Distributed MILS Architectural Approach for Secure Smart Grids

Denis Bytschkow(✉), Jean Quilbeuf, Georgeta Igna, and Harald Ruess

fortiss GmbH, An-Institut Technische Universität München,
Guerickestr. 25, 80805 München, Germany
{bytschkow,quilbeuf,igna,ruess}@fortiss.org
http://www.fortiss.org

Abstract. Successful decentralized and prosumer-based smart grids need to be at least as dependable and secure as the prevailing one-way, generation-transmission-distribution-consumer power grids. With this motivation in mind, we propose a two-phase model-based design methodology for secure architectural design and secure deployment of such a security architecture on a distributed separation kernel. In particular, we are modeling essential parts of a smart micro grid with several interacting prosumers, and demonstrate exemplary security/privacy requirements of this smart grid. The security policy architecture of this smart grid is deployed on a secure distributed platform, relying on a combination of separation kernels and deterministic network, as developed in the Distributed MILS project.

Keywords: Smart grid security · Distributed MILS · Separation kernel · Formal verification · Security policy architecture · Configuration compiler

1 Introduction

The electricity industry is in the middle of a paradigmatic shift towards smart power grids in order to meet the emerging needs of a highly reliable, efficient and sustainable society. In particular, the management of renewable, decentralized energy sources, higher volatility of power production, the consumer's active behavior coupled with sustainability objectives, and efforts towards new market designs drive the on-going transformation of the traditional power grid.

The control of smart grids requires significant change towards decentralized energy management systems (EMS) with a tight coupling of energy control with

This work has been carried out as part of the D-MILS project (www.d-mils.org) which is funded by the European Commission under the 7th Framework Programme for Information and Communications Technology. The smart grid case study has been supported by Siemens, the EIT ICT Labs, and the Bavarian Ministry of Economics.

DOI: 10.1007/978-3-319-10329-7_2

new monitoring, processing, optimizing, and controlling devices based on real-time information and communication technology (ICT). Moreover, electricity consumers are evolving into economically motivated *prosumers* that not only consume, but can also produce and store electricity. Prosumers can become smart energy ecosystems if they are equipped with market access and suitable ICT that allow them to achieve their own objectives.

Prosumer-based smart grid systems, however, are subject to a multitude of new types of attacks and threats [5]. Smart meter data may be manipulated to consume electricity without or with reduced payment. Manipulation of actuator components may damage the physical prosumer system, or even allow a burglar to break in. Finally, attacks to the communication infrastructure may lead to leaks of private and financial data.

A number of security guidelines and requirement specifications for smart grid infrastructure have recently been developed, including the ENISA Smart Grid Security report [13], the ENISA guidelines for security measures [12], and the NIST report NIST-IR-7628 [17]. The ISO/IEC TR 27019 report provides security guidelines based on ISO/IEC 27002 for process control systems specific to the energy industry. It has been pointed out, however, that engineering dependable and secure energy systems based on these guidelines is rather costly and time-consuming due to the need for extensive reviewing and testing of critical smart grid protocols, their low-level implementations, and the inherent dynamics of the environment of smart grid components [22].

In this paper we are proposing a novel approach for engineering secure smart grids based on model-based design methodology for embedded systems. In particular, we are proposing the model-based MILS approach for designing and implementing dependable and secure smart grids. MILS [1,21] was originally an acronym for Multiple Independent Levels of Security and is popularly characterized as the use of a separation kernels and information flow mechanisms to support both untrusted and trusted applications from diverse security domains on one computational system. Key concepts of the MILS architectural approach include separation, component integration, policy architecture, and physical resource sharing [3]. Separation concepts are consistent with approaches like intransitive noninterference [19] and partitioning in integrated-modular avionics [20]. Nowadays MILS is a standard platform in the US for deploying ultra-dependable mixed-criticality systems, but it is still virtually unknown in Europe. The Distributed MILS (D-MILS) project [9] extends the classical work of MILS to distributed embedded systems by realizing a distributed separation kernel by means of deterministic and predictable network communication (e.g. via time-triggered Ethernet). In this way D-MILS provides the capability to use one policy architecture that seamlessly spans across multiple MILS nodes—as required for smart grids. Moreover time and space separation of the D-MILS platform might be used to minimize the effect of faults or attacks to certain regions only, thereby avoiding rolling blackouts. Given a high-level model of the system, security properties in D-MILS are established in two phases

1. The policy architecture defined by the high-level model is established and enforced by the configured D-MILS platform.
2. Security properties are checked on a high-level model assuming that the policy architecture is enforced.

The policy architecture indicates how information is allowed to flow amongst the different components of the system. Consequently the construction of security assurance cases is separated into (1) establishing security of the high-level model and (2) enforcing the policy architecture defined by the high-level model through configuration of the distributed platform. The D-MILS platform[1] is specified to have provable domain isolation and information flow controls, thereby increasing assurance of the absence of hidden channels and unwanted information flow. We are using the D-MILS architectural approach for demonstrating various privacy requirements for a prosumer-based smart grid.

In this paper we are applying and demonstrating key concepts of the D-MILS methodology for establishing representative security properties of a smart micro grid. This case study is based on the smart grid demonstrator at fortiss [15], which has been built-up in the context of the European network of smart grid living labs of the EIT ICT Labs. This micro grid is mainly used for experimenting with distributed controls for (self-) stabilization of prosumer-based smart grids.

This paper is organised as follows. In Sect. 2, we introduce the smart grid example that we will use to illustrate the D-MILS approach. In Sect. 3, we present the D-MILS platform. In Sect. 4 we show how our case study is formalized using AF3. Finally, in Sect. 5 we show how (1) the policy architecture is enforced by configuring the platform and (2) how a simple security property is verified assuming that the underlying policy architecture is enforced by the platform.

2 Case Study

The smart micro grid entails a large variety of security requirements. Figure 1 shows an overview of the system. The system consists of a finite number of prosumers communicating with a Micro Grid EMS. Communication channels are represented by arrows in the figure.

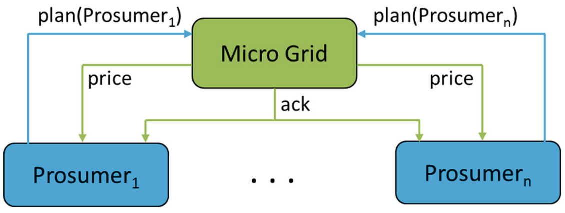

Fig. 1. High-level description of the smart micro grid system

[1] The D-MILS platform consists of a LynxSecure hypervisor provided by LynuxWorks and a TTEthernet solution provided by TTTech.

Each prosumer communicates with the Micro Grid through well-defined interfaces. Initially, the Micro Grid sends the energy price (through channel `price`) for the next day. As a response, each prosumer sends its planned consumption and production (through channel `plan`) for the next day to the Micro Grid. The Micro Grid validates the plans received by checking that the overall energy flow through the grid implied by these plans does not exceed the power line capacity. The Micro Grid sends back an acknowledgment message `ack` that contains the value 0 if the plans are within the power line capacity, otherwise it contains the amount of energy that exceeds the line capacity. Based on this acknowledgment message `ack`, each prosumer is responsible to update its own plan and send it back to the Micro Grid. The negotiation terminates when `ack`=0, meaning that the energy flow on the grid does not exceed the line capacity.

The security requirements that the Micro Grid system should fulfill are described in detail in [10]. In this paper, however, we focus on one of the most relevant requirements, which is related to privacy

RQ: No prosumer knows the consumption of another prosumer.

Using the D-MILS approach, the requirement RQ is ensured by two simpler requirements:

RQ1: No prosumer is able to bypass the defined communication channels to find out the consumption plan of any other prosumer.
RQ2: No prosumer is able to deduce the consumption plan of any other prosumer with the received information.

The first requirement refers to the low-level implementation of the system. This requirement is enforced through separation capabilities of the platform and its configuration by a configuration compiler (Sect. 5.1). The configuration files are built from a formal model of the system. A configured D-MILS platform guarantees the absence of unintended communication channels that are not in the formal model. The second requirement is ensured by checking that the formal model satisfies a security hyperproperty as described in Sect. 5.2.

3 D-MILS Platform

The D-MILS platform consists of a set of MILS nodes that communicate over a deterministic network. A MILS node implements a minimum separation kernel, which is a low-level hypervisor that controls the information exchange between the applications and virtualizes hardware resources. Its minimalistic design allows the separation kernel to be exhaustively tested and evaluated to meet low-level properties, such as being not by-passable and tamper-proof. The networking system, e.g. time-triggered Ethernet, guarantees message delivery and separation.

The separation kernel of each node as well as each switch of the network is statically configured. The configuration guarantees the absence of unintended information exchange in the system deployed on the distributed MILS platform. An example of a possible D-MILS platform configuration is shown in Fig. 2.

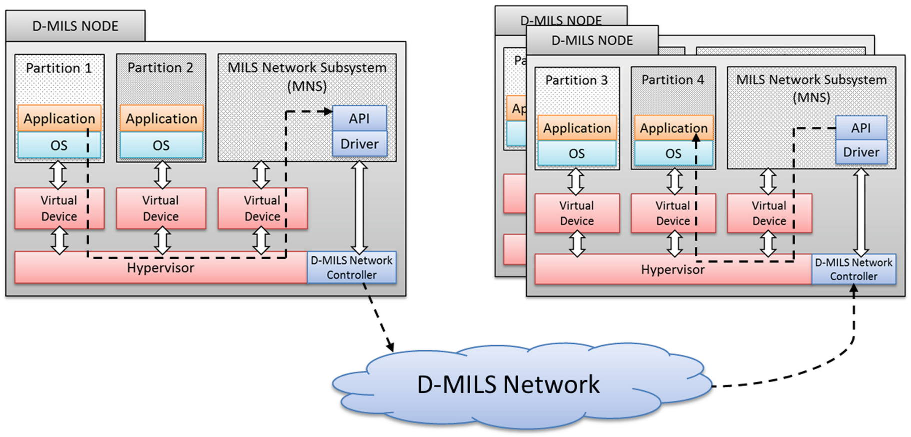

Fig. 2. D-MILS platform and possible information flows between different partitions

The platform consists of several MILS nodes and a Time-Triggered Ethernet as the networking system. The D-MILS configuration allocates a partition for each component of the system. Communications inside a node are handled by the local hypervisor, which also provides the resources as virtual devices for the local partitions. In each node, a dedicated partition called MILS Network Subsystem (MNS) handles the communications over the network. The D-MILS network ensures separation of (virtual) communication channels between the nodes through time partitioning. A virtual channel is depicted in Fig. 2 as a dashed line.

4 Formalization of the Information Flow

The system representation of Fig. 1 illustrates the intended communication channels. We formalize the system as a component-based model. We use AUTOFOCUS 3 (AF3), which is a modeling tool[2] based on the semantics of the FOCUS theory [4]. Each component has a well-defined interface explicitly represented as a set of ports. AF3 provides strong data encapsulation, i.e. each data variable resides within an atomic component and each atomic component has exclusive access to its variables. Therefore, data-sharing is only possible through explicit communication channels.

Components can be hierarchically decomposed into subcomponents in order to model complex systems in a comprehensible manner. Components are synchronously executed based on a discrete global time. The semantics of a component can be specified with state-automata, tables with input/output specification, or a stateless code specification. This semantics defines a *stream* [4] for each port, representing the successive values taken by that port at each time step.

[2] AF3 is an open source tool available at http://af3.fortiss.org.

Figure 3 shows our model. The model contains three prosumers, a micro grid and communication channels to the environment. Black dots represent output ports, whereas the white dots represent input ports. Arrows encode the communication channels. The communication between the components of the model define the policy architecture.

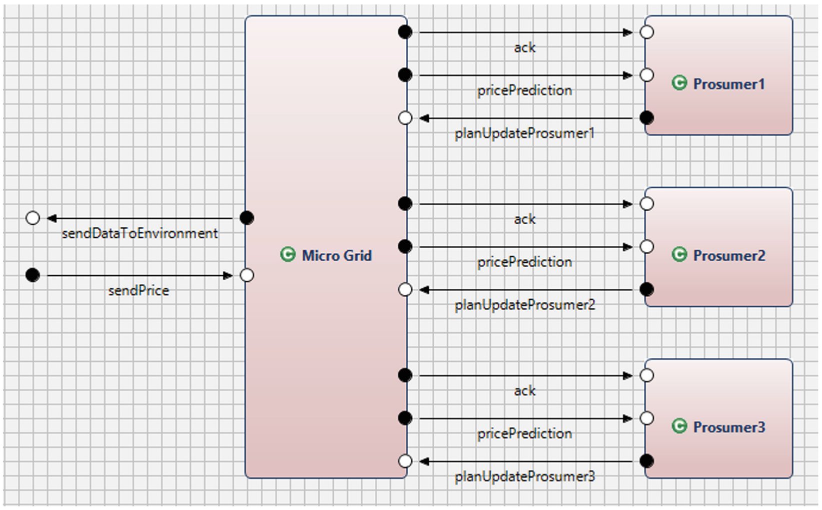

Fig. 3. The connections between the components define the policy architecture for the smart micro grid system.

The Micro Grid receives the energy price for the next day and transmits it to prosumers through the channels `pricePrediction`. The prosumers react to the price by generating their plan for the next day. Each prosumer `Prosumer` i sends its plan through the channel `planUpdateProsumer` i. A plan consists of a nonnegative production plan P_i, a non-positive consumption plan C_i and battery usage plan B_i indicating whether the battery will store energy (negative value) or provide energy (positive value). We summarize this information in a tuple (P_i, C_i, B_i). We furthermore assume that for each prosumer i, $P_i \in \{0, 1, 2\}$, $C_i \in \{-3, -2, -1, 0\}$ and $B_i \in \{-1, 0, 1\}$. By adding the three values, we get the global contribution of the prosumers to the grid. For the plans to be accepted, the balance of energy produced/consumed must be admissible for the system, which is specified by line capacity bounds L and H. More precisely, the set of plans is accepted if $L \leq \sum_{i=1}^{n}(P_i + C_i + B_i) \leq H$. In this case, the Micro Grid sends zero on the `ack` port. In the sequel, we define that $L = -7$ and $H = 4$. If the sum is below L or above H, the difference is sent back on the `ack` port to the prosumers. In this case, each prosumer updates its plan accordingly and sends it back to the Micro Grid. This negotiation process terminates when the plans are accepted.

The trace of an execution consists of successive values written on the ports at each step. The first steps of a trace (for ports `planUpdateProsumer`$\{1, 2, 3\}$ and `ack`) are depicted in Table 1. The `ack` port is represented once since it is the same for each prosumer. This trace represents a negotiation, where the plans initially consume too much energy (an excess of four energy units in the first round, and one unit in the second). Accordingly, Prosumers 1 and 2 reduce their consumption by discharging the battery instead of charging it.

Table 1. The beginning of a trace for the smart micro grid

Execution step	1	2	3	4	5	6	...
planUpdateProsumer1	$(0, -3, -1)$		$(0, -3, 0)$		$(0, -3, 1)$		...
planUpdateProsumer2	$(0, -3, -1)$		$(0, -3, 0)$		$(0, -3, 1)$		...
planUpdateProsumer3	$(0, -3, 0)$		$(0, -2, 0)$		$(0, -2, 0)$		...
ack		-4		-1		0	...

5 Ensuring Security Properties

As said in Sect. 2, the D-MILS approach separates our main requirement RQ into two simpler requirements. The requirement RQ1 is obtained by enforcing the policy architecture defined by the connections in the high-level model. A platform configuration automatically derived from the high-level model by a configuration compiler is used for this purpose, as described in Subsect. 5.1.

At the software component level, security policies are expressed through hyperproperties [8]. The assumption that communications match the policy architecture is supported by the platform configuration. Hyperproperties allow us to formalize security requirements such as RQ2, as shown in Subsect. 5.2.

5.1 Ensuring Security of the Platform

A D-MILS platform establishes and enforces an intransitive security policy, provided that it is configured properly. Such a security policy defines communication channels between components. A configured D-MILS platform ensures that the only possible communications are the ones defined in the security policy. In particular, it guarantees that there are no hidden channels.

A D-MILS platform consists of a set of nodes, connected through a network. Each node is a physical machine equipped with a separation kernel. The latter defines *partitions* that are completely isolated and host components. Two components running on two distinct partitions of the same node are able to communicate only if the configuration allows it. The communication between components deployed on distinct nodes relies on a network able to enforce separation of channels. In the D-MILS project, a time-triggered network is used for

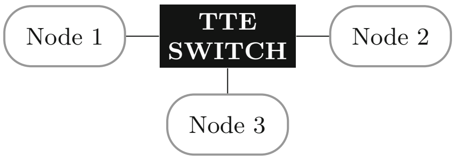

Fig. 4. A simple D-MILS platform

this purpose and separation is enforced through time partitioning. In the sequel, we assume a D-MILS platform as depicted in Fig. 4.

The configuration compiler is a tool that produces a configuration for each node and each switch of the platform. To this purpose, the configuration compiler is fed with the policy architecture, such as in Fig. 3, a model of the platform, such as in Fig. 4 and deployment information. For instance, one could assume that Prosumer 1 and Micro Grid components from Fig. 3 are deployed on nodes 1 and 2 from Fig. 4 respectively, and that Node 3 hosts both Prosumer 2 and Prosumer 3. Therefore, the configuration compiler generates configuration files such that Node 3 runs two separate partitions and that the communication channels between partitions are restricted to the channels of Fig. 3. The deployed system is depicted in Fig. 5. Each node contains a dedicated partition for hosting a Mils Network Subsystem (MNS) in charge of the communications over the time-triggered network.

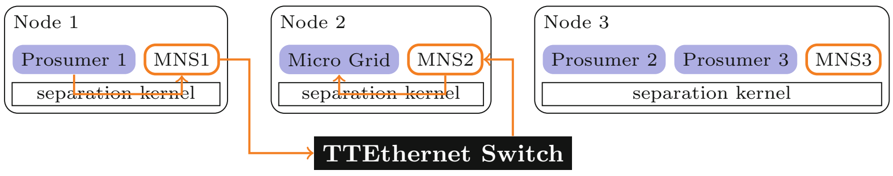

Fig. 5. A communication channel in a deployed D-MILS smart micro grid.

In order to state some correctness conditions for the MNS components, we derive an intermediate model between the high-level model and the corresponding deployed software. The intermediate model includes the components of the high-level model and the MNS components. Two different types of channels are defined:

- Channels between components deployed on the same node. These channels model communications handled by the separation kernel.
- Channels between MNS components. Each of these channels model a virtual link defined over the time-triggered network.

The channels between components hosted on different nodes are modified to be routed through the MNS components. Figure 6 shows how a channel *ab* between components deployed on distinct nodes is transformed. For each inter node-channel, a new virtual link is defined. In the figure, vl_{ab} is the virtual link defined to support the channel ab from the high-level model. For each channel of the high-level model either incoming to the node or outgoing from the node, the MNS component includes two corresponding ports. One port is connected to the component of the node involved in the channel, the other port to the virtual link associated to the channel. Each MNS transfers outgoing messages on the virtual link corresponding to the input port that received the message. Similarly, it transfers incoming messages on the output port corresponding to the virtual link that conveyed the message.

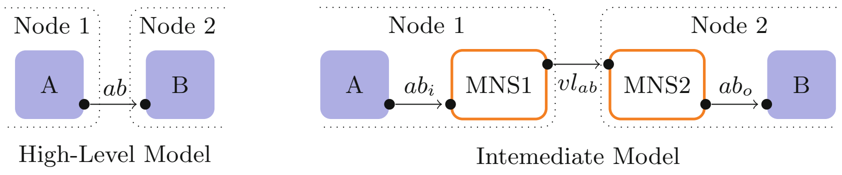

Fig. 6. Transformation of a channel between components deployed on distinct nodes.

Using this model, we say that the set of MNS components is correct if it implements the information flow described in the high-level model as in Fig. 1 for our example. On Fig. 6, the MNS are correct if in every execution, the sequence of values observed on the channel ab_i and the sequence of values observed on the channel ab_o are identical. A sufficient condition for correctness is stated through filter functions [6]. A filter function refines an existing information flow by specifying whether a given message is allowed according to the history of received and sent messages. In our case, a separate filter function is attached to each output port of the MNS component. The filter function allows an outgoing message only in the sequence of messages seen on the output port is a prefix of the sequence of message seen on the corresponding input port. The combination of these filters function correspond to our correctness criteria. Consider again the Fig. 6, the filter function on ab_o requires that it transmits the same sequence of messages as vl_{ab}. Similarly, the filter function on vl_{ab} requires that it transmits the same sequence of messages as ab_i. According to [18], checking that the filter functions are respected can be done locally for each MNS and does not depend on the implementation of other components.

Essentially, the platform guarantees that the information exchanged bet ween components follows the channels defined in the high-level model. Each component is only aware of the values sent to its input ports. For instance, each prosumer can only see the information sent by the Micro Grid component, and cannot directly communicate with another prosumer. However, a prosumer obtain get some information about other prosumers through the Micro Grid component.

5.2 Checking Security of the High-Level Model

In this section, we focus on requirement RQ2 which demands that no prosumer can deduce the consumption plan of any other prosumer. In order to formalize this requirement, we use the theory of knowledge [14]. In this theory, an agent observes only a part of the system and knows the set of all possible traces allowed by the system. If a particular property holds in all traces that are (1) possible and (2) consistent with the current observation, then the agent knows that this property holds in the current execution. In our case, the agent is a prosumer and its observation is restricted to its input and output ports. We want to ensure that the property "the consumption plan of prosumer i is X" cannot be deduced by any other prosumer.

For our analysis, we focus on the knowledge gained during one exchange of plans and acknowledgements. In our case, the set of traces is determined by the bounds imposed on the prosumer plans and the line capacity. Recall that each prosumer i may send any plan (P_i, C_i, B_i) such that $P_i \in \{0, 1, 2\}$, $C_i \in \{-3, -2, -1, 0\}$ and $B_i \in \{-1, 0, 1\}$. Then, the Micro Grid returns the value 0 on the port `ack` if $-7 \leq \sum_{i=1}^{n}(P_i + C_i + B_i) \leq 4$, otherwise, it returns the difference between the sum of the plans and the bound exceeded. We assume that each prosumer knows the set of possible traces, or, equivalently, the bounds for the prosumer plans and the line capacity. Such an assumption is supported by the fact that these bounds may be publicly available (line capacity), estimated (production from solar energy), or learned by observing multiple executions of the system.

Figure 7 presents a set of possible traces and the corresponding observations made by Prosumer 3. The bounds imply that only one trace is consistent with Observation 1, as it corresponds to the case where both Prosumer 1 and Prosumer 2 requested 4 units of energy ($P_1 = P_2 = 0$, $C_1 = C_2 = -3$ and $B_1 = B_2 = -1$), which is an extreme value. In this case, there is only one trace consistent with Observation 1, where $C_2 = -3$. Thus Prosumer 3 can deduce from Observation 1 that the consumption plan of Prosumer 2 is -3. However, several traces are consistent with Observation 2. In some traces $C_2 = -3$ whereas in some others $C_2 = -2$. Therefore, Prosumer 3 cannot deduce the exact consumption plan of Prosumer 2 from Observation 2. We conclude that this implementation is insecure because it allows one case (i.e. Observation 1) where a prosumer can infer the consumption of another prosumer.

The implementation is secure if there is no observation that allows a given prosumer to deduce the consumption of another one. This intuitive definition is formalized as follows. We denote by τ a trace (in our case a set of plans and the corresponding acknowledgement), and by T the set of all possible traces. Given a set of ports V, we denote by $\tau|_V$ the values taken by the ports in V when executing τ. In particular, prosumer i can observe the ports `planUpdateProsumer` i (abbreviated $\texttt{pUP}_i$) and `ack`. Hence, $\tau|_{\{\texttt{pUP}_i, \texttt{ack}\}}$ denotes the observation of prosumer i during the execution of τ. For ease of notation, we denote by $\tau|_{C_i}$ the value of the consumption in the plan sent by Prosumer i. We formalize the property "Prosumer i is not able to deduce the consumption plan of prosumer j" by

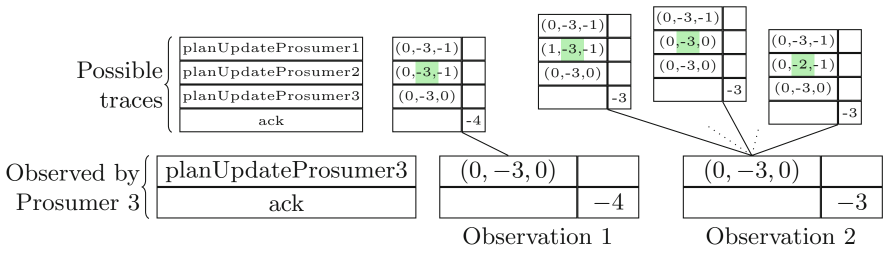

Fig. 7. Some possible traces and corresponding local observations. There is a single trace consistent with Observation 1, and several for Observation 2. The list of possible traces is not exhaustive.

$$\forall \tau \in T \ \ \exists \tau' \in T \ \ \tau|_{\{\mathtt{pUP}_i,\mathtt{ack}\}} = \tau'|_{\{\mathtt{pUP}_i,\mathtt{ack}\}} \wedge \tau|_{C_j} \neq \tau'|_{C_j} \tag{1}$$

The formula states that for any trace τ of the system, there exists at least another trace τ' where Prosumer i observes the same values as in τ, but such that the consumption of j differs between τ and τ'.

For this kind of properties, it is not sufficient to check that each trace individually complies with the property. Rather, the property depends on the exact set of traces T allowed by the system. Such properties are called hyperproperties and were introduced by Clarkson and Schneider [8]. Several formalisms exist to specify such security properties. Van der Meyden proposed several semantics for intransitive non-interference [16]. Intransitive non-interference is possibly reinforced through filter functions [6], that are used in [18] to specify a weaker security property for the same Micro Grid as discussed here. Balliu uses epistemic logic [2] in a more general case than ours, Clarkson et al. defines a temporal logics for hyperproperties [7].

In order to check the security property, we wrote a SMT-lib script. The script encodes the constraints for a trace where a prosumer i can deduce the consumption of another prosumer j. Formally, we search for a trace τ such that

$$\tau \in T \ \wedge \forall \tau' \in T \ \tau|_{\{\mathtt{pUP}_i,\mathtt{ack}\}} = \tau'|_{\{\mathtt{pUP}_i,\mathtt{ack}\}} \implies \tau|_{C_j} = \tau'|_{C_j} \tag{2}$$

If these constraints are unsatisfiable for each pair i, j such that $i \neq j$, no prosumer can deduce the consumption of another prosumer. Z3 [11] found a trace satisfying the constraint (2), which corresponds to Observation 1 from Fig. 7. Thus, the system is insecure.

In order to make the system secure, we add further constraints on prosumers plans. We ask that the global contribution of the prosumer remains within the bounds corresponding to the maximal consumption or production. Formally, for a plan (P_i, C_i, B_i), we require that $-3 \leq P_i + C_i + B_i \leq 2$. In this case, the extreme values for each plan can be reached in several ways, as in Observation 2 of Fig. 7, which hides the real value of the consumption. By adding these constraints on the prosumers plans, we modify the set of traces T allowed by the system. Consequently, Z3 outputs that the constraint (2) is unsatisfiable, meaning that the security property (1) is met.

6 Conclusion

We have outlined an application of the two-phase D-MILS architectural methodology for demonstrating security properties of the distributed implementation of a prosumer-based smart grid.

The low-level platform view of D-MILS is provided by a secure technology layer, which guarantees the sharing of resources using components, such as separation kernels and partitioning communications systems that deliver the required guarantee of separation. A correct configuration of those components provides the assurance that no unintended channels exist, and data leakage is prevented on the resource sharing level.

The high-level architectural view is represented by a formal model, which constitutes the system design and defines available channels for data exchange. The assurance argument that the high-level model indeed satisfies the given security goals is independent of low-level considerations of the platform implementation. This high-level architectural model is used as input for the configuration compiler, which produces configurations of the D-MILS components, which enforce the intended high-level information flow on the low-level technical platform; in particular, a correct configuration compiler does not introduce any hidden channels on the resource-sharing platform.

The separation into an architectural and a platform-dependent part structures and considerably simplifies the construction of assurance cases for security properties of the micro grid case study.

We have demonstrated that representative privacy properties of the micro grid case study can be encoded in terms of hyperproperties. We proposed a preliminary encoding of facts deduced from the execution traces which are visible at the input ports. These encodings allowed us to detect a case where privacy is broken and propose an alternative model that is more secure. The analysis can be extended by taking into account the history of actions (instead of one step of execution as currently). Ultimately, the privacy property should be stated in term of the quality of approximation that a prosumer can obtain from a given observation.

Altogether, our smart grid case study demonstrates that the D-MILS approach is suitable to reason about security requirements. Together with a secure low-level technical platform, which can be configured to allow desired information flow channels, and more importantly prevent undesired information flow channels, the D-MILS approach provides an environment for design, analysis, verification, and implementation of scalable, interoperable and affordable trustworthy smart grid architectures. However, significant progress along several lines is needed for reaching our ultimate goal of a complete and cost-effective solution for securing smart grids. In particular, the D-MILS methodology as outlined needs to be supported by (1) a high-level architectural language (e.g. AADL) for specifying the security policy architecture and a wide variety of security properties, (2) a suitable automated and suitable *verification framework*, (3) *assurance and verification methods* for compositional assurance, and (4) a *runtime monitoring plane* for testing, diagnosis, assessment, auditing and management of

D-MILS systems. Moreover, the current versions of D-MILS are static and do not support the evolution of smart grids in time, as configurations are determined a priori and cannot be dynamically changed during runtime. Both the D-MILS architectural design methodology and platform need to be extended to support dynamically changing information flow policies and platform configurations.

References

1. Alves-Foss, J., Harrison, W.S., Oman, P., Taylor, C.: The MILS architecture for high-assurance embedded systems. Int. J. Embed. Syst. **2**(3/4), 239–247 (2006)
2. Balliu, M.: A logic for information flow analysis of distributed programs. In: Riis Nielson, H., Gollmann, D. (eds.) NordSec 2013. LNCS, vol. 8208, pp. 84–99. Springer, Heidelberg (2013)
3. Boettcher, C., DeLong, R., Rushby, J., Sifre, W.: The MILS component integration approach to secure information sharing. In: IEEE/AIAA 27th Digital Avionics Systems Conference, 2008, DASC 2008, pp. 1.C.2-1–1.C.2-14. IEEE (2008)
4. Broy, M., Stølen, K.: Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement. Springer, Secaucus (2001)
5. Camek, A., Holzl, F., Bytschkow, D.: Providing security to a smart grid prosumer system based on a service oriented architecture in an office environment. In: Proceedings of Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES (2013)
6. Chong, S., van der Meyden, R.: Using architecture to reason about information security. In: Layered Assurance Workshop (2012)
7. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. In: Abadi, M., Kremer, S. (eds.) POST 2014 (ETAPS 2014). LNCS, vol. 8414, pp. 265–284. Springer, Heidelberg (2014)
8. Clarkson, M.R., Schneider, F.B.: Hyperproperties. J. Comput. Secur. **18**(6), 1157–1210 (2010)
9. D-MILS: Distributed MILS for dependable information and communication infrastructures. STREP, FP7. http://www.d-mils.org
10. D-MILS: Safety and security requirements for the fortiss Smart Micro Grid demonstrator (2013), d-MILS project deliverable
11. de Moura, L., Bjørner, N.S.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
12. ENISA: Appropriate security measures for smart grids - guidelines to assess the sophistication of security measures implementation. Study of the European Network and Information Security Agency (ENISA) (2012)
13. ENISA: Smart grid security - recommendations for Europe and member states. Study of the European Network and Information Security Agency (ENISA) (2012)
14. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press, Cambridge (1995)
15. Koss, D., Sellmayr, F., Bauereiß, S., Bytschkow, D., Gupta, P.K., Schätz, B.: Establishing a smart grid node architecture and demonstrator in an office environment using the SOA approach. In: SE4SG, ICSE, pp. 8–14. IEEE (2012)
16. van der Meyden, R.: What, indeed, is intransitive noninterference? In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 235–250. Springer, Heidelberg (2007)

17. NIST: NIST IR 7628: guidelines for smart grid cyber security (2011). http://csrc.nist.gov/publications/PubsNISTIRs.html
18. Quilbeuf, J., Igna, G., Bytschkow, D., Ruess, H.: Security policies for distributed systems. CoRR abs/1310.3723 (2013)
19. Rushby, J.: Noninterference, transitivity, and channel-control security policies. SRI International, Computer Science Laboratory (1992)
20. Rushby, J.: Partitioning in avionics architectures: requirements, mechanisms, and assurance. Technical report, DTIC Document (2000)
21. Vanfleet, W.M., et al.: MILS: architecture for high assurance embedded computing. Cross Talk **18**, 12–16 (2005)
22. Yardley, T., Berthier, R., Nicol, D., Sanders, W.: Smart grid protocol testing through cyber-physical testbeds. In: ISGT, 2013 IEEE PES, pp. 1–6 (2013)