Simulation based validation of a Smart Energy Use Case with Homomorphic Encryption

Johannes Kölsch, Christopher Heinz, Axel Ratzke, Christoph Grimm
Chair "Design of Cyber-physical Systems"
TU Kaiserslautern
Kaiserslautern, Germany
koelsch|heinz|ratzke|grimm@cs.uni-kl.de

Gomathi Nandagopal
Vel Tech Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology
Chennai, India
gomathinandhagopal@gmail.com

***Abstract*—IoT systems consist of HW/SW systems (e.g. sensors) that are embedded in a physical world, networked and that interact with complex software platforms. The validation of such systems is a challenge and currently mostly done by prototypes. This paper gives an overview of an approach for the simulation- and emulation-based validation for large and complex IoT systems. The validation is supported by a simulation framework that also permits interaction with online-software, e.g. an IoT platform (emulation). The framework is demonstrated by a comprehensive case study. The example consists of the complete IoT "Smart Energy" use case with focus on data privacy by homomorphic encryption.**

***Index Terms*—Simulation, IoT, Homomorphic encryption, Smart grid, Validation**

I. Introduction

Nowadays, the number of applications in which IoT networks are deployed grows rapidly. These infrastructures operate in different domains such as smart cities, energy, eHealth, and transportation and behave like isolated islands in the global IoT ecosystem [1]. Their secure interconnection is a big challenge that still needs to be resolved. One promising approach towards interoperability of IoT networks across different domains is proposed by the VICINITY project. The project is supported by European Union Horizon 2020 program with the duration of 4 years (Jan. 2016 - Dec. 2019) and the consortium of 15 partners from 7 different countries. In this paper we present a simulation framework to validate the security of an smart energy use case, which is enabled by using homomorphic encryption. The performance of the overall IoT ecosystem is evaluated with and without using the additional homomorphic encryption layer. The real micro-service for the encryption is integrated into the simulation as hardware in the loop with the approach presented in [2].

A. The VICINITY project

The goal of the VICINITY project is to develop a platform that connects isolated IoT infrastructures into one global ecosystem called **virtual neighborhood** where users can select to which other systems their smart objects should be connected in a peer-to-peer network. The platform automatically supports interoperability from technical up to semantic level. The semantic interoperability of smart objects coming from different operators and using different standards is enabled with the semantic model integrated into the VICINITY platform [3].

The VICINITY architecture offering interoperability "as a service" is shown in Figure 1. It is based on a decentralized, bottom-up and cross-domain approach that resembles a social network, where users can configure their setups, integrate standards according to the services they want to use and fully control their desired level of privacy in a P2P (peer-to-peer) network.

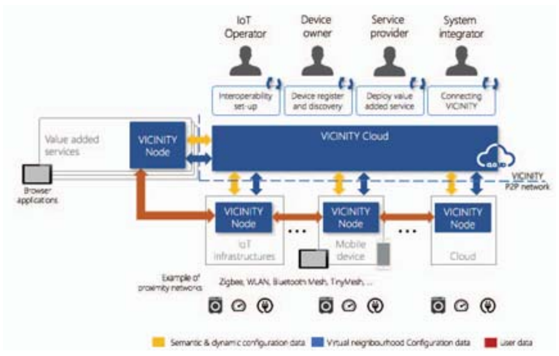

Fig. 1. High-level VICINITY architecture [4]

At end of the project VICINITY's approach will be demonstrated on four pilot sites coming from the following different domains: energy, building automation, health, and transport. VICINITY's potential to create new, cross-domain services will be demonstrated by value added services, such as micro-trading of demand-side management capabilities, AI-driven optimization of smart urban districts and business intelligence over IoT.

The focus of the work presented in this paper is to build a virtual environment for simulation and validation of IoT infrastructures and their use cases in real life scenarios before their real deployment. Here, a smart energy use case at one of the VICINITY pilot sites, located in Tromsø, Norway, will be used for demonstration.

As a part of the VICINITY project, this work has been supported by EU (European Union) program Horizon 2020 under grant agreement number 688467.

B. Homomorphic encryption

VICINITY's architecture offers privacy built-in by design, as only metadata on the connected devices is stored in a central cloud. Sensitive information like e.g. sensor readings are transmitted peer-to-peer only from the data producer to its intended consumer. Additionally, this needs to be approved by the data owner in advance and on an individual basis. Still, once the data owner has given his consent, his data is given out and will be available to potentially malicious third parties, which may seem trustworthy at first glance. As *Value-Added Services (VAS)* are the key to making the whole IoT "smart", a user may be tempted to share his data with such a service, even though he would rather not give out his information. An even better approach would be, to never give away any sensitive data (in clear text), yet still allowing the VAS to work. What may sound contradicting is exactly what is possible with the use of *homomorphic encryption (HE)* Schemes. Homomorphic encryption enables certain calculations, or in the case of fully homomorphic encryption, any arbitrary function to be executed on encrypted ciphertexts without the need to decrypt this data first! Partially homomorphic encryption has been subject to research for quite some time, only enabling a limited number of operations to be executed. With the introduction of a fully homomorphic encryption Scheme by Craig Gentry in 2009 [5] [6], any arbitrary computation is now possible on ciphertexts with no need to decrypt and giving out any cleartext information at all. However, fully homomorphic encryption is more expensive in terms of computational demand, so people need to decide on a trade-off between performance and versatility. As we will demonstrate in Chapter V-D, using our Framework enables us to evaluate different options in a realistic but controlled environment.

II. State of the Art

The size of the IoT market today is growing at enormous speed and will continue to do so. The number of connected devices has already exceeded the worlds human population [7]. In such complex IoT networks simulation plays an important role for the early-phase validation before their deployments in the real-life world.

[8] proposes the agent-driven *Smart Shire* (S^3) simulator that supports large-scale simulations with different communication mechanisms such as TCP/IP, MPI and shared memory. In [9] the authors extend *Smart Shire* towards IoT for the multi-level cross-domain simulation where the agent-driven S^3 simulator [8] is used for modeling objects and services at the higher level and the discrete-event OMNeT++ simulator[1] for modeling communication between objects at the lower level. After a number of experiments, they came to the conclusion that the simulation performance degrades as the number of entities simulated at the lower level with OMNeT++ increases.

To deal with this problem, [10] proposes an approach based on parallel and distributed simulation (PADS). The approach in [10] is based on the use of a hybrid simulator at the lower level where OMNeT++ is combined with the Matlab/Simulink-based simulator ADVISOR and works with it in succession. However, this approach does not provide solutions for general problems of the PADS simulation, such as lack of interoperability among simulators and lack of approaches for the automatic deployment and management of simulators on the distributed infrastructure.

Another hybrid simulation approach was proposed by [11] and [12]. [11] replaces the agent-based simulator S^3 with the Agent-based COoperating Smart Object (ACOSO) simulator for modeling smart objects in IoT networks. [12] presents the general-purpose hybrid simulation platform that supports simulation of interconnected IoT devices characterizing them by mobility, communication, and energy models.

III. Simulation of IoT networks

As powerful as it is, the discrete-event network simulation framework Omnet++ has it's fair share of problems, when tasked to simulate the entirety of a smart city. One of these is the performance loss when simulating the work of and communication between the number of models that are necessary for the simulation of an entire city. However, with the possibility to simulate such a smart city, not only single applications but the entirety of the complex interactions between them could be simulated and used to further facilitate concepts of smart cities and the Internet of Things.

Therefore this work aims to integrate the powerful Omnet++ framework for its capabilities of simulating network traffic and more important as a base for INET to simulate the Internet, with a lightweight custom Simulation framework, to reduce performance problems of Omnet++. To achieve this goal, the possibilities of the hierarchical modeling of DEVS models are used, to define a "time hierarchy" within the simulation. The idea here is, to dynamically switch between models that simplify significant portions of the simulated city (e.g. a quarter or district of the city) with fairly rough-grained time resolution, and coupled networks of models, that model smaller parts of the aforementioned models with increasingly finer grained time resolution. This allows us to reduce unimportant parts of a simulated city to resource saving rougher abstractions and dynamically observe important details, where they are of interest.

This paper brings the following novelties over state of the art: The proposed simulation framework supports multi-level simulation using only one simulation technique based on discrete-event simulation. The framework allows dynamic switching between models at different levels of abstraction that simplify significant portions of a simulated IoT network with fairly rough-grained time resolution. This further allows us to dynamically observe details that are relevant and filter ones that are not of interest for a particular simulation scenario.

A. Concept

As stated in chapter I, the aim of this paper is to create a general purpose DEVS simulator with hybrid simulation

[1]https://www.omnetpp.org

capabilities. for large scale IoT and Smart City simulations within the *VICINITY* use cases. Based on the requirements for general IoT simulations as well as requirements unique to the *VICINITY* project which are presented in section III-B, an approach to the problem is devised in the following sections.

B. Requirements

From the various works on the subject which were presented in section II, the most prevailing requirements for a simulator of the IoT have been gathered. Additionally, some requirements have arisen of the involvement of the Chair Design of Cyber-physical Systems with the VICINITY project. Those requirements will be examined in the following:

- possibility to simulate thousands of interconnected devices [8]: Depending on the scenario, it can be necessary to simulate thousands of entities that partake in the IoT. While for example in the simulation of a smart home, there are only a relatively small few devices that need to be simulated, during the simulation of a new area wide service, the number of simulated entities has to rise rather quickly, to provide useful data.
- ability to run in (almost) real-time for proactive approaches: While high detailed simulation runs can provide insight into the fine-grained processes within the interplay of different devices and technologies, with the sheer size of Internet of Things scenarios in the context of a smart city, these simulation runs tend to be too slow in order to enable proactive approaches. The framework under development should provide some techniques to enable these approaches.
- hardware in the loop capabilities: In order to analyze the behaviour of prototypes, it would be most beneficial if the developed simulation framework would provide hardware in the loop simulation capabilities or provide the interfaces to enable simple integration of already existing approaches.
- arising from the first two: high scalability of scenarios: Somehow connected to the first two requirements is the high scalability of scenarios. The simulation framework's real-time capabilities should not be lost, when thousands of entities that communicate with each other need to be simulated on a large scale.
- possibility to employ parallel and distributed simulations: Based on the observations in [10], the developed framework should provide either out of the box parallel and distributed capabilities, or at least be built in a way, that facilitates later adaption of these principles to further support the aforementioned requirements.
- fast model development for fast employment in use cases: In order to be employed in the *VICINITY* project, the developed framework should offer possibilities that support rapid model development and possibly even the use of functional mock-up interfaces.
- possibility to unify multiple heterogeneous technologies on all levels of IoT: The simulator should support modelling the prevalent technologies of the respective domain and enable the interplay between them.
- possibility to integrate further domain specific simulators into the framework: This last requirement pretty much speaks for itself. If during the deployment of the developed simulation framework the necessity of integrating further domain specific simulators arises it should be doable with little effort.

IV. Approach

Discrete event systems have been topic of research for over 40 years now, so techniques for parallel and distributed simulation are well-known and there exist multiple extensions to the original specification for various problems and fields of application, such as PDEVS, DynDEVS (which enables dynamically changing connections within DEVS [13]) and many more. Also modelling techniques for discrete event simulations are well known. Due to their relationship to finite state automata easy to grasp for unexperienced developers. For that reason, this approach uses the *DEVS* specification as foundation.

The majority of the examined related work identified the scalability as crucial for large scale simulations. Thus, the main focus of this work is to introduce a simulation framework for large-scale IoT simulations together with a modelling technique to enable rapid modelling of large-scale use cases.

Rather than combining different domain specific simulators to a multi-level simulation that invokes different simulators at different points in time during the simulation, this approach aims at dynamically altering the advancement of time and the models' level of detail during the simulation. This means that in order to allow high scalability even in large scale scenarios, the proposed framework simulates areas that are of no interest for the user with far less detail and bigger steps in time advancement. Specifically, the framework uses multiple models with different degrees of detail and time advancement for the same simulated entity and changes between them during the simulation, depending on the interest of the modeller. A real-world comparison could be imagined as a magnifying glass, that shows its focal point in great detail, while the everything else stays the same. This enables simulations of large-scale scenarios like the introduction of a new service in the area of a whole smart city, for example, where at the same time one can study the relationships between simulated entities all over the area, as well as detailed processes in single simulated entities or hardware in the loop.

Similarly to the multi-level simulations which were discussed in chapter II, this approach uses multiple levels of detail to simulate the desired scenarios. But rather than implementing the different levels through different domain-specific simulators, here, the levels in the hierarchy of the simulation are defined by the degree of time advancement and detail in the used models. This means of course additional costs in terms of modelling, since different models with varying degree of detail for the same system have to be developed, but through the re-usability of the used models, this is a one time cost.

In practice, the use and substitution of models during simulation runs is achieved by a switch between a low detail *atomic* DEVS model and a whole *network* DEVS model, that models the atomic model in more detail with finer grained time advance steps, and thus creating a new level of simulation. Hence, a good balance between models of areas that are of special interest and models that provide only the necessary background should efficiently lower the amount of produced events by the simulation. Besides, when the coarse and finer grained models are placed inside partitions together, already well-used PADS techniques for discrete event simulators can be applied. Of course, such lower levels could be represented through domain specific simulators as in the approaches shown in section II. For that reason, in this framework OMNeT++ with its INET expansion is used to implement the more detailed lower level models of *smart things* and their interplay.

A. Level Hierarchy and Structure

1) Expanding DEVS: To define the proposed approach in terms of the *DEVS* specification, a new kind of model is introduced. For the rest of this paper it will be called **Hierarchical Atomic**. It is a combination of an *atomic* and a *network* DEVS model that additionally provides functions to transfer one model's state into another and to select, which one should be the currently active:

$$hatomic \equiv \langle atomic, network, \{transport\}, select\rangle, \quad (1)$$

where atomic and network are the contained *atomic* and *network* DEVS model respectively. {transport} is the set of transport functions, that transfer the models' state and select is the function that chooses the currently active model. More detailed, the new type of model - in the following called *Hierarchical Atomic* - is defined:

$$\begin{aligned} hatomic \equiv \langle & S_A, X_A, X_N, Y_A, Y_N, \\ & D, \{M_N\}, \{I_N\}, \{Z_N\}, \delta_{int}, \delta_{ext}, \delta_{con}, \\ & \lambda, ta, select, \{transport\}\rangle, \end{aligned} \quad (2)$$

where A and N denote if the component belongs to the atomic or the network part of the combined model respectively. The *transport* functions

$$transport_{A\to N} : S_A \to S \subset \cup_i S_i \in D, for S_i \in D_i \quad (3)$$

and

$$transport_{N\to A} : S \subset \cup_i S_i \in D \to S_A, for S_i \in D_i \quad (4)$$

denote the functions that are necessary to transport the states of the contained models between each other, when a new level in the simulation is opened respectively closed. Note that neither of the functions has to be an *isomorphism*, since the contained network should be of course more expressive than the contained atomic. As a consequence, only a subset of possible states of the network models' components is used.

In more detail, the different contained models define the boundaries of the levels of a multi-level simulation: The encapsulated *atomic* DEVS model is one the same level as the new defined model, while the encapsulated *network* DEVS model belongs to the simulation level below. This way, the *transport functions* mark the transition between levels. Emanating from the top-level model in the simulation, all networks that are the same amount of encapsulation away from this root belong to the same level. This should be taken into account when developing models for a given scenario: When the different network models that define a level differ too much in terms of their step size in time advance, the use of that level for performance gain can degrade. I.e. when one of the network models that belong to that level would advance in time in much smaller steps than the rest, it would impact their efficiency as well.

B. Model reuse and the model tree

The organization in *atomic* and *network* DEVS models and the *closure under coupling* characteristic of DEVS allow for easy (re-)arrangement of model building blocks to more advanced and complex models in a tree-like structure. This procedure is supported by the here proposed approach. As can be seen in figure 2, the here introduced model that contains parts of the level architecture can easily be treated just as plain *atomic* DEVS model that is part of a *network* model in all positions in the tree structure. Moreover, the *atomic* models that belong to the contained *network* model can be exchanged with the new model as well, thus forming the different levels of hierarchy within the level of detail and time advancement of the simulation.

The consequences of this also tree-like organization of the simulation's levels are of importance for the general approach to modelling with the proposed simulator and influence the possible parallelization of the sequential approach.

First, regarding the approach to modelling, a bottom-up procedure is recommended, where the deepest and most detailed level is designed in its whole and then partitioned into smaller leaves of the tree. As the performance gain of the dynamic exchange between models with varying level of detail is dependent on the specific scenario, it is possible that more efficient partitions of models along the tree are only eventually evaluated under simulation. The bottom-up construction here ensures that the main part of the scenario has to be developed only once and then merely has to be partitioned accordingly. Also, if the models that are to be exchanged are the kind of whole protocol-stacks or space-divided parts of the environment, they in turn can be heavily be reused. If the capabilities of exchanging models with more or less elaborated representations of the system under simulation are used efficiently, in this way, the environment of smart home scenarios could be simulated following a user of a service through his way through a smart city, exchanging models on the fly, as soon as they are needed. In the implementation of the proposed approach, it is of importance where in this tree the integrated domain specific simulators are placed. Obviously,

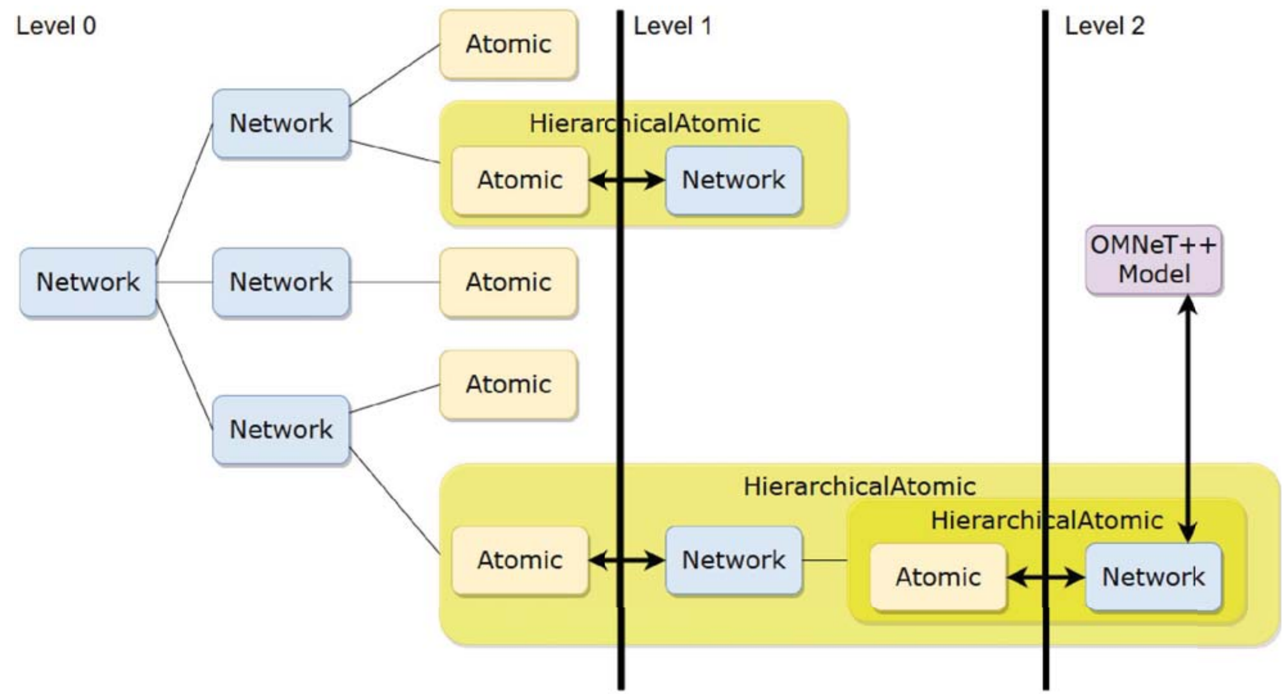

Fig. 2. The model tree and organization of hierarchy levels

a placement along the nodes of the tree as well as a simple hierarchy of one domain specific simulation kernel per level is conceivable.

Second, regarding possible partitioning techniques that could be used in a parallelized employment, it has to be taken into consideration how the different networks are placed. A high density of encapsulated networks in unluckily partitioned models could negatively affect the simulations performance. Therefore, modelling for a parallel scenario could be prolongated in comparison to other approaches.

During the implementation of the here proposed simulator, it was decided to place multiple instances of a simulator across the tree. The placement of simulators inside the introduced *hierarchical atomics* allowed them a more flexible implementation of their functionality. At the same time, it is ensured that the simulator that advances a specific model and the model itself remains separated. Furthermore as this approach is meant to exchange models on various levels of the tree at the same time, a placement of a single instance of a domain specific per level would prove impractical since several models along the whole simulation would be thwarted through a bottleneck.

C. Synchronization

Whenever an *atomic* model gets interchanged with a *network* model and vice versa, synchronization errors can occur. This is the result of already scheduled autonomous events, that now may become invalid. Looking at the mode of operation of the implemented *abstract simulator*, one can see that subsequently to an autonomous event the imminent model gets rescheduled with its time advance function. This little detail can be used to save complicated synchronization procedures: If the exchange of models can only occur during events, the subsequent rescheduling of the containing model will minimize the chance of falsely scheduled models.

However, depending on the concrete implementation of the FES, perhaps there are still internal events scheduled that are now too early or too late. For example, when the simulator switches from a detailed, slower advancing *network* model to the respective *atomic* one, it can happen that internal events of one of the network's faster advancing components are still inside the FES. The easiest solution to this is of course a future event schedule, that keeps at most one event of every *atomic* model inside itself. Through the encapsulation of the *network* model inside what looks like an *atomic* DEVS model and the immediate rescheduling after execution of the internal event, the model would be scheduled correctly. But in practise, this not always possible that easy. Different areas of application can demand for different scheduling strategies and the possibility exists that not all of them can respect the requirements of this approach. OMNeT++ already is designed with several possible scheduling classes. To maintain maximal flexibility of the proposed approach it is therefore necessary to answer the problem directly. Thus, the proposed new kind of model has to keep track of its own schedule on its own and decide on how to react to a falsely scheduled autonomous event independently. While models can keep track of their own model time through their time advance function and the fact that they get informed about the elapsed time since the last internal event (which supposedly followed a correct time advance of the model), when the occurrence of internal events loses reliability, this is not sufficient anymore. Hence the model needs the ability to acquire the global simulation time. This can be achieved by, for example allowing the single models to access the simulation environment or the simulation kernel like OMNeT++ does, or it can be accomplished by providing the global simulation time to the model with every function call to one of the state transition functions or the output function.

V. Case Study: Smart Energy Use Case

To illustrate the applicability and performance of the developed multi-level simulator, we modeled and simulated a smart energy use case with applied homomorphic encryption. This use case is an extension of the use case from [14]. This particular use case describes a smart energy scenario within a city. The city has a photovoltaic system and a windmill as power suppliers and a parking lot and a couple of houses as consumers. Electric vehicles can move inside the city and the parking lot. By using a smart parking service through a mobile app, users of the system can request to reserve their parking slot of choice within the participating parking facilities. The availability of the parking slots is then displayed through the mobile app as well as through the optical indicators located on the respective parking slots for random people, that do not participate in the smart parking service.

The described scenario has been modeled and simulated using the proposed approach at three distinct levels of abstraction: The first two higher levels have been implemented only using classes provided by the implemented core simulator. The third (lowest) level has been implemented with OMNeT++ 5.4.1[2] and its INET extension 4.0 [3].

A. The highest abstraction level - Level 0

The highest level of abstraction models abstract processes that are needed to provide basic information for the following lower levels of the simulation scenario. Furthermore the power generating entities are modeled on this high and abstract level. This is shown in Figure 3.

The *CarGenerator* atomic model acts as a source to the rest of the modules and provides the information needed to simulate users and random cars at the lower abstraction levels. This information is then passed to the *CarProcessor*.

The *CarProcessor* then determines if the received information is used to simulate a scenario with a random visitor of the parking facility or with a user of the smart parking mobile app. In the latter case, information about the desired parking slot is generated and used in an attempt to make a reservation via the model of the smart parking app. If this reservation fails, the information of the app user is treated like information about a random visitor of the facility and sent further to the *ParkingFacility*.

[2]https://www.omnetpp.org/21-articles/3752-omnet-5-4-1-released
[3]https://inet.omnetpp.org/2018-06-28-INET-4.0.0-released.html

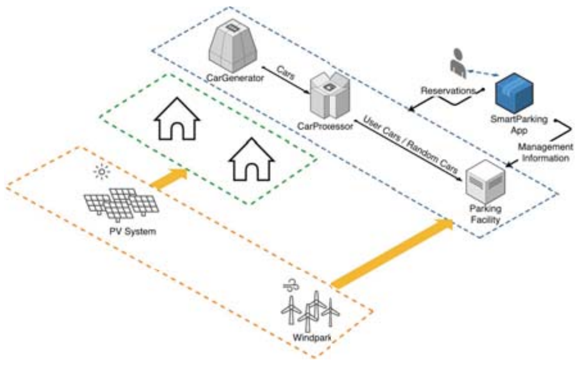

Fig. 3. Smart Energy use case: Level 0

Once entering the *ParkingFacility* model, the received information is used to model the abstract behavior of both random visitors and smart parking service users competing for available parking slots, parking and subsequently leaving the facility again.

Users of the app that succeed with a reservation will directly target their desired parking slots while random arrivals and users that failed to reserve their desired slot will choose the first free available parking slot. When arriving at the slot, it will be determined if it is still free or in the meantime has already been reserved by a user or taken by another random car that arrived first. If it has already been taken, they will head for the first free parking opportunity again. If they do not succeed in finding one, the car will leave the parking facility. If the parking process succeeds, the car will occupy the chosen parking slot for a while and then subsequently leave the parking facility again.

B. The middle abstraction level - Level 1

The following level of abstraction has been used to further detail the processes inside the *ParkingFacility*. It is shown in Figure 4. It divides the raw *ParkingFacility* into three different parking decks that internally mimic the behavior of the parking facility in Tromsø.

The information about car arrivals will be forwarded to the different parking decks in sequence; When entering the facility, the parking decks have to be traversed until the desired parking spot is reached. When a car leaves the *ParkingFacility*, the decks have to be traversed again in order to reach the exit. Additionally, the *ParkingDeckControl* is used to send information from and to the model of the app.

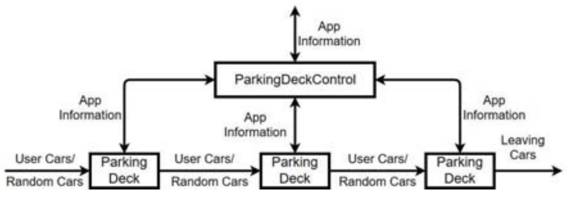

Fig. 4. Smart Energy use case: Level 1 - the Parking Facility

C. The lowest abstraction level - Level 2

The lowest level of the simulation scenario has been modeled with OMNeT++ 5.4.1 and INET 4.0. Here, the information produced by the higher levels described above is used to dynamically instantiate simulated entities and to represent the communication between sensors, actuators, and the app with the advanced capabilities of INET.

At this level, the parking facility has been modeled as an OMNeT++ *compound module* and the single parking decks as submodules of it. Although the different submodules can interact with each other across submodule boundaries, only the ones which are associated with the respective active higher level parts of the simulation will be actually active.

If the higher level parking deck model switches to the respective part of the OMNeT++ module at Level 2, cars will be created as mobile nodes with specific characteristics. The characteristics depend on the information generated at the levels above and the cars behavior is determined by the corresponding states at Level 1. Depending on if a car is now in the phase of searching for a parking slot or if it is already parked or even leaving the parking deck, the wireless node will be created and its goals will be set accordingly. One such parking deck modeled in OMNeT++ can be found in Figure 5.

Every Car has a battery, that is discharging as long as the car is moving inside the environment. When a car is parked inside the parking facility its accumulator is charged. When it is fully charged, it stops charging and the car is ready to drive away.

D. Integration of the homomorphic encryption micro-service

As briefly discussed in Section I-B, homomorphic encryption is particularly useful, when data is supposed to be processed by a third party, which cannot fully be trusted. One Example could be a value-added service by someone else. If a user is expecting some particular benefits or is even forced into using it, keeping his sensitive data private is a major concern.

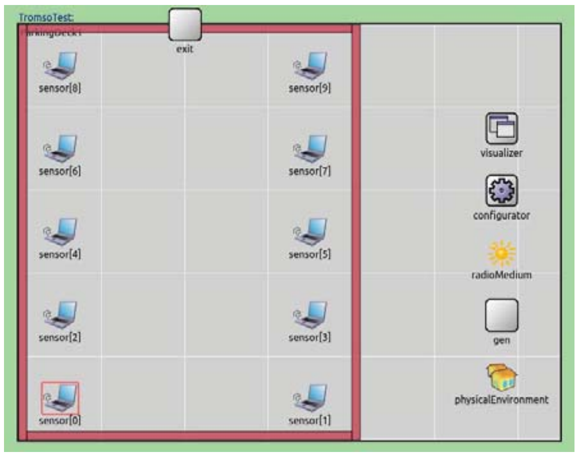

Fig. 5. Smart energy use case: Omnet++ Model of Parking Deck at Level 2

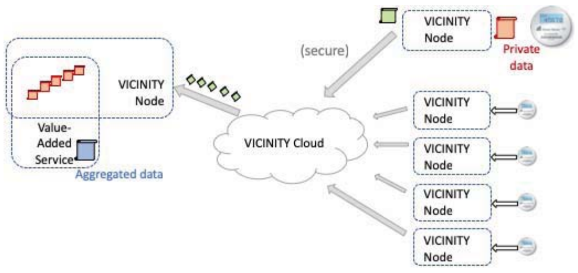

Fig. 6. Use Case integrated into the VICINITY network. Private data is available to value-added service in clear text.

Missing trust can potentially even be a showstopper for the whole Internet of Things.

Homomorphic encryption can help us with the above mentioned privacy concerns. However, this privacy comes at the price of increased computational effort for encryption, decryption and also functions evaluated on ciphertexts are more costly in terms of computational demand. Currently, the VICINITY project (see section I-A) is investigating the use of homomorphic encryption for these scenarios. To evaluate its practical feasibility, the computational overhead introduced by the use of homomorphic encryption needs to be analyzed. As there are numerous potential encryption schemes, which offer homomorphic properties, it is also important to evaluate them. To this end, we will simulate one of VICINITYs use cases both with and without the use of homomorphic encryption as to calculate the overhead this introduces under Lab conditions. Our Lab Setup utilizes Hardware-in-the-Loop simulations of the VICINITY infrastructure and can be adjusted way quicker and cheaper than an actual re-deployment on site. Figures 6 and 7 visualize the two simulated use cases:

As shown in Figure 6, Energy Consumption Data of the simulated cars is gathered and transmitted to the operators' value-added service. The operator is only interested in the overall energy consumption of its whole fleet and so it calculates the sum over all inputs first. As the energy consumption allows to draw some conclusion about the owners behaviour, we will consider this data as private data and also wish to keep this data private and not share if with any third-party.

As neither the operator, nor the users themselves have any particular interest in sharing their individual, private data, we

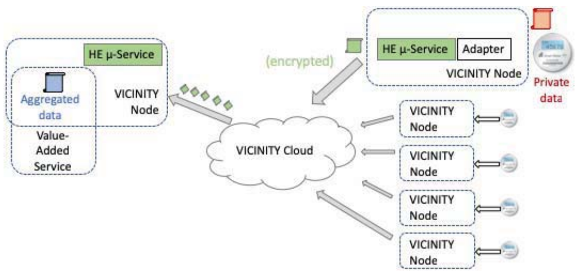

Fig. 7. Homomorphic encryption micro-service applied to Use Case

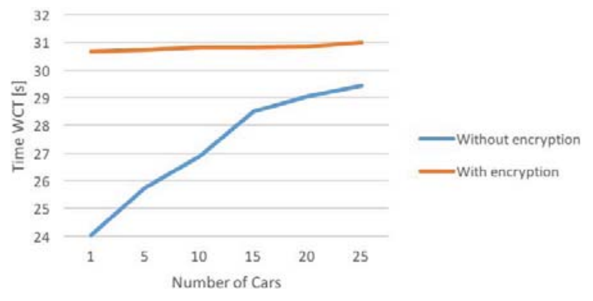

Fig. 8. Runtime of the simulation with and without homomorphic encryption

introduce the homomorphic encryption micro-service. It is integrated into the VICINITY dataflow as shown in Figure 7: Input Data (e.g. the Energy Consumption of each car) is encrypted using the Brakerski-Gentry-Vaikuntanathan (BGV) scheme, which is a fully homomorphic encryption scheme. The encrypted payload is then transmitted through the VICINITY peer-to-peer network and to the operators value-added service again. The homomorphic encryption micro-service performs the addition on the encrypted inputs, followed by a decentralized decryption on the aggregated data. This way, only the anonymized, aggregated data is visible and handed over to the value-added service in the first place.

VI. Conclusion

In order to evaluate the overhead introduced into the dataflow, due to the homomorphic encryption, we have evaluated two scenarios, with and without homomorphic encryption in order to compare their runtimes. As a starting point, the HElib Library [15] was used to implement the micro-service for homomorphic encryption. More encryption schemes will be evaluated and compared using the simulation framework presented in this paper.

A. Experimental Setup

The simulations were performed on a Mac Pro with a 3.5 GHz 6-Core Intel Xeon E5 CPU, 16 GB 1866 MHz DDR3 RAM, AMD FirePro D500 3072 MB GPU on MacOS High Sierra.

B. Results

As can be seen in figure 8, the runtime results of different simulation runs increase with an increasing number of cars which participated in the described VAS. This can be explained by the rising computational cost of simulating the necessary communications between the VAS participants. The increase of runtime of the simulation scenario which uses homomorphic encryption compared to the scenario without it, stems from two separate causes: First the pure computational cost on the service side for the encryption. And second the time necessary to send the huge (in comparison to unencrypted data) response ciphers into the simulated network.

Striking is the perceived difference in the growth rates of runtimes of both simulation scenarios: The growth of runtime of the scenario with homomorphic encryption is far less than without homomorphic encryption. This is probably due to the limits of the used realtime scheduler used for the hardware in the loop. With too many communications into a real network, the used scheduler has to drop some in- or out going communications in order to keep its realtime capabilities. In further experiments, different implementations of schedulers with realtime capabilities have to be evaluated.

References

[1] A. Mynzhasova, C. Radojicic, C. Heinz, J. Kölsch, C. Grimm, J. Rico, K. Dickerson, R. García-Castro, and V. Oravec, "Drivers, standards and platforms for the IoT: Towards a digital VICINITY," in *2017 Intelligent Systems Conference (IntelliSys)*, 9 2017, pp. 170–176.

[2] J. Kölsch, C. Heinz, S. Schumb, and C. Grimm, "Hardware-in-the-loop simulation for Internet of Things scenarios," in *2018 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 4 2018, pp. 1–6.

[3] R. García Castro (leading author), "Vicinity d2.2: Detailed specification of the semantic model," UPM, Tech. Rep., 2017. [Online]. Available: https://www.vicinity2020.eu/vicinity/sites/default/files/documents/vicinity_d2.2_vicinitysemanticmodel_v1.0.pdf

[4] V. Oravec (leading author), "D1.6: VICINITY Architectural Design," BVR, Tech. Rep., 2017. [Online]. Available: https://www.vicinity2020.eu/vicinity/content/d16-vicinityd16architecturaldesign10

[5] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, https://crypto.stanford.edu/craig, 2009.

[6] ——, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, Mar. 2010.

[7] D. Evans, "The Internet of Things How the Next Evolution of the Internet Is Changing Everything," Cisco Internet Business Solutions Group (IBSG), Tech. Rep., 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[8] S. Ferretti and G. D'Angelo, "Smart Shires: The Revenge of Countrysides," in *Proceedings of the IEEE Symposium on Computers and Communications*, ser. ISCC '16. Washington, DC, USA: IEEE Computer Society, 2016.

[9] G. D'Angelo, S. Ferretti, and V. Ghini, "Multi-level Simulation of Internet of Things on Smart Territories," *Simulation Modelling Practice and Theory (SIMPAT)*, vol. 73, pp. 3–21, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X16302507

[10] ——, "Distributed Hybrid Simulation of the Internet of Things and Smart Territories," *To appear in Concurrency and Computation: Practice and Experience*, vol. 30, no. 9, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4370

[11] G. Fortino, R. Gravina, W. Russo, and C. Savaglio, "Modeling and Simulating Internet-of-Things Systems: A Hybrid Agent-Oriented Approach," *Computing in Science Engineering*, vol. 19, no. 5, pp. 68–76, 2017.

[12] G. Brambilla, M. Picone, S. Cirani, M. Amoretti, and F. Zanichelli, "A Simulation Platform for Large-scale Internet of Things Scenarios in Urban Environments," in *Proceedings of the First International Conference on IoT in Urban Space*, ser. URB-IOT '14. Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 50–55. [Online]. Available: http://dx.doi.org/10.4108/icst.urb-iot.2014.257268

[13] G. A. Wainer, *Discrete-event modeling and simulation: a practitioner's approach*. CRC press, 2009.

[14] J. Kölsch, A. Ratzke, and C. Grimm, "Co-Simulating the Internet of Things in a Smart Grid use case scenario," in *2019 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 4 2019.

[15] S. Halevi and V. Shoup, "Helib - homomorphic-encryption library." [Online]. Available: https://github.com/shaih/HElib