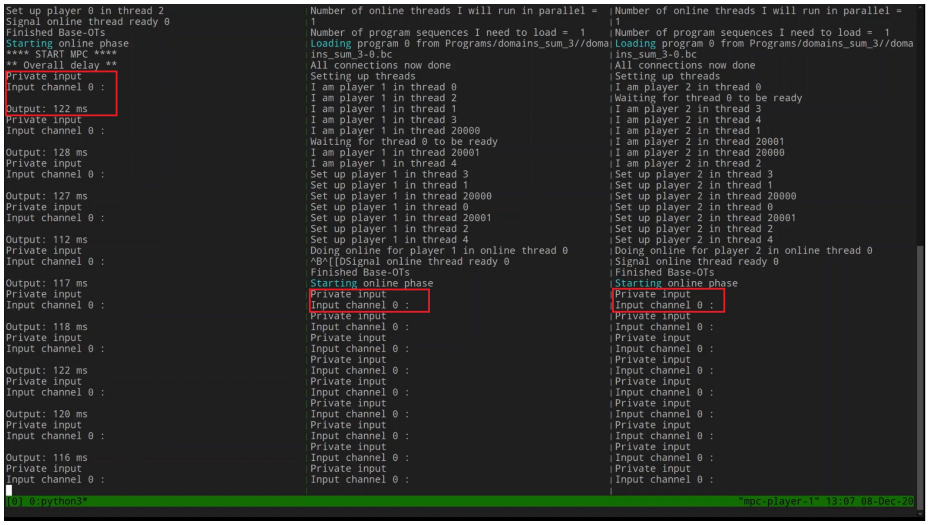


Fig. 4. Console view of the data analysis execution. The view includes three windows that correspond to the domains. In the red box, the first analysis iteration is marked.

3 Performance Analysis

3.1 Testbed

While the original PoC was developed using Docker containers, the performance analysis was conducted with Virtual Machines that provide better resource separation. This decision was made on account of the fact that SCALE-MAMBA is a framework with a high demand for resources. For a PoC this is fit for purpose, but for a real-life implementation, a scalable MPC framework should be selected or developed. For our experiments, three to eight virtual machines were used. The hardware resources allocated to each of the virtual machines were equal: four virtual cores E5-2683 of Intel Xeon CPU running at 2.1 GHz with 16384 KB cache and 8 GB of RAM. The network is organised in a star topology: the Virtual Machines are all connected to an instance of OvS that operates on the hypervisor. All hosts were connected to each other with a 15 Gbps emulated network interface (the bandwidth of the link was measured using iperf), and default link latency of approximately 0.461 ms with 0.081 ms standard deviation (link latency was measured using ping). The operating system of choice on each machine was Ubuntu 18.04.4 LTS with a 4.15 Linux Kernel. Additionally, all nodes were connected to a Network Time Protocol (NTP) Server - Stratum-1 resulting in a time synchronisation for the nodes ranging from -0.042 μs to 0.142 μs in relation to the NTP server.

3.2 Test scenarios

Three possible scenarios for the analysis of network data are considered. These three scenarios directly correspond to the types of description presented in Section 2.2:

- Calculating the aggregated sum of delays measured in each domain.
- Comparing data against predetermined values.
- Determining which of the domains participating in the protocol gives a maximum value for a targeted function (argmax).

The test applications available within SCALE-MAMBA assume that while the protocol can be executed between multiple parties, only one of them is responsible for the introduction of input data. To make the functions more realistic and correspond to the situation of analysing network telemetry data, each of the domains involved in the execution of the protocol inputs their own data during the execution. This means that one iteration of the function first takes data from each of the parties and then conducts a calculation defined within it. Sample code of the third function is included in Appendix A.

In the analysis, the influence of the following parameters on the execution of the protocol was checked:

- Number of domains
- Network latency
- Transmission rate
- Parallelization of the input data

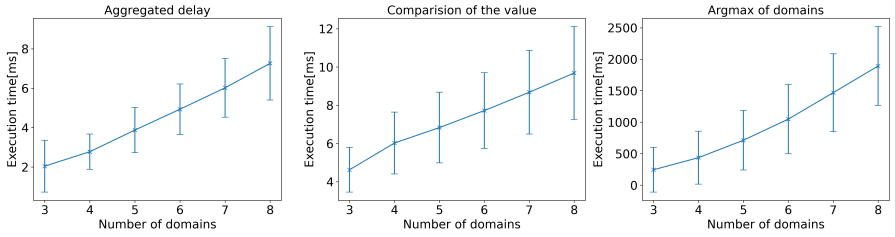
Their impact was determined based on the time to execute one protocol iteration.

The purpose of the analysis is to determine the performance and scalability of the part of the architecture responsible for performing computation on the data, but not the part responsible for the network measurements. For this reason, in order to determine the upper-bound for the performance, each of the control nodes responsible for initialising the function and providing data to the MPC protocol will produce a synthetic data point each time the protocol asks for the next value to the function. This achieves the necessary separation between the performance of the MPC framework and the performance of the telemetry technology.

3.3 Results

This section will present the results of our performance measurements. To the best of our knowledge, at the time of writing this paper, results of a similar nature have not been published. Existing results pertain to the performance of MPC frameworks and comprise calculations of a different nature, for example the multiplication of matrices, which makes direct comparison impossible.

Our results are presented as follows. First, the execution times of each of the three functions are shown, depending on the number of domains involved in the computation. Then one of the functions shows the impact of network delay, transmission rate, and parallelization of the input data.



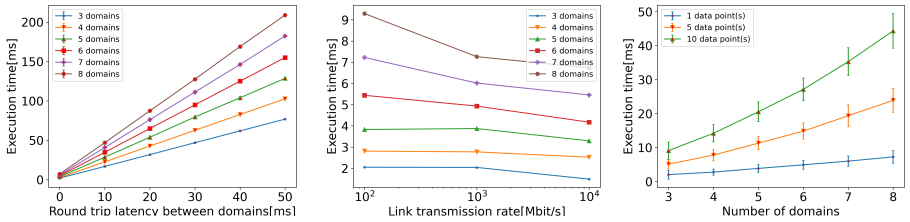
(a) (b) (c)

Fig. 5. The execution time in milliseconds in relation to the number of domains in case of three functions. The vertical bar represents the standard deviation.

Number of domains For the experiment, a range of 3 to 8 domains was used in order to check the change in execution time. The data presented in the figures were obtained by averaging the results over 10,000 consecutive iterations of the function.

The effect of increasing the number of domains on the execution time is presented in Figure 5. For all three functions, a higher number of domains results in a greater number of inputs and, as a result, an increase in communication between the parties, and hence an increase in the execution time of a single function iteration. The relation between the number of domains and execution time seems linear, which is desirable behaviour in terms of scalability. The execution time also illustrates the potential for real-time analysis of the data. For the first two functions, the execution time of a single iteration is between 2 and 10 ms, which shows that for simple functions, MPC allows making fast calculations. For more complex functions, such as argmax determination, the execution time is in seconds. Hence, in case of a real system for telemetry data analysis based on MPC, very complex functions can be used for the quasi-real-time analysis of events.

Network Latency The effect that network latency has on the execution time of a single iteration was tested calculating the aggregated delay. As in the previous experiment, the test results were obtained by averaging over 10,000 iterations. In order to obtain predictable values of the delay, we used the Linux tool *tc*. The latency in the test ranged from 0, which represents the initial latency on the testbed, to 50 ms. The value of the latency corresponds to the round-trip network delay between the domains. In Figure 6a, the effect on the execution time in relation to network latency is presented. As can be observed, the execution time grows significantly with increasing network latency. An important conclusion is that the network latency plays a key role in successfully deploying the MPC-based solution.



(a) (b) (c)

Fig. 6. (a) The execution time of a single function iteration in relation to the number of domains and network latency. (b) The execution time of a single function iteration in relation to the number of domains and transmission rate. (c) The execution time of a single function iteration in relation to the number of domains for three types of data parallelization presented on a single plot.

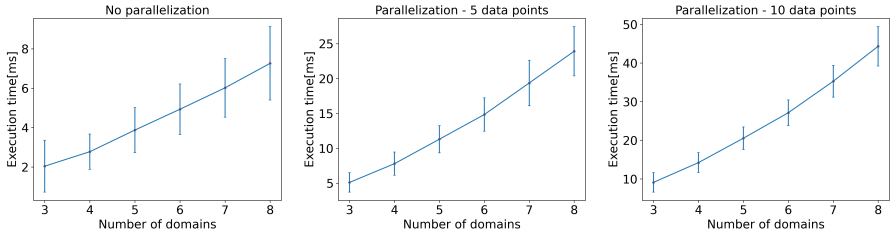
Transmission rate As in the case of network latency, the *tc* program was used to experiment with the transmission rate. The transmission rate was then verified using the *iperf* tool. The impact of the transmission rate is presented in Figure 6b (as the standard deviation ranged from approximately 1 ms to 4 ms, we omitted it for the sake of readability). Figure 6b illustrates that the transmission rate has no significant influence on the execution time.

Parallelization of input data To examine the influence of data parallelization, the function aggregating the delay over domains was modified in such a way that during a single function iteration, instead of conducting calculations on one data point, it is executed on five and ten data points. The effect of data parallelization in relation to function execution is presented in Figure 6c and Figure 7. As can be seen in the figures, while the parallelization of the input data increases the execution time of a single iteration, it results in more data points being analysed during that time. For example, "no parallelization" requires on average twice as much time to perform computation on 10 data points than is the case when inputting 10 data points at the same time. However, for very time-sensitive cases, this trade-off may not be possible.

4 Related work

Improving transparency should be the result of the efforts of many network domains. This section therefore presents related work in the context of secure data sharing.

SEPIA [9] is a solution based on secure multiparty computation to enable correlation and aggregation of network events, such as rule triggering in an Intrusion Detection System. In order to create their solution, the authors proposed their own protocol within which they developed a set of basic operations.



(a) (b) (c)

Fig. 7. The execution time of a single function iteration in relation to the number of domains for three types of data parallelization: (a) No parallelization, (b) parallelization of 5 data points, (c) parallelization of 10 data points.

The difference between SEPIA and the solution presented in this paper is that SEPIA discusses how to use the protocol in specific scenarios, such as correlation of events, but does not address the issue of increasing transparency from the end-user perspective.

The goal of GAIA-X [4] is to support the development aimed at achieving trustworthiness and sovereignty of digital infrastructure in Europe. This work also emphasizes the importance of transparency. Moreover, the GAIA-X requirements underscore the need for a decentralized system capable of secure data exchange. While GAIA-X creates a conceptual framework focused mainly on cloud solutions, our work proposes a modular solution for sharing descriptions and measurements originating from infrastructure processing the traffic, to users using it.

The purpose of SCION [5] is to allow the user to control the route that is selected by their traffic. This is achieved by introducing the Isolation Domain Concept, which is a logical presentation of a group of autonomous systems. The user, in the SCION architecture, is informed about possible paths, which allows him/her to choose the more appropriate one. As in SCION, our work aims to improve the lack of transparency in the current Internet structure. The difference is that we also allow the use of general descriptions not related to traffic processing, but resulting from the infrastructure used. Additionally, the implementation of the architecture presented in SCION requires agreements with the ISP(s) in case of creating Isolation Domains.

Finally, the Responsible Internet [10] is a proposal for sovereignty and transparency in the digital world. This visionary paper does not propose a concrete solution, but its concepts lie at the basis of our present work.

5 Conclusion

In this paper, we have presented a multi-domain diagnostic system as a means to improve Internet transparency. Our system leverages several technologies, like multi-party computation, software-defined networking, and iOAM, to realize an

overall design and proof-of-concept implementation. Via experiments, and in a multi-domain context, the execution time of our solution was evaluated based on various factors, such as variable number of domains, various functions analyzing data, and variable network parameters used for communication between the domains. The performance analysis demonstrates the feasibility of our approach, yet it does hinge on the resource-efficiency of the MPC framework of choice.

Acknowledgements This research was supported by the Netherlands Organisation for Scientific Research (NWO) under the CATRIN project and by the Netherlands Organization for Applied Scientific Research (TNO).

References

1. Apache kafka documentation. https://kafka.apache.org/documentation/, last accessed: Aug. 13, 2020
2. Rabbitmq. https://www.rabbitmq.com/, last accessed: Apr. 15, 2020
3. Zeromq. http://zeromq.org/, last accessed: Apr. 17, 2020
4. Gaia-x. https://www.data-infrastructure.eu/GAIAX/Redaktion/EN/Publications/gaia-x-the-european-project-kicks-of-the-next-phase.pdf (2020)
5. A.Perrig, P.Szalachowski, R.L.: SCION: a secure Internet architecture. Springer (2017)
6. A.Rastogi: Wysteria: A programming language for generic, mixed-mode multiparty computation. https://bitbucket.org/aseemr/wysteria/wiki/Home (2014), last accessed: Mar. 26, 2020
7. B.Mood: Frigaterelease. https://bitbucket.org/bmood/frigaterelease/src (2020), last accessed: Sept. 30, 2020
8. Brockners, F., Mizrahi, T., Bhandari, S.: Data fields for in-situ oam (07), https://tools.ietf.org/html/draft-ietf-ippm-ioam-data-10, last accessed: Aug. 13, 2020
9. Burkhart, M., Strasser, M., Many, D., Dimitropoulos, X.: Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. Network **1**(101101) (2010)
10. C.Hesselman, P.Grosso, R.F.F.J.M.J.R.A.R.D.G.o.: A responsible internet to increase trust in the digital world. Journal of Network and Systems Management **28**(4), 882–922 (2020)
11. Cloud Security Alliance: Csa star framework. https://cloudsecurityalliance.org/star/levels/, last accessed: Dec. 17, 2020
12. ENISA: Insider threat. https://www.enisa.europa.eu/topics/cyber-threats/threats-and-trends/etl-review-folder/etl-2020-insider-threat (2020), last accessed: Aug. 13, 2023
13. Mauricio, S.J.: Further Implementation of iOAM using IPv6 in FD.io Vector Packet Processor. Master's thesis, Department of Electrical and Computer Engineering, Technische Universität Kaiserslautern (2020)
14. N.Buescher: Cbmc-gc-2. https://gitlab.com/securityengineering/CBMC-GC-2 (2018), last accessed: Mar. 24, 2020
15. Smart, N.: Scale-mamba software. https://homes.esat.kuleuven.be/~nsmart/SCALE/, last accessed: Aug. 15, 2020

16. Van Den Berg, D., Glans, R., De Koning, D., Kuipers, F.A., Lugtenburg, J., Polachan, K., Venkata, P.T., Singh, C., Turkovic, B., Van Wijk, B.: Challenges in haptic communications over the tactile internet. IEEE Access **5**, 23502–23518 (2017)

A MPC code details

This appendix contains some code that is part of the Proof-of-Concept implementation.

```
4
RootCA
7
10.0.1.192
Player0.crt
10.0.1.141
Player1.crt
10.0.1.105
Player2.crt
10.0.1.175
Player3.crt
10.0.1.196
Player4.crt
10.0.1.29
Player5.crt
10.0.1.201
Player6.crt
2
9223372036855103489
1
```

Listing 1.1. A sample setup file used to configure the SCALE-MAMBA framework.

```
print_ln('**** START MPC ****')
print_ln('**** Max delay ****')

def maximum(a,b):
    return a - ((a - b) & (a < b)) #SCALE returns -1/1 when comparing values

def argmax (a,b):
    return (a[0] - ((a[0] - b[0]) & (a[0] < b[0])) ,
          ((a[0] < b[0]) & b[1]) | ((a[0] > b[0]) & a[1]))

@while_do(lambda x: x < 1, 0)
def cal_max_delay(i):

    a = (sregint(sint.get_private_input_from(0)),sregint(0))
    b = (sregint(sint.get_private_input_from(1)),sregint(1))
    c = (sregint(sint.get_private_input_from(2)),sregint(2))

    max = argmax(a,b)
    max = argmax(max,c)
```

```
    print_ln('Output: value of max delay is caused
                by Player %s', max[1].reveal())

    return i + 1

if __name__ == '__main__':
    cal_max_delay(0)
```

Listing 1.2. Code for an MPC function calculating the argmax of the presented values. This function is written for an environment of three domains.