Open Security Issues for Edge Named Function Environments

Michał Król[1], Claudio Marxer[2], Dennis Grewe[3], Ioannis Psaras[1], and Christian Tschudin[2]

[1]University College London, United Kingdom
[2]University of Basel, Switzerland
[3]Robert Bosch GmbH, Corporate Sector Research, Renningen, Germany

Abstract

In this article, we focus on *in-network function execution* in Information-Centric Networks (ICN) and we highlight the open issues for secure distributed computing given an ICN substrate. Within such an ICN-based ecosystem, we present an exemplary use-case from the autonomous automotive IoT area, namely, localised sensing. Based on this use-case, we identify security challenges, discuss the different options to tackle these challenges and provide directions for future work.

I. Introduction

There is a constant stream of evidence that the centralized, data-center model of cloud computing appears inferior in light of emerging applications. The Internet of Things (IoT) is reversing the data flow from the *edge to the core* (as opposed to the current core-to-edge data flow) with much of the data produced at the edge requiring processing before being sent to the back-end cloud. At the same time, autonomous vehicles and Virtual/Augmented Reality applications demand for minimal latency. In some cases this means sub-ms response times which cannot be satisfied by classic cloud-based solutions.

As a result, recent research efforts have focused on distributed cloud (or cloudlet) infrastructures, where applications (or functions) are executed in specialized execution nodes at the edge of the network [1]. Such infrastructures can provide lower delay, minimize the traffic that needs to travel all the way to the back-end data center, and reduce the exposure to failures of a single cloud provider.

As a representative example, connected vehicle systems will depend on information sharing with other vehicles, the surrounding infrastructure and cloud backends. One of the big game-changers in the automotive industry is the introduction of automated driving which will heavily rely on timely information and computation of results (*e.g.,* proximity sensing for traffic flow coordination to avoid accidents). To ensure the functional operation of applications in autonomous vehicle environments, execution in the cloud does not meet some of the requirements (*e.g.,* very low response requirements), while execution at the edge is a promising solution. In all cases data exchange in such large scale vehicle-to-everything (V2X) networks is a challenge due to the high degree of mobility of participants, among other reasons. Today, the host-centric communication model challenges the network at many levels such as the management level by maintaining address changes while mobile participants switch multiple times between access points. This results in many handovers, requires frequent updates of host addresses as well as the data delivery routes, in order to discover the closest computing node. Information-Centric Networks present a promising solution with regard to consumer mobility. At the same time, resolution of the edge computing node's IP address (through DNS) introduces excessive delay. Instead, native, network layer "route-by-name" resolution avoids an extra RTT to the DNS and has the potential to reduce significantly response times.

Based on a loosely coupled communication model, Information-Centric Networks (ICN) advocate against a host-centric communication model. The paradigm uses *content-identifiers* such as naming schemes which simplifies discovery and the direct access to data and offers mobility support by nature [2]. Therefore, the paradigm can be a great fit for provider-agnostic distributed clouds: *it does not matter where and by whom an application/function is executed, as long as the result is correct, valid, verified and can be trusted.* This is in stark contrast to a host-centric edge-computing environment where edge devices need to connect to some specific IP address operated by a trusted entity (*e.g.,* redirected from the cloud).

Despite its conceptual fit, the vast majority of ICN approaches to date focus on naming, routing/forwarding and distribution of static content. As we move towards a distributed, edge-computing environment, there is an urgent need to define a *"computation-centric architecture"* to support decentralised and distributed computation.

Named Function Networking (NFN) [3] and Named-Function as a Service (NFaaS) [4] are two exemplary developments towards dynamic in-network computations in ICNs. NFN and NFaaS extend ICN to execute functions/services at any element in the network such as intermediate/edge routers and end-user devices. Fig. 6 illustrates the differences between static content delivery and named functions. In static content delivery, nodes simply forward Interests to fetch Data messages from the network. Instead, in case of function invocation, consumers can request computations by describing them in the Interest (**I1**) name (NFN uses λ-expressions, while NFaaS hierarchical names). The requests are automatically forwarded to the most suitable execution node and are decomposed into subroutines (**I2**) outsourced to additional nodes if needed. The execution nodes can request missing function code or additional input by issuing Interests as with static content delivery (**I3, I4**). The execution nodes finally return the computation result encapulated into Data messages back to the requestors (**D1**).

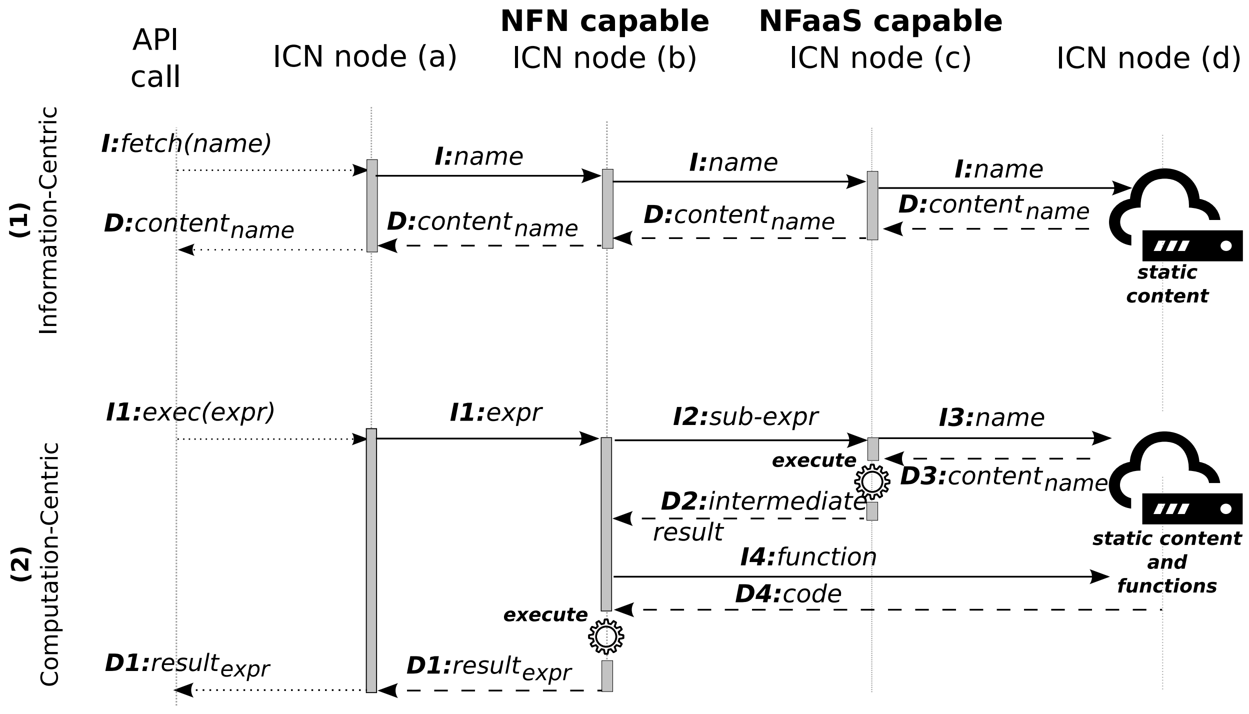

Fig. 1: Differences between static information-centric networking (top part of Fig.) and named function execution (bottom part of Fig.). While in ICNs content is fetched from the network, in computation-centric architectures the network is able to execute computations.

A computation-centric architecture has the potential to increase system-wide performance considerably, but comes with new security challenges. That is, instead of relying on a central cloud provider's reputation, effectively forcing users to trust its service, mechanisms must be put in place to preserve privacy and at the same time make it impossible for the edge-infrastructure providers to cheat or misbehave.

Although this is not impossible to achieve, there are open issues that require further attention. In this paper, we build on the concept of *"Named Functions"* [3], [4] and identify security vulnerabilities of computation-centric environments that are not present when dealing with static-content in traditional Information-Centric Networking environments. We illustrate the need for a computation-centric edge computing architecture through an autonomous vehicle setup, where localised sensing information is processed in the network and retrieved by fast-moving cars. The core contribution of this work is as follows: for the above-mentioned autonomous vehicle scenario, we identify five high-level security challenges, which we split into eight technical problems. For each of the problems we provide mitigation approaches and point to issues that need further work.

II. Use Case: Localised Sensing for Autonomous Vehicles

In-network function execution can improve the performance in many areas and named functions can enable services not possible to achieve with IP. Constrained IoT devices requesting computations on high-volume data, augmented reality with low-delay video processing or smart buildings can benefit from bringing computation from the cloud to the edge. In this work, we focus on automated driving as a representative example with resource-heavy computations, large data sets and high user mobility, but the discussed issues can be applied to other areas as well.

A. Use-Case Overview

Automated driving relies on information such as very detailed maps about the vicinity, in order to avoid collisions. Vehicular systems as well as intelligent road infrastructure will be equipped with sensors (*e.g.,* front/rear/blind spot cameras, radar systems or brightness sensors) to monitor the environment as well as connectivity units to be able communicate with each other as well as with edge infrastructure components [1].

We illustrate the *"localised sensing"* use-case in Fig. 7. An approaching vehicle (*e.g.,* green vehicle) requests a report for the next geo-specific area$_2$ where it will arrive shortly. The report should contain a list of potentially dangerous events created by analysing information gathered from infrastructure and sensors located in the area. For instance, the red car coming out

[1]https://newsroom.intel.com/editorials/krzanich-the-future-of-automated-driving/

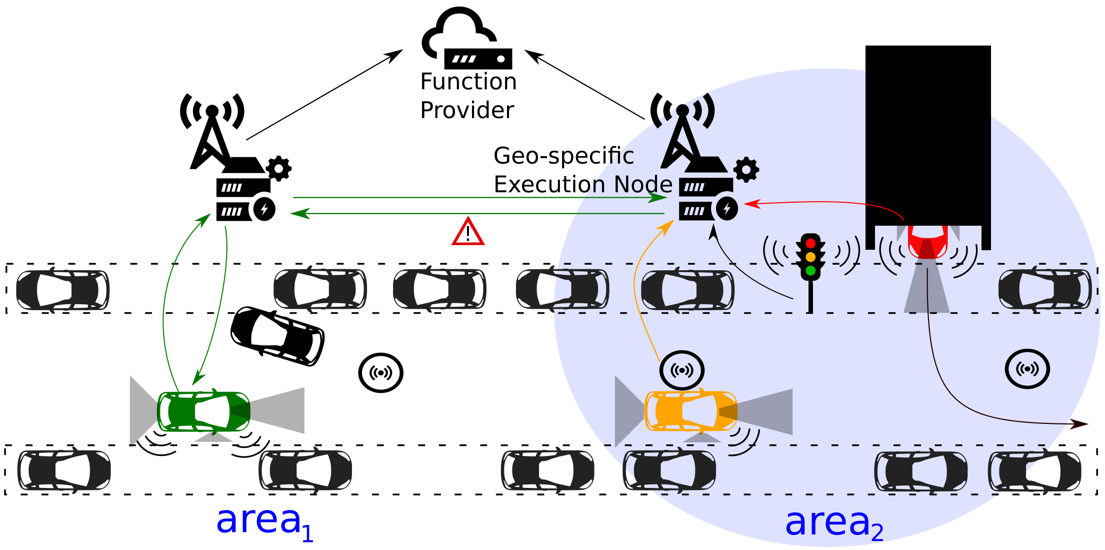

Fig. 2: Exemplary scenario of the localized sensing use case.

of the driveway will be detected and an alert will be sent back to the approaching vehicle to slow down or stop. There are several ways to deal with the situation:

- **Traditional Cloud Processing.** Processing of large amounts of proximity data, comes with strict response deadlines. Acceptable response delays can be as low as a few milliseconds. For those cases, sending everything up to the cloud requires excessive amounts of bandwidth, but most importantly induces prohibitive round-trip latency [5].
- **Direct Car-to-Car Communication.** Although one might argue that this type of information can be communicated directly between vehicles, this is not always the case. For instance, in non-residential roads vehicles are likely to move with speeds that are prohibitive for direct car-to-car communication (*i.e.,* vehicles move faster than proximity sensors can connect and communicate between them).
- **In-Car Processing.** Exchanging raw data between cars (either through car-to-car communication, or through relays) and processing data "in-car" can be another option. However, in-vehicle processing capabilities are restrained due to limited on-board compute capabilities [2]. Furthermore, each car would have to gather the required information and generate its own report leading to highly inefficient resource usage.

In contrast, a network that supports execution of in-network functions can assist the vehicle as well as the cloud infrastructure by aggregating road sensor data at nodes with sufficient computational power placed at the edge of the network. Processing data closer to the required geo-location reduces latency and the load in the core network.

B. Environment Overview

We assume that fast moving cars do not have routable prefixes and act only as consumers. In this way, we protect users privacy and avoid producer mobility problem that is considered to be difficult to solve in ICN. Functions are invoked by location independent names in a form *execute(function)* specific to the underlying function invocation system. To obtain location specific information from the road infrastructure, we adopt a scheme proposed in [6], where different size geographic areas can be easily translated to hierarchical names.

Named functions are used to gather, process and filter information about a specified area. When a vehicle sends a request for a report, the network orchestrates the corresponding computation on a local execution platform. This procedure includes choosing a suitable node in the given geo-location, splitting the task into sub-computations and integrating intermediate results into a final reply, where necessary. Following the loosely coupled communication model of the computation-centric paradigm, function execution is not rigidly tied to a certain location, but can take place close to functions, data or end-users. Furthermore, consecutive requests to generate a report for the same area and time interval can be satisfied by already computed results cached in the network.

By introducing named functions in the *localised sensing* use case, the following actors are involved (Fig. 8):

- **Requestor:** client requesting computation from the network. In our case, requestors are autonomous vehicles.
- **Function Provider:** provider of named functions. In our use-case this might be the traffic regulation authority that provides the software that runs on edge nodes.

[2]http://www.nordsys.de/en/car2x-produkte-2.html

- **Orchestration and Execution Node:** a node involved in orchestrating a distributed computation and/or actually performing the application of functions.
- **Information Supplier:** provider of primary *i.e.,* sensor data from which computation results are derived. In our use-case, intelligent infrastructure and sensors in the vicinity are providing information when requested by an execution node.

Each of the roles above can be under the control of one or different entities. For instance, it is possible, that the function provider and the execution node will reside on the same physical location while exchanging information. They should not need to have a pre-established trust relation, but the correct process flow should be guaranteed by the architecture and associated mechanisms.

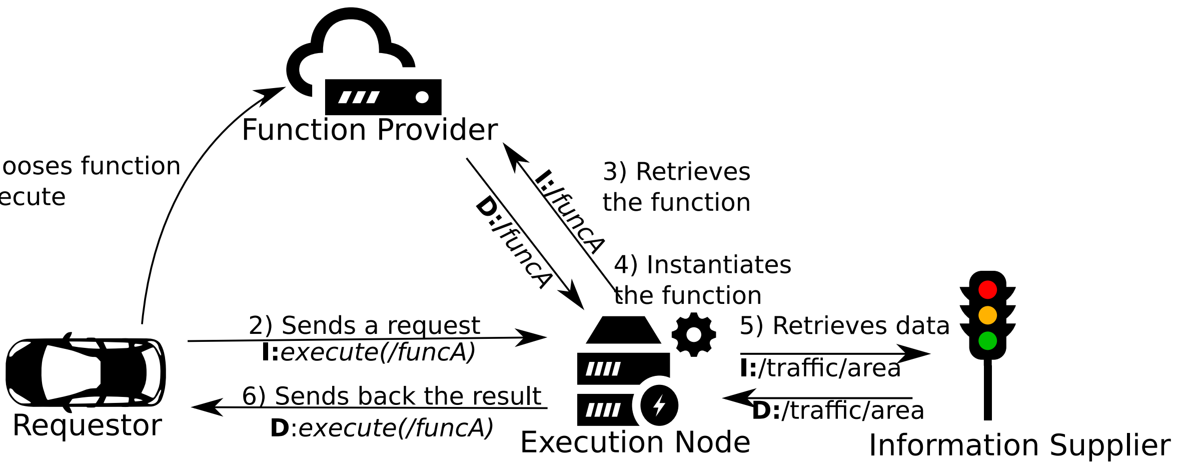

Fig. 3: System interactions.

III. Open Issues and Suggested Solutions

Assuming the described environment, we identify the following high-level security challenges. We then go on to discuss in detail 8 distinct security problems that emerge out of these 5 areas:

C1 **Requestor Identity and Authentication** (Section III-A): access must be restricted to authorized consumers. To defend against Denial of Service (DoS) attacks, an execution node should be able to identify requestors and determine whether they have the rights to invoke requested computations.

C2 **Secure Input Submission** (Section III-B): the requestors should be able to submit input parameters (such as credentials or speed/direction information) in a way that does not open additional attack surfaces in the network.

C3 **Privacy and Confidentiality** (Section III-C): submitted input parameters and returned results must be encrypted and protected from intermediate nodes, as well as the execution nodes themselves.

C4 **Result Correctness** (Section III-D): requestors must be able to verify if a remote execution node executes the requested tasks and that the returned result is correct.

C5 **Automated Interoperability** (Section III-E): various security mechanisms implemented by different vendors need a common language to interact in an decentralized computing environment.

A. Requestor Identity and Authentication

Problem 1: An execution node receiving a request must verify the identity of the requesting vehicle to protect against DoS attacks (e.g., nodes attacking the network with invalid requests) and to allow for accounting and billing.

In ICN, nodes sending Interests are not identified in the packet. Instead, an Interest leaves a trail of entries in the Pending Interest Table (PIT) allowing the data to be returned using the same path. This protects privacy and allows PIT entry aggregation, but makes identity recognition a challenging task. In case of static content, producers and caches can serve the content to each consumer and realise access control by encrypting the content for authorized users [7]. However, applying the same approach to a function execution scenario would expose the system to costly Denial of Service attacks (*i.e.,* malicious requestor asking for dump computations to consume CPU capacity of execution nodes).

The most common solution to identify requestors consists of signing Interests using their private key. This approach is already used in Named Data Networking (NDN) for localhost routing commands[3]. However, this solution cannot be applied in open communication, as the Interest can be overheard by a malicious node and used to invoke unnecessary computations

[3]http://named-data.net/doc/ndn-cxx/current/tutorials/signed-interest.html

(replay attack). This situation is difficult to avoid, as the packet does not differ from the valid one. A timestamp and interest expiry time included in the signed part of the packet alleviate the problem, but does not represent a complete solution.

To reliably identify a requestor, classic IP networks, use various cryptographic algorithms (*i.e.,* Diffie–Hellman key exchange). However, cryptographic algorithms require exchanging multiple packets and establishing a session between the two nodes, using their respective IP addresses. Vanilla ICN, however, allows only for session-less Interest-Data exchange, without specifying source-destination identifiers. An additional mechanism to maintain a session between two nodes would allow for easy adoption of already verified authentication solutions. While some mechanisms for session establishment in ICN have been proposed [8], none of them has been widely adopted or analysed in context of secure function invocation. Furthermore, a session establishment mechanism can suffer from problems typical to endpoint-centric communication such as easy DoS attacks.

B. Secure Input Submission

Problem 2: If a computation requires input data from the requestor, the latter needs a secure way to submit the missing information.

The ICN pull model of operation demands that data can flow only from the producer towards the consumer, making sure that nodes receive only data that they requested. Therefore, if a computation requires additional input to perform the requested computations, the execution node needs to fetch the additional data from the user. A fast moving car cannot act as a producer with a globally routable prefix as it changes its position too quickly. Furthermore, producer (*i.e.,* vehicle in our case) mobility is a problem that still has no widely adopted solution in the ICN area, requires additional signalling and introduces significant overhead. The requestor could create ephemeral, on-demand FIB entries pointing towards him when sending an Interest, but allowing an untrusted node to manipulate forwarders' state opens new ways to perform flood attacks with entry creation requests. Moreover, pulling data by the execution platform introduces additional delay that, in contrast to static data retrieval, can be unacceptable by multiple applications such as Augmented Reality. On the other hand, directly attaching large payloads to the Interests exposes nodes to Denial of Service attacks, as they can be overwhelmed with large volumes of data they did not request.

Problem 3: If a function requires external information to perform the requested computations, the execution node will issue interests to fetch the missing data. A malicious requestor could exploit this mechanism to generate huge amounts of additional traffic in the network (reflection attack).

Interest flooding attack is a serious problem in ICN networks with multiple proposed countermeasure mechanisms [9]. However, the majority of the mechanisms focus on punishing the node/link issuing the interests (*i.e.,* by dropping some interests). In function execution, this problem is even more pronounced and more difficult to solve, as a malicious requestor can hide behind one or multiple execution nodes (Fig. 9). If the attack is distributed among hundreds of nodes, local mechanisms will not be able to detect it. At the same time security mechanisms deployed in the network, when discovering the attack can limit the bandwidth allocated to the execution nodes. This would in turn decrease the performance of other computations being executed on the same machines.

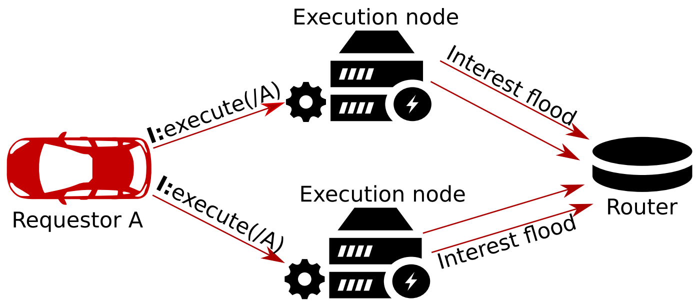

Fig. 4: Reflection attack.

The interests issued by the execution nodes, should be thus identified as sent by the requestor. At the same time we must protect requestor's privacy and limit the possibility of a malicious execution platform to perform its own interest flood attack using the requestor's identity.

C. Privacy and Confidentiality

Problem 4: When the requestor sends an Interest with an input and later receives data packets containing the results, the packets are forwarded by multiple untrusted nodes. The information both inside the Interest input and the resulting data should be encrypted to protect it from intermediary nodes and the execution node.

In static data encryption, the producer owns the data and can read it before encrypting. This changes when we outsource computations. The requestor wants to hide the data not only from the network, but also for the execution platform. Recent advances in Trusted Execution Environments (TEE) technologies allow applications to create private parts of memory called enclaves. Enclaves are protected from the other applications, operating system or hypervisor [10]. Functions can thus acquire or generate data encryption keys and securely store them in the host's memory.

If all the computations are done in an enclave, they are invisible to the execution node. Encrypting the data produced by a function for transport is then straightforward: Interests invoking computation can include the public key of the requestor and this key can be used to encrypt the data. However, securing the input itself (*i.e.,* information in the Interest and related function input fields) is less trivial. The network can forward the request to any node for execution; it is thus impossible to establish a secure connection in advance. In the ideal case, the requestor would like to use the function's public key for encryption of the input, in order to take full advantage of function-centric communication. To achieve that, the public key should be obtained from the function provider before invoking the computation (Fig. 10). Next, in order to decrypt the input, the function must possess the corresponding private key. The function obtains the private key from the function provider using an encrypted channel protected from the execution node. Once, the function decrypts the input, it can still leak it to the execution node or the function provider, but such behaviour can be prevented by verifying the correctness of the function (Sec. III-D).

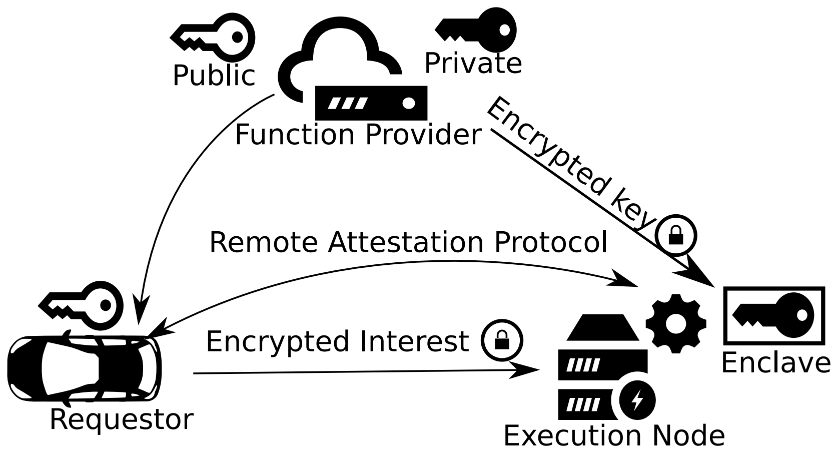

Fig. 5: Using Intel SGX to verify the function and encrypt Interests using private/public key scheme.

Relying on TEEs provides additional security but requires users to trust hardware vendors. Furthermore, TEEs use classic IP communication and exchange multiple messages. Their adaptation to pure ICN networks would require consumers to establish sessions with corresponding execution platforms. However, enhancing ICN with sessions must be done carefully in order to preserve the main advantage of information-centric communication (*Problem 1*).

Problem 5: As distributed in-network processing – aggregation, conversion, filtering – plays an essential role in our use case, access control policies and mechanisms need to handle results of distributed computations and therefore must be logically distributable themselves.

Access control is about restricting data access to authorized principals only. ICN advocates for content-based security (instead of securing connections), management of encryption/decryption keys is thus the essential task in performing access control. A common style for static data is the differentiation of Data Encryption Keys (DEK) and Key Encryption Keys (KEK) [11]. Each access controlled document is encrypted with a symmetric DEK and authorized principals hold a KEK to subsequently unwrap DEK and the document.

However, in-network content aggregation can involve access controlled content from multiple independent sources (*e.g.,* all sensors within a geo-location) which are fusioned in a distributed result calculation process. This increases complexity of access control because (i) policies from all sources (*e.g.,* road sensors) must be applied to an on-demand content object, while (ii) ensuring that consumers can not bypass access restriction by gaining information from computations results. In [12] and [13], the authors propose to explicitly publish access policies for restricted documents. These feed the result production process for policies of computation results, and thus (recursively) act as a decision basis for access control checks.

Furthermore, to avoid the network carrying out a certain computation multiple times, re-usability of cached results must not be affected by access control solutions. An approach is to decouple (in terms of time and location) the mechanism which produces a computation result and the one synchronizing the applied DEK. This way, the DEK for a certain result can be requested by many consumers even if their requests are not simultaneous and/or reach different execution locations. To do so, key derivation procedures must be in place which (i) generate deterministic/reproducible DEKs, but (ii) can not be successfully carried out by any party which is not authorized to read the output DEK. The authors in [12] propose to deploy an algorithm whose output key is fully determined by the DEK(s) of the data from which the thereby secured result is derived.

Finally, given that the computation provider needs to read a computation's plain input, there need to be trust relationships between all input information suppliers and the executing computation provider. For computation orchestration and/or propagation of interest in access-controlled computation results, this must be taken into consideration. Practically, only few computation providers might satisfy these requirements to enhance named function orchestration and forwarding machinery in this regard. We consider this as a rather challenging task especially if data from many sources are aggregated.

D. Correctness of Functions and Results

Problem 6: When a requestor chooses a function to execute, he must ensure that the function is authentic and will perform the required computations and will not leak any private data.

Verifying a function is a two-step process. First, when the function is published by the function provider, its behavior must be verified (if it is open source) or trusted (if its source code is closed). Second, the downloaded and instantiated function must be checked before every execution to verify if it actually represents the chosen function.

Open source functions can be verified by the community, assigned a rank or be placed on whitelists that requestors or execution nodes trust. This step must be performed only once and does not influence the overall performance of the system. A request to invoke an unknown function should be rejected. If the function is trusted, the corresponding image can be requested from the network. Its authenticity can be easily verified using signatures before instantiating as in static content retrieval.

However, it is more difficult for the requestor to verify the integrity of the instantiated function. A malicious function provider could create a function with the same name and private keys as the legitimate one, but with a different behaviour. This is because the requestor cannot reliably verify the signature of a function run on a remote node.

To help with this problem, Intel recently released the Remote Attestation protocol as a part of its TEE environment called Intel SGX [10]. Using the Remote Attestation flow, a function can attest to a requestor that it is trusted, and establish an authenticated communication channel with that entity. As part of attestation, the function confirms its identity, verifies that it has not been tampered with, and proves that it is running on a genuine platform enabled with Intel SGX. However, using this technology currently requires involving an external attestation server and exchanging multiple packets.

Problem 7: There is a threat that a malicious computation provider generates fake results in order to save resources (i.e., execute a lightweight function instead of the specified one or simply returns random bits). Therefore, the correctness of each result must be verified by each consumer.

In case of static content, if the requestor trusts the publisher, checking the integrity of static data can be easily done using signatures. That is, a produced chunk of data is signed by the publisher using its private key and can be verified using the corresponding public key later on. If the content is modified in the network, as is the case with in-network functions, the signature will be invalid, since the node that alters the data (by applying functions on the data) does not have the publisher's private key.

In case of function execution, *function providers* cannot directly sign the produced content, as the function is executed on remote hardware. Since potentially any node can download a function and become an *execution node*, they cannot be authorized to sign the generated content and the data should be signed by the function itself. To implement this, the function must have access to the private key of the provider that can be used for signing function-produced data. Similarly to input parameters encryption, secure memory enclave could store the key and use it to sign generated results. The result can be then cached in the network, retrieved and verified by other consumers without needing to execute the function.

E. Automated Interoperability

Problem 8: It is expected that due to the diversity and evolution of the information supplying ecosystem, a heterogeneous security solution landscape will be shaped. Automated tuning of a data consumer's security machinery is therefore essential for versatile car-to-environment coupling.

In applied security, practical questions like the followings arise: *Which are the specific procedures for decryption, signature and provenance verification? In which modes do they operate? What are conventions of the specific solutions to access keys and certificates?* In a M2M context, it is impossible to assume that humans intervene to answer such questions. On the other side, for the sake of flexibility, a solution should not rely on implicit namespace conventions and agreements which must be configured permanently into consumption logic.

On the contrary, to allow flexible combinations of data and functions, transparency for consumers must be enhanced and fully automated consumption of arbitrarily secured content must be enabled. First steps into this direction are made regarding content authenticity in [14] and access control in [13]. Both works adopt the approach to *schematise*, *i.e.,* explicitly publish, signing authority capabilities and read access rights of principals. In [15], the authors focus on the operational aspects and

TABLE I: Summary of identified problems with potential solutions and open issues

Problem	Potential Solution	Open Issues
1) Requestor Identity	Secure Session Establishment	Endpoint communication problems
2) Input Submission	Ephemeral FIB	Protection against flood attacks; Minimizing the delay
3) Reflection Attack	Identify Interest Origin	Requestor privacy; Usage of a valid Interests by malicious parties
4) Information Encryption	Encryption keys stored in TEE	No TEE for ICN, hardware vendor trust issues
5) Access Control	Explicit policies, key and policy derivation procedures [12] [13]	Trust management: placement of computations on eligible nodes
6) Function Verification	Remote Attestation Protocols	No attestation protocols for ICN, hardware vendor trust issue
7) Result Verification	Function Provider's keys stored in TEE	No TEE for ICN, hardware vendor trust issues
8) Automated Interoperability	Schematised security policies [14] [13], explicit procedures [15]	General-purpose language to express consumption logic

describe a "trust engine" which makes automated signing and verification logic feasible. Although these are not comprehensive solutions they show that fully automated, flexible and secure production/consumption of named data is beneficial and feasible, but still require further effort in order to be applied to a distributed computing environment.

F. Summary

The transition from centralised and trusted cloud infrastructures to a federation of untrusted nodes at the edge of the network opens many new challenges. Security mechanisms are thus an essential step in constructing a trusted system from untrusted parts. We summarise the identified challenges along with potential solution approaches and unsolved issues in Table I. Although many well-tested mechanisms are already deployed in IP networks, their adaptation to computation-centric networks is an essential step in order to support edge computing setups. That said, on-going research in the area is of great importance and represents a significant step in closing the gap between well-tested host-centric approaches and emerging computation-centric environments.

IV. Conclusion

We discussed open security research challenges for the secure distribution and execution of in-network functions within Information-Centric Networks. Based on the introduction of an autonomous automotive use case – localised or proximity based sensing – the paper focused on the following hot security challenges: (i) Consumer Identity & User Authentication, (ii) Secure Input Submission, (iii) Privacy and Confidentiality, (iv) Correctness of Functions and Results as well as (v) Automated Interoperability. By analyzing the related work, we presented future research directions in the context of secure distribution and execution of named functions at the edge of the network. Also we highlighted unsolved issues which we see as important topics for future work.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[3] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, "An information centric network for computing the distribution of computations," in *ACM ICN'14*, pp. 137–146, 2014.

[4] M. Król and I. Psaras, "Nfaas: Named function as a service," in *ACM ICN'17*, pp. 1–11, ACM, 2017.

[5] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, "Information-centric mobile edge computing for connected vehicle environments: Challenges and research directions," in *Proceedings of the Workshop on Mobile Edge Communications*, 2017.

[6] D. Pesavento, G. Grassi, C. E. Palazzi, and G. Pau, "A naming scheme to represent geographic areas in ndn," in *Wireless Days (WD), 2013 IFIP*, pp. 1–3, IEEE, 2013.

[7] M. Ion, J. Zhang, and E. M. Schooler, "Toward content-centric privacy in icn: Attribute-based encryption and routing," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pp. 39–40, ACM, 2013.

[8] M. Gasparyan, G. Corsini, T. Braun, E. J. Schiller, S. de Arco, and J. Eduardo, "Session support for scn," 2017.

[9] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *IFIP Networking Conference, 2013*, pp. 1–9, IEEE, 2013.

[10] S. Gueron, "A memory encryption engine suitable for general purpose processors.," *IACR Cryptology ePrint Archive*, vol. 2016, p. 204, 2016.

[11] Y. Yu, A. Afanasyev, and L. Zhang, "Name-based access control," tech. rep., IRL UCLA, January 2016.

[12] C. Marxer, C. Scherb, and C. Tschudin, "Access-controlled in-network processing of named data," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, (New York, NY, USA), pp. 77–82, ACM, 2016.

[13] C. Marxer and C. Tschudin, "Schematized access control for data cubes and trees," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, (New York, NY, USA), pp. 170–175, ACM, 2017.

[14] Y. Yu, A. Afanasyev, D. Clark, V. Jacobson, L. Zhang, *et al.*, "Schematizing trust in named data networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*, pp. 177–186, ACM, 2015.

[15] C. Tschudin, E. Uzun, and C. A. Wood, "Trust in information-centric networking: From theory to practice," in *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, pp. 1–9, IEEE, 2016.

V. Authors

- Michał Król (m.krol@ucl.ac.uk) is a research associate at the Electrical and Electronic Engineering Department of UCL. He received his Ph.D. degree in computer science from University Grenoble-Alpes, France in 2016. His research focus is on the next-generation Internet architecture, niformation-centric connectivity and secure device-to-device communication. His current research interests include a variety of topics ranging from distributed ledgers, blockchain, trusted execution environments and network security.
- Dennis Grewe (dennis.grewe@de.bosch.com) is a Ph.D. candidate in Computer Science at University of Koblenz-Landau, Germany. He received his B.Sc. in Media Informatics (2013) and his M.Sc. in Computer Science (2015) from the University of Applied Science Stuttgart (HdM). His current research interests include information-centric network architectures for vehicular systems with respect to data dissemination and data caching as well as data/service description to be able to describe software-based automotive services.
- Claudio Marxer (claudio.marxer@unibas.ch) is a Ph.D. candidate in the Computer Networks Group at University of Basel, Switzerland. He received his M.Sc. in Computer Science from University of Basel. He is conducting research in information-centric and named function networking. Currently his focus includes access control for in-network computation results as well as organisation of named data.
- Ioannis Psaras [M] (i.psaras@ucl.ac.uk) is an EPSRC Early Career Fellow and Lecturer of Computer Networks at the Electrical and Electronic Engineering Department of UCL. His interests are in the areas of Internet routing and congestion control, Information-Centric Networks, Edge- and Fog-Computing and Mobile, Opportunistic Networks. Lately, he has been investigating the applications of Distributed Ledger Technology and Blockchains to networking problems. He has received four (4) Best Paper Awards for his contributions to high-quality conferences and workshops, all in the area of Information-Centric Networks and mobile communications.
- Christian Tschudin [M] (christian.tschudin@unibas.ch) is a full professor at the University of Basel and leads the Computer Networks Group. Before joining the University of Basel, he was a PostDoc at ICSI in Berkeley and assoc prof at Uppsala University, Sweden. He has a diploma degree in Mathematics and earned a computer science Ph.D. at the University of Geneva. Regarding information centric networking, he is the main author of CCN-lite and introduced the concept of Named-Function Networking as a complement to NDN.

VI. Figures

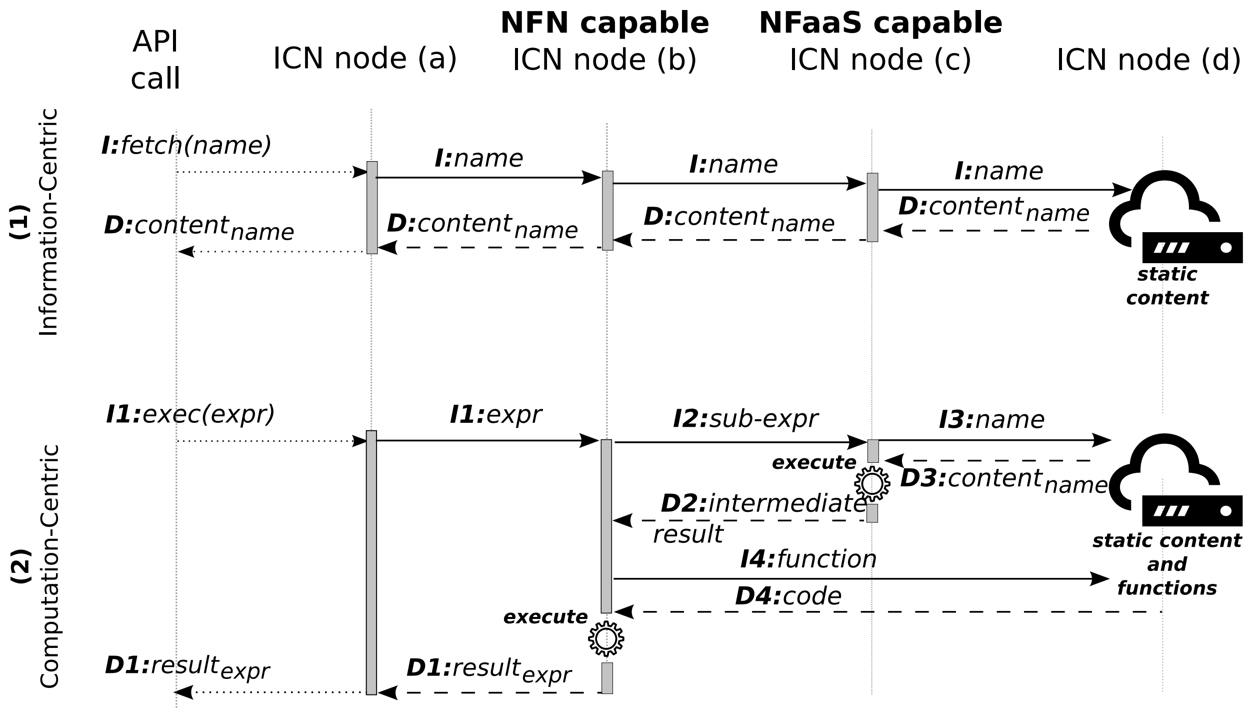

Fig. 6: Differences between static information-centric networking (top part of Fig.) and named function execution (bottom part of Fig.). While in ICNs content is fetched from the network, in computation-centric architectures the network is able to execute computations.

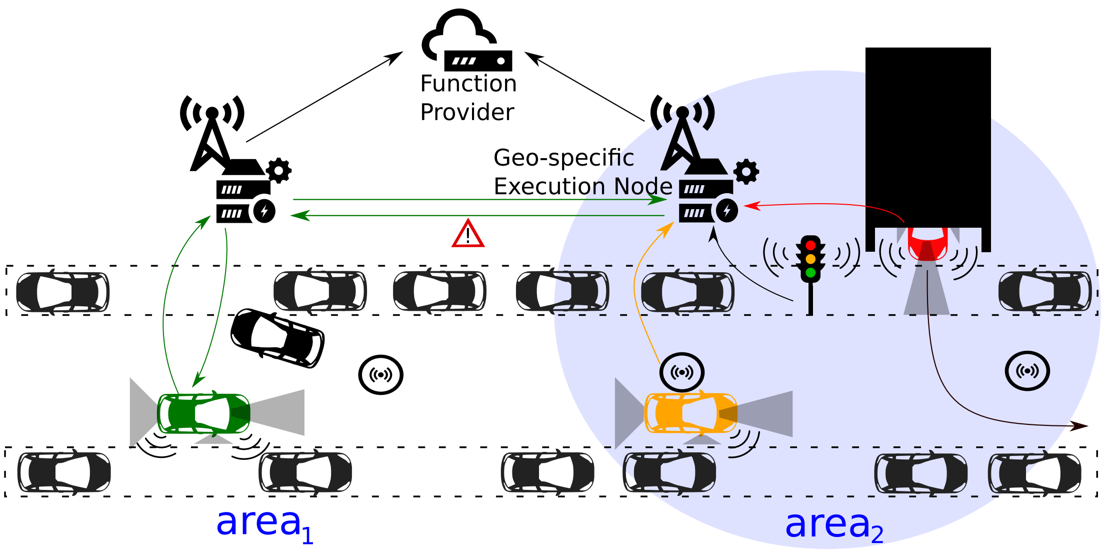

Fig. 7: Exemplary scenario of the localized sensing use case.

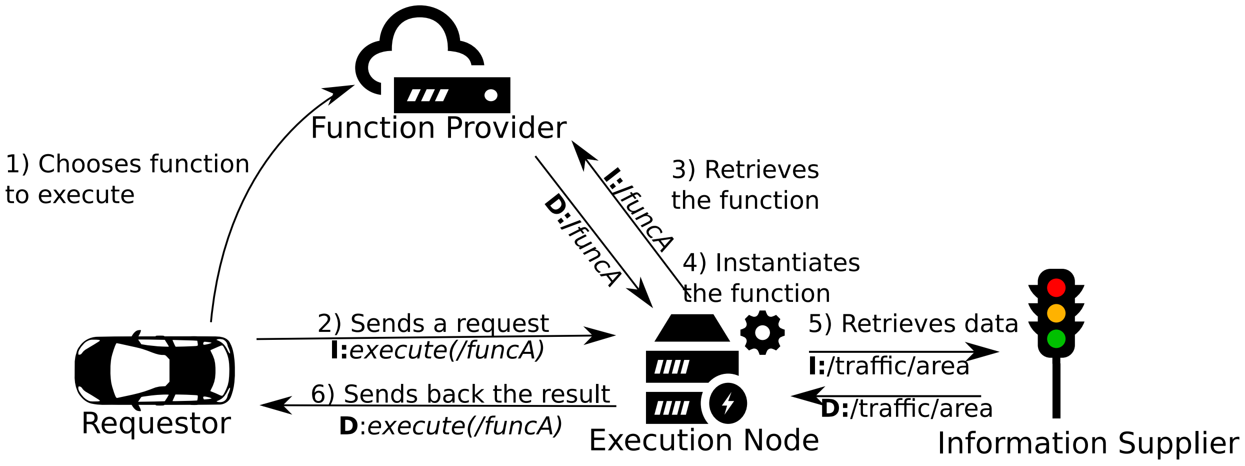

Fig. 8: System interactions.

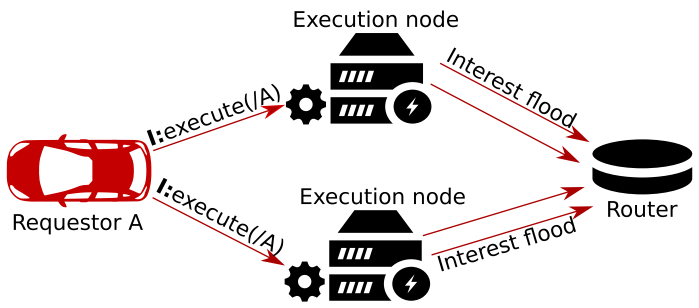

Fig. 9: Reflection attack.

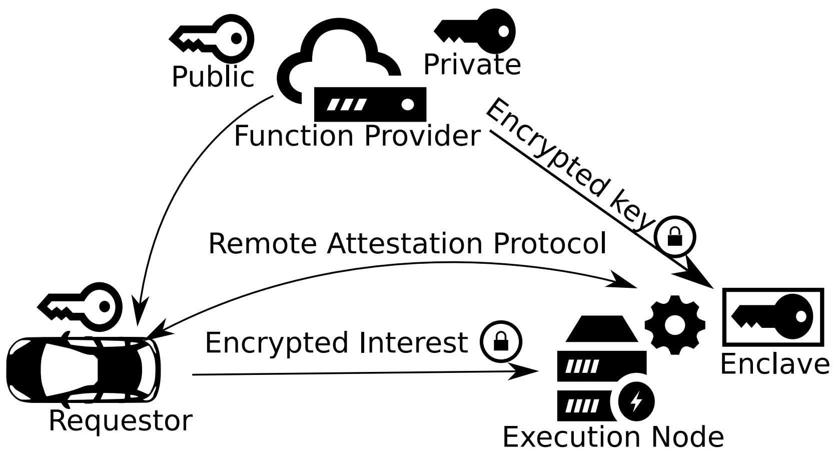

Fig. 10: Using Intel SGX to verify the function and encrypt Interests using private/public key scheme.