METIS: a Two-Tier Intrusion Detection System for Advanced Metering Infrastructures[⋆]

Vincenzo Gulisano, Magnus Almgren, and Marina Papatriantafilou

Chalmers University of Technology, Gothenburg, Sweden
`vinmas,almgren,ptrianta@chalmers.se`

Abstract. In the shift from traditional to cyber-physical electric grids, motivated by the needs for improved energy efficiency, Advanced Metering Infrastructures have a key role. However, together with the enabled possibilities, they imply an increased threat surface on the systems. Challenging aspects such as scalable traffic analysis, timely detection of malicious activity and intuitive ways of specifying detection mechanisms for possible adversary goals are among the core problems in this domain.
Aiming at addressing the above, we present *METIS*, a *two-tier streaming-based intrusion detection framework*. *METIS* relies on probabilistic models for detection and is designed to detect challenging attacks in which adversaries aim at being unnoticed. Thanks to its two-tier architecture, it eases the modeling of possible adversary goals and allows for a fully distributed and parallel traffic analysis through the data streaming processing paradigm. At the same time, it allows for complementary intrusion detection systems to be integrated in the framework.
We demonstrate *METIS*' use and functionality through an *energy exfiltration* use-case, in which an adversary aims at stealing energy information from AMI users. Based on a prototype implementation using the Storm Stream Processing Engine and a very large dataset from a real-world AMI, we show that *METIS* is not only able to detect such attacks, but that it can also handle large volumes of data even when run on commodity hardware.

Key words: Advanced Metering Infrastructures, Intrusion Detection Systems, Data Streaming

1 Introduction

The shift from traditional to cyber-physical grids relies on the deployment of Advanced Metering Infrastructures (AMIs) in which communication-enabled meters share data with the utility's head-end and are remotely controlled. In this context, the strict coupling between threats' cyber and physical dimensions (that can possibly result in human losses or physical damage [4]) demands for appropriate defense mechanisms. As Stuxnet[7] taught us, malicious activity designed

[⋆] Some preliminary results of METIS that do not overlap with the contribution of this work have been presented in a poster paper at the fifth International Conference on Future Energy Systems (ACM e-Energy), 2014 to hide its malicious behavior can be carried out during years before being detected.

Despite the limited number of real attacks documented so far, a considerable number of possible attack vectors has been uncovered [17]. Specification-based Intrusion Detection Systems (IDSs) [1, 18], the main defense mechanism proposed so far for this domain, detect malicious activity by means of deviations from defined behavior. Such IDSs usually require a considerable amount of manual labor by a security expert in order to tune them to specific installations [1]. At the same time, they do not provide a comprehensive protection against all possible adversary goals. As an example, they might distinguish messages that comply with a given protocol from messages that do not, but might fail in distinguishing whether a message that does not violate the protocol is sent by an intact or a compromised device.

Challenges Kush et al. [14] claim traditional IDSs cannot be used effectively in these environments without major modifications and they mention nine challenges, four of which are taken into account in this paper: *scalability*, *adaptiveness*, *network topology* and *resource-constrained end devices*. As discussed in [1], AMIs consist of several independent networks whose overall traffic cannot be observed by a centralized IDS. Hence, the IDS should process data in a distributed fashion in order to embrace the different networks composing the AMI. Furthermore, the processing capacity of a centralized IDS would be rapidly exhausted by the big, fluctuating volume of data generated by AMIs' devices. To this end, the IDS should also process data in a parallel fashion in order to cope with the volumes of data and detect malicious activity timely. It should be noted that existing privacy regulations play an important role when it comes to the information accessed to spot malicious activity. As discussed in [19], fine-grain consumption readings reveal detailed information about household activities and could be used to blackmail public figures [8]. For this reason, while being interested in detecting malicious activity, the utility maintaining the AMI might not have access to underlying information owned by energy suppliers. Hence, the IDS should be able to detect malicious activity while relying on partial evidence (i.e., while accessing a limited set of traffic features). Finally, the IDS should avoid expensive per-site customization by providing an efficient way to specify how to detect malicious activities.

Contributions We present *METIS*[2], an Intrusion Detection framework that addresses these challenges by employing a two-tier architecture and the data streaming processing paradigm [23]. *METIS* has been designed giving particular attention to the detection of malicious activity carried out by adversaries that want to go unnoticed. The challenge in the detection of such malicious activity lies in that suspicious traffic proper of a given adversary goal can be caused by both legitimate and malicious factors. We provide the following contributions:

[2] Named after the mythology figure standing for good counsel, advice, planning, cunning, craftiness, and wisdom.

1. A two-tier architecture that provides a scalable traffic analysis that can be effective while (possibly) relying on a limited set of traffic features. Its two-tier architecture eases the system expert interaction (who can model the traffic features affected by an adversary goal by means of *Bayesian Networks*) and allows for complementary detection mechanisms such as specification-based and signature-based ones to be integrated in the framework.
2. A prototype implementation programmed using Storm [24], a state of the art Stream Processing Enginge used mainstream applications (such as twitter).
3. One of the first evaluations based on data extracted from a real-world AMI and focusing on *energy exfiltration* attacks in which the adversary aims at stealing energy consumption information from AMI users. The evaluation studies both the detection capabilities of the framework and its applicability while relying on commodity hardware. To the best of our knowledge, detection of such attacks has not been addressed before.

The paper is structured as follows. We introduce some preliminary concepts in Section 2. In Section 3 we overview the *METIS*' architecture while we discuss its implementation in Section 4. An example showing how the framework is applied to the energy exfiltration use-case is presented in Section 5. We present our evaluation in Section 6, survey related work in Section 7 and conclude in Section 8.

2 Preliminaries

2.1 Advanced Metering Infrastructure model

We consider a common AMI model, composed of two types of devices: *Smart Meters* (SMs), in charge of measuring energy consumption and exchanging event messages such power outage alarms or firmware updates, and *Meter Concentrator Units* (MCUs), in charge of collecting such information and forwarding it to the utility head-end. Different network topologies exist in real-world AMIs (e.g., point-to-point, hierarchical or mesh ones). In order to encompass all possible networks and represent AMIs that can evolve over time, we consider a generic network, in which SMs are not statically assigned to specific MCUs.

Among the messages that are exchanged by the AMI's devices, two are of particular interest with respect to the use-case that will be introduced in the following: Energy Consumption Request (ECReq) messages, sent by MCUs, and Energy Consumption Response (ECResp) messages, sent by SMs. Such messages are used to retrieve energy consumption and can be exchanged several times per day.

2.2 Intrusion Detection in Advanced Metering Infrastructures

AMIs are characterized by their slow evolution and limited heterogeneity. That is, they are composed by a limited set of device types and their evolution is dictated by small (and often planned) steps (e.g., deployment of a new meter, replacement of a broken meter, and so on). Given a time frame that ranges from days to months, such evolution is “slow” and thus enables for detection techniques, such as anomaly-based ones, building on machine learning mechanisms. Nevertheless, the same evolving nature demands for a continuous learning that evolves together with the AMI (thus addressing the *adaptiveness* and *network topology* challenges discussed in [14]).

As introduced in Section 1, distributed and parallel network traffic analysis should be employed in order to embrace the different networks that compose AMIs while coping with the large and fluctuating volume of data produced by their devices. The distinct deployment options for an IDS in this domain can be characterized in a spectrum. At one extreme, the analysis could be performed by the utility head-end system. In this case, the devices should be instructed to report their communication exchanges to the head-end (at least, the ones that are required to detect a given attack). On the other extreme, the computation could be performed by the AMI’s devices themselves, as investigated recently in [20]. This option would also be limited by the computational resources of the devices. Intermediate solutions could rely on a dedicated sensing infrastructure that runs the analysis together with the utility head-end system, as discussed in [10]. To our advantage, relying on the data streaming processing paradigm simplifies the deployment of an AMI defense framework to the requirement of providing a set of nodes (sensing devices or servers) that embraces the possible existing networks of the AMI. We refer the reader to [11] for a detailed discussion about how data streaming applications can be deployed at arbitrary number of nodes (thus addressing the *scalability* challenge discussed in [14]).

2.3 Adversary model

Several types of attacks can be launched against AMIs. On one hand, attacks such as Denial of Service (DoS) or Distributed Denial of Service (DDoS) are meant to be noticed (i.e., they impose a challenge in their mitigation rather than detection). On the other hand, more subtle attacks can be carried out by adversaries that want to go unnoticed. This second type of attacks (imposing a challenge in their detection) are the main target of *METIS*. Such adversaries could be interested in installing a malicious firmware that, while leaving the device’s communication unaffected, would allow them to use the AMI as a communication medium [10]. At the same time, a malicious firmware could also be installed to lower bills by reducing the consumption readings reported by the meters (causing an *energy theft* attack [16]).

Energy Exfiltration use-case In this scenario, the adversary aims at stealing energy consumption information from AMI users. As discussed in [19], fine-grained consumption readings collected over a sufficiently large period reveal detailed information about household activities and could be used to blackmail public figures [8]. Given our AMI model, such malicious activity can be carried out after successfully logging into an MCU or by deploying a (malicious) MCU replica and collecting energy consumption readings over a certain number of days. The subtle nature of this attack lies in that suspicious exchanges of ECReq and ECResp messages can be caused not only by the adversary, but also by legitimate factors (e.g., noisy communication between devices, unreachable devices, and so on).

2.4 Data Streaming

A stream is defined as an unbounded sequence of tuples $t_0, t_1, \ldots$ sharing the same schema composed by attributes $\langle A_1, \ldots, A_n \rangle$. Data streaming *continuous queries* are defined as graphs of operators. Nodes represent operators that consume and produce tuples, while edges specify how tuples flow among operators. Operators are distinguished into *stateless* (e.g., *Filter*, *Map*) or *stateful* (e.g., *Aggregate*, *EquiJoin*, *Join*), depending on whether they keep any evolving state while processing tuples. Due to the unbounded nature of streams, stateful operations are computed over *sliding windows* (simply windows in the remainder), defined by parameters *size* and *advance*. In this context, we focus on time-based windows. As an example, a window with size and advance equal to 20 and 5 time units, respectively, will cover periods $[0, 20)$, $[5, 25)$, $[10, 30)$ and so on.

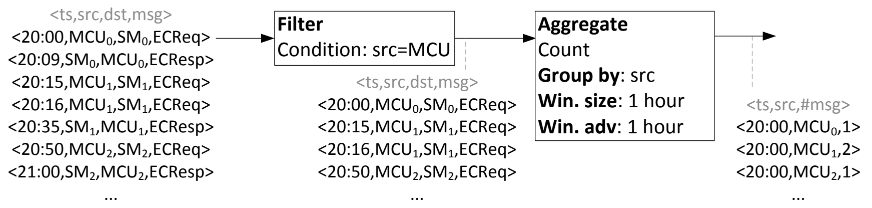


Fig. 1: Sample query that computes the number of messages forwarded by each MCU during the last hour. The figure includes the abstract schema and a set of sample tuples for each stream.

The generic schema of the streams generated by the AMI's devices is composed by attributes $\langle ts, src, dst, msg \rangle$, specifying the timestamp ts at which message msg is forwarded by source src to destination dst. In the remainder, we use the terms tuple and message interchangeably when referring to the devices' communication. Figure 1 presents a sample query that computes the number of messages forwarded by each MCU during the last hour for a given set of input tuples (also shown in the figure).

2.5 Bayesian Networks

Bayesian Networks (BNs) provide a probabilistic graphical model in which a set of random variables (and their dependencies) are represented by means of a Directed Acyclic Graph. Given two random variables A and B, a directed edge from A to B specifies that the latter is conditioned by the former [9]. The conditional probability $P(B = b_j | A = a_i)$ represents the probability of observing b_j given that a_i has already been observed. Figure 2 presents a sample Bayesian Network in relation with our AMI model.

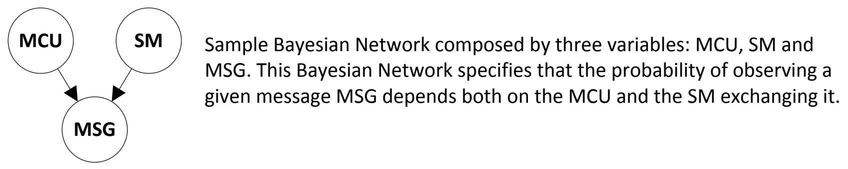

MCU
SM
MSG
Sample Bayesian Network composed by three variables: MCU, SM and MSG. This Bayesian Network specifies that the probability of observing a given message MSG depends both on the MCU and the SM exchanging it.

Fig. 2: Sample Bayesian Network.

3 *METIS* - Overview

This section overviews *METIS*' architecture and presents how adversary goals can be specified by the system expert. Multiple adversary goals can be specified at the same time. For the ease of the exposition, we provide examples that focus on our energy exfiltration use-case.

3.1 Architecture overview

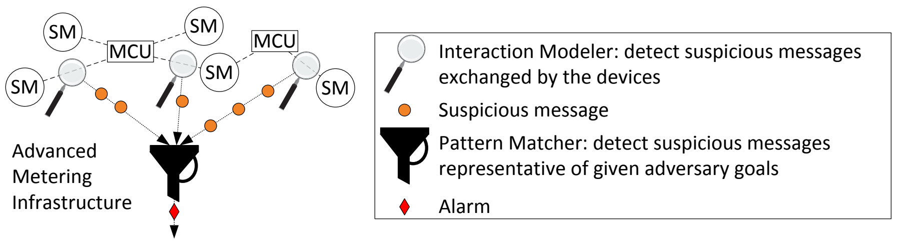

SM
SM
SM
SM
SM
MCU
MCU
Advanced Metering Infrastructure
Interaction Modeler: detect suspicious messages exchanged by the devices
Suspicious message
Pattern Matcher: detect suspicious messages representative of given adversary goals
Alarm

Fig. 3: Overview of *METIS* two-tier architecture.

Millions of messages are generated on a daily basis by the AMI's devices. Such messages carry heterogeneous information related to energy consumption, energy quality and power outages, among others. If we put ourselves in the role of the system expert, it might be hard to specify how evidence of a given adversary goal could be detected while processing such traffic as a whole. The work required by the system expert can be simplified by splitting it into two narrower tasks: (i) specify how an adversary goal could affect the interaction of certain types of devices (possibly belonging to different networks) and (ii) specify the pattern of suspicious interactions that could be observed over a certain period of time. This decomposition would also ease the deployment of a scalable distributed and parallel traffic analysis. The interaction of the devices could be studied close to the devices themselves (i.e., embracing the different networks of an AMI and monitoring the potentially huge amounts of traffic in parallel). Based on these observations, we designed *METIS* to analyze the AMI traffic by means of two tiers: the *Interaction Modeler* and the *Pattern Matcher* (as presented in Figure 3). Among its benefits, this two-tier architecture allows for other IDS to be *plugged* into the framework (e.g., by replacing the provided *Interaction Modeler* with a specification-based IDS such as [1]).

Interaction Modeler This tier analyzes the messages received and sent by each device and relies on anomaly-based detection to distinguish the ones that are expected from the *suspicious* ones.

The anomaly-based technique employed by the *Interaction Modeler* distinguishes between expected and *suspicious* messages based on the probability of observing them. It should be noticed that such probability evolves over time and is potentially influenced by several factors. As an example, the probability of observing an ECReq message could depend on the MCU forwarding it, on the SM receiving it, on the quality of the communication between these two devices, and so on.

If we tackle this aspect from the system expert point of view, it is desirable to have an intuitive way of specifying with traffic features should be taken into account for a given adversary goal. To our advantage, Bayesian Networks (BNs) provide an effective and graphical way of representing such features and their inter-dependencies. At the same time, BNs can also be automatically translated into data streaming queries, as we discuss in Section 4.2.

Since *METIS* relies on the data streaming processing paradigm, probabilities are maintained over a window of size $\mathrm{IM}_{\mathrm{WS}}$ and advance $\mathrm{IM}_{\mathrm{WA}}$ (specified by the system expert), thus coping with the evolving nature of AMIs. $\mathrm{IM}_{\mathrm{WS}}$ represents the period of time during which traffic should be observed in order to have representative probabilities. $\mathrm{IM}_{\mathrm{WA}}$ specifies the amount of information that should be discarded each time the window slides. As an example, if parameters $\mathrm{IM}_{\mathrm{WS}}$ and $\mathrm{IM}_{\mathrm{WA}}$ are set to 12 months and 1 months, respectively, probabilities based on the traffic observed during the last year would be produced every month.

Pattern Matcher The anomaly-based detection mechanism employed by the *Interaction Modeler*, based on the probability with which messages are expected, can result in legitimate messages being considered as *suspicious*. As an example, this could happen when lossy communication between a pair of devices leads to a low expectation associated to a certain legitimate message. For this reason, the *Pattern Matcher* consumes the *suspicious* messages forwarded by the *Interaction Modeler* in order to distinguish the ones that are isolated from the ones representative of a given adversary goal, raising an *alarm* in the second case.

The system expert is required to specify how *suspicious* messages should be processed by means of four parameters. An *alarm* is raised if a threshold T of *suspicious* messages sharing the same values for the set of attributes GB are observed given a window of size $\mathrm{PM}_{\mathrm{WS}}$ and advance $\mathrm{PM}_{\mathrm{WA}}$.

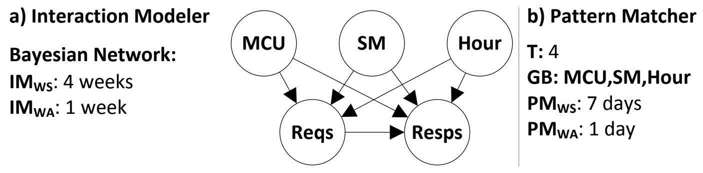


Fig. 4: Input provided by the system expert for *METIS*' *Interaction Modeler* and *Pattern Matcher*

3.2 Energy exfiltration use-case

Interaction Modeler Given our adversary model for the energy exfiltration use-case, the malicious traffic would result in an unusual exchange of ECReq and ECResp messages between a pair of MCUs and SMs. Hence, the system expert could define a BN composed by two variables: Reqs (the number of ECReq messages observed in the window) and Resps (the number of ECResp messages observed in the window), with Reqs being a conditional variable for Resps. In our model, SMs are not statically connected to MCUs. Moreover, energy consumption readings can be retrieved multiple times at different hours during each day (the hour actually depends on the MCU). For this reason, more variables could be added to the BN, as shown in Figure 4.a. Since SMs do not change the MCU to which they connect on a daily basis, a window of four weeks (IM_{WS}=4 weeks) updated every week (IM_{WA}=1 week) could be long enough to detect unexpected exchanges of ECReq and ECResp messages.

Pattern Matcher As discussed in Section 2.3, the adversary is willing to collect energy consumption readings over a certain number of days in order to infer detailed information about the victim's household activities. In this example (Figure 4.b), the system expert specifies that an *alarm* should be raised if at least four suspicious messages (T=4) are observed for the same MCU, SM and hour (GB=MCU,SM,Hour) given a window of size seven days (PM_{WS}=7 days) and advance one day (PM_{WA}=1 day).

4 Detecting anomalies by means of continuous queries

As discussed in Section 3.1, one of the motivations of *METIS* is to ease the system expert's interaction with the framework. For this reason, *METIS* decouples the semantics of the analysis from its actual implementation and deployment. That is, it requires the expert to specify how to detect a given adversary goal by means of a BN and a set of parameters, while it is responsible for compiling such information into a data streaming query. In the following sections we overview the processing carried out by the query, also discussing how the BN is learnt by means of data streaming operators.

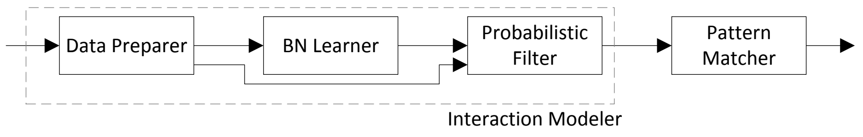


Fig. 5: Overview of the query created by *METIS*.

4.1 Continuous query - overview

Both the traffic analysis of the *Interaction Modeler* and the *Pattern Matcher* are carried out by a single data streaming query compiled by *METIS*. For the ease of the exposition, we present this query by means of four modules: the *Data Preparer*, the *BN Learner*, the *Probabilistic Filter* and the *Pattern Matcher* (as presented in Figure 5). The first three modules perform the analysis of the *Interaction Modeler* while the last module is responsible for the analysis of the *Pattern Matcher*.

The *Data Preparer* pre-processes the information required to learn the given BN. It relies on a Filter operator to discard messages that are not relevant for the BN and on an Aggregate operator to aggregate the information based on the BN's variables. The tuples forwarded by the *Data Preparer* are consumed by the *BN Learner*, in charge of maintaining the probabilities over the window of size IM_{WS} and advance IM_{WA}. The exact number of operators that compose the *BN Learner* depends on the number of variables specified by the BN, as we discuss in the following section. The tuples produced by the *BN Learner* associate the messages observed during the given window to a certain probability. This information is processed, together with the information produced by the *Data Preparer*, by the *Probabilistic Filter*. As discussed in Section 2.2, the evolving nature of AMIs demands for continuous learning. For this reason, the *Probabilistic Filter* compares each tuple produced by the *Data Preparer* with its associated probability learned over the latest completed window. As an example, if parameters IM_{WS} and IM_{WA} are set to 10 and 5 time units, respectively, the window will cover periods $P_1 = [0, 10)$, $P_2 = [5, 15)$, $P_3 = [10, 20)$, and so on. Messages observed in period $[10, 15)$ would be matched with the probabilities learned during period P_1, messages observed in period $[15, 20)$ would be matched with the probabilities learned during period P_2, and so on. A tuple produced by the *Data Preparer* is forwarded by the *Probabilistic Filter* based on a probabilistic trial. As an example, if the probability learned for a certain message is 0.9, such a message will be forwarded with a probability equal to 0.1. Tuples forwarded by the *Probabilistic Filter* represent the tuples considered as *suspicious* by the *Interaction Modeler*. As discussed in Section 3.1, an *alarm* is raised if at least T *suspicious* messages sharing the same values for the set of attributes GB are observed given a window of size PM_{WS} and advance PM_{WA}. The *Pattern Matcher* relies on an Aggregate operator to count how many *suspicious* messages sharing the same values for the set of attributes GB are received given a window of size PM_{WS} and advance PM_{WA}. A Filter operator is used to filter only the tuples produced by the Aggregate operator whose counter is greater than or equal to T. We provide an example of the continuous query associated to the energy exfiltration use-case in Section 5.

4.2 Learning BNs by means of data streaming operators

The number of operators composing the *BN Learner* depends on the variables defined for the BN. As we discuss in the following, the ability to automatically convert a BN to a query boils down to the ability of computing the probabilities of its variables by means of data streaming operators.

Given two discrete variables X such that $supp(X) \in \{x_0, x_1, \dots, x_m\}$ and Y such that $supp(Y) \in \{y_0, y_1, \dots, y_n\}$ and a sequence S of observations $o_1, o_2, \dots$ such that $o_s = \langle x_i, y_j \rangle$ and all observations belong to the same window, the conditional probability can be computed as

$$P(Y = y_j | X = x_i) = \frac{|\{o_s \in S | o_s = \langle x_i, y_j \rangle\}|}{|\{o_s \in S | o_s = \langle x_i, . \rangle\}|}$$

In order to compute such a value, we need to count the number of occurrences of each pair $\langle x_i, y_j \rangle$ and each value x_i. In terms of data streaming operators, these numbers can be maintained by two Aggregate operators. The first Aggregate operator would count the occurrences of each pair $\langle x_i, y_j \rangle$. Similarly, the second Aggregate operator would count the occurrences of each value x_i. Subsequently, values referring to the same x_i value could be matched by an EquiJoin operator and the resulting division computed by a Map operator.

Figure 6 presents a sample execution of the operators for a given sequence of tuples. In the example, variable X assumes values $\{x_0, x_1\}$ while variable Y assume values $\{y_0, y_1\}$. In the example, the windows' size and advance parameters are both set to 10 time units.

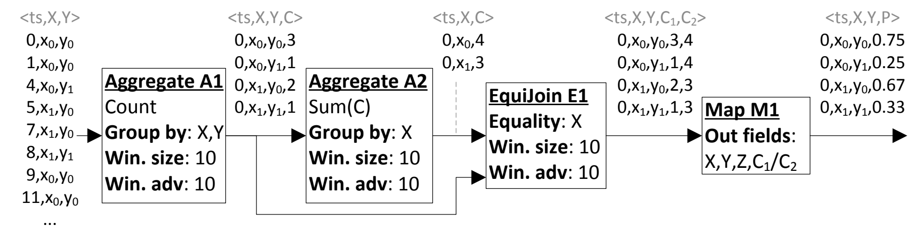
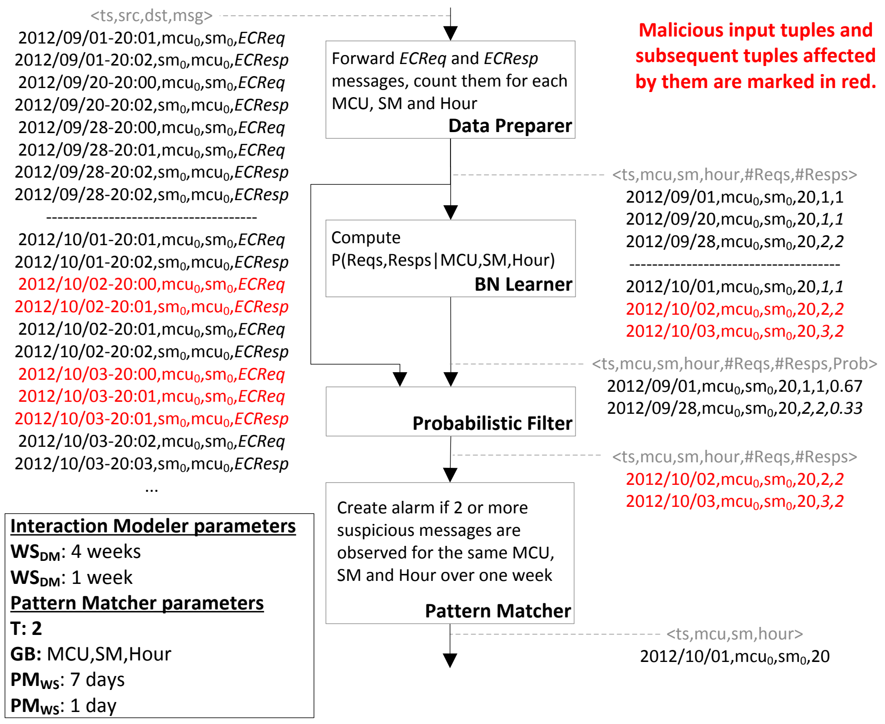


Fig. 6: Continuous query used to compute $P(Y|X)$. The figure includes the abstract schema and a set of sample tuples for each stream.

5 Energy exfiltration use-case - Sample Execution

In this section, we provide a sample execution of the continuous query compiled by *METIS*, given the BN and the parameters presented in Section 3.2. The query is presented in Figure 7. For the ease of the exposition, we focus on the messages exchanged between a single pair of MCUs and SMs, $\langle mcu_0, sm_0 \rangle$.


Malicious input tuples and subsequent tuples affected by them are marked in red.
Forward ECReq and ECResp messages, count them for each MCU, SM and Hour
Data Preparer
Compute P(Reqs,Resps|MCU,SM,Hour)
BN Learner
Probabilistic Filter
Create alarm if 2 or more suspicious messages are observed for the same MCU, SM and Hour over one week
Pattern Matcher
Interaction Modeler parameters
WS_DM: 4 weeks
WS_DM: 1 week
Pattern Matcher parameters
T: 2
GB: MCU,SM,Hour
PM_WS: 7 days
PM_WS: 1 day

```
<ts,src,dst,msg>
2012/09/01-20:01,mcu0,sm0,ECReq
2012/09/01-20:02,sm0,mcu0,ECResp
2012/09/20-20:00,mcu0,sm0,ECReq
2012/09/20-20:02,sm0,mcu0,ECResp
2012/09/28-20:00,mcu0,sm0,ECReq
2012/09/28-20:01,mcu0,sm0,ECReq
2012/09/28-20:02,sm0,mcu0,ECResp
2012/09/28-20:02,sm0,mcu0,ECResp
--------------------------------------
2012/10/01-20:01,mcu0,sm0,ECReq
2012/10/01-20:02,sm0,mcu0,ECResp
2012/10/02-20:00,mcu0,sm0,ECReq
2012/10/02-20:01,sm0,mcu0,ECResp
2012/10/02-20:01,mcu0,sm0,ECReq
2012/10/02-20:02,sm0,mcu0,ECResp
2012/10/03-20:00,mcu0,sm0,ECReq
2012/10/03-20:01,mcu0,sm0,ECReq
2012/10/03-20:01,sm0,mcu0,ECResp
2012/10/03-20:02,mcu0,sm0,ECReq
2012/10/03-20:03,sm0,mcu0,ECResp
...

<ts,mcu,sm,hour,#Reqs,#Resps>
2012/09/01,mcu0,sm0,20,1,1
2012/09/20,mcu0,sm0,20,1,1
2012/09/28,mcu0,sm0,20,2,2
--------------------------------------
2012/10/01,mcu0,sm0,20,1,1
2012/10/02,mcu0,sm0,20,2,2
2012/10/03,mcu0,sm0,20,3,2

<ts,mcu,sm,hour,#Reqs,#Resps,Prob>
2012/09/01,mcu0,sm0,20,1,1,0.67
2012/09/28,mcu0,sm0,20,2,2,0.33

<ts,mcu,sm,hour,#Reqs,#Resps>
2012/10/02,mcu0,sm0,20,2,2
2012/10/03,mcu0,sm0,20,3,2

<ts,mcu,sm,hour>
2012/10/01,mcu0,sm0,20
```

Fig. 7: Sample execution of the query compiled for the energy exfiltration use-case. The figure includes the abstract schema and a set of sample tuples for each stream.

The *Data Preparer* module relies on its Filter operator to forward only ECReq and ECResp messages. These messages are then consumed by the Aggregate operator, in charge of counting how many ECReq and ECResp messages are exchanged between each MCU and SM and for each hour. In the example, malicious messages (injected by the adversary) are marked in red. As shown in the figure, an exchange of a single ECReq and a single ECResp message is observed twice while an exchange of two ECReq and two ECResp messages is observed only once during the month of September. Similarly, exchanges of one ECReq and one ECResp messages, two ECReq and two ECResp messages, and three ECReq and two ECResp messages are observed once during the month of October. The last two tuples produced by the Aggregate operator are marked in red since they are influenced by the malicious input messages.

The probability of observing each combination is computed by the *BN Learner* module. The probability of observing an exchange of one ECReq and one ECResp messages is 67% while the probability of observing an exchange of two ECReq and two ECResp messages is 33%. The probabilities computed by the *BN Learner* and the tuples produced by the *Data Preparer* are matched by the *Probabilistic Filter*. As discussed in Section 4.1, each tuple produced by the *Data Preparer* is matched with its associated probability observed in the latest completed window. In the example, tuples produced during the month of October will be matched with the probabilities observed for the month of September. Tuples $\langle 2012/09/01, mcu_0, sm_0, 20, 1, 1\rangle$, $\langle 2012/09/02, mcu_0, sm_0, 20, 2, 2\rangle$ and $\langle 2012/09/03, mcu_0, sm_0, 20, 3, 2\rangle$ have a probability of 0.33, 0.67 and 1, respectively, of being considered as suspicious. In the example, tuples $\langle 2012/09/02, mcu_0, sm_0, 20, 2, 2\rangle$ and $\langle 2012/09/02, mcu_0, Sm_0, 20, 3, 2\rangle$ are considered as suspicious and forwarded. Since the threshold T is set to two, an alarm is raised by the *Pattern Matcher*.

6 Energy Exfiltration use-case - Evaluation

In this section we evaluate *METIS* with respect to our energy exfiltration use-case and show that (i) it is able to detect malicious activity and that (ii) it can be leveraged by relying on commodity hardware. We first present the evaluation setup, discussing the real world AMI from which data is extracted and the attack injection methodology for the energy exfiltration attacks. We continue by presenting the detection accuracy for a given configuration of the *Interaction Modeler* and the *Pattern Matcher*, also discussing how different configurations affect their detection capabilities. Subsequently, we evaluate the processing capacity of *METIS* (in terms of throughput and latency) when executed by a server that could be deployed at the utility head-end.

6.1 Testbed and dataset description

METIS has been implemented on top of Storm, version 0.9.1. The continuous query (*topology* in Storm's terminology) is composed by fourteen operators. The real-world AMI used in our evaluation is composed by 300,000 SMs that communicate with 7,600 MCUs via IEEE 802.15.4 and ZigBee. The network covers a metropolitan area of 450 km^2 with roughly 600,000 inhabitants. The utility extracted data for a subset of 100 MCUs that communicate with approximately 6,500 SMs and made it available for us. The input data covers a period of six months ranging from September 2012 to February 2013. To the best of our knowledge, this dataset is free from energy exfiltration attacks. SMs are not statically linked to MCUs. At the same time, SMs appear and disappear (e.g., because of new installations or decommissioning). MCUs are in charge of collecting energy consumption readings at different hours, usually two or three times per day (the hours at which the collection happens is specific for each MCU). Due to the wireless communication, it is common for MCUs and SMs to lose messages that are thus forwarded multiple times. Each MCU has a maximum of three attempts per hour to retrieve the energy consumption of a given SM. The information kept by the utility does not contain the exact number of messages exchanged for a given MCU, SM and day. Nevertheless, we are able to compute the probabilities with which a message is lost (and hence sent again) based on the logs stored by the MCUs. The ECReq and ECResp messages for each MCU, SM and day are simulated based on such probabilities.

In order to inject adversary traffic, we randomly pick a MCU-SM pair and, during a period that goes from seven to ten days, we inject ECReq and ECResp messages. In total, we inject 50 energy exfiltration attacks, resulting in 995 malicious messages. Note that these messages are subject to the same probability of being lost as any legitimate message. Furthermore, in order to simulate the behavior of a subtle adversary, malicious messages are exchanged at the same hour at which the MCU is actually retrieving energy consumption readings (as it would be trivial to detect an energy exfiltration attack if messages are exchanged when the MCU is not supposed to communicate).

6.2 Detection Accuracy

In this experiment, the BN is the one presented in Section 3.2. The *Interaction Modeler*'s parameters IM_{WS} and IM_{WA} are set to four weeks and one week, respectively. The *Pattern Matcher*'s window parameters PM_{WS} and PM_{WA} are set to seven days and one day, respectively. The *Pattern Matcher* is instructed to raise an alarm if at least a threshold T of five suspicious messages sharing the same values for the set of attributes MCU,SM and Hour is observed. A summary of the results is presented in Table 1.

AMI data	Number of attacks	50
	Number of malicious messages	995
	Overall number of messages	$4,146,327$
	Messages per day (average)	$23,743$
	Suspicious messages per day (average)	450
Interaction Modeler	Malicious messages considered as suspicious	857
	Malicious messages not considered as suspicious	138
Pattern Matcher	Number of alarms	488
	Alarms, True Positive	245
	Alarms, False Positive	243
	Detected Attacks	45

Table 1: Summary of the *Interaction Modeler*'s and the *Pattern Matcher*'s detection results.

During the six months covered by the data, more than 4.2 million messages are exchanged between the 100 MCUs and the $6,500$ SMs taken into account (more than $23,000$ messages on average on a daily basis). Nevertheless, a small number of approximately 450 messages are considered suspicious on average by the *Interaction Modeler* on a daily basis. 857 out of the 995 malicious messages are considered as suspicious. In total, 488 alarms are raised by the *Pattern Matcher*, 245 of which are related to real attacks (45 attacks are actually detected).

We say an alarm raised by the *Pattern Matcher* is a true positive (resp., false positive), if the period of time covered by its window of size PM_{WS} and advance PM_{WA} actually includes days in which malicious activity has been injected for the given MCU, SM and Hour. It should be noticed that since the window slides every day (PM_{WA} is set to one day), multiple alarms can be raised during consecutive days for one or more suspicious messages referring to a given MCU, SM and Hour. The number of false positives (243) raised during the six months period results in one or two false positives per day, on average. This number of false positives is reasonable for the system expert to use the framework (a reasonable threshold is set to no more than ten false positives per day in [15]). We further analyzed the cause of these alarms and interestingly, most of these false positive alarms are due to new smart meters that appear in the traffic. As this evaluation is based on a real deployment, we can draw the conclusion that the number of devices in this environment is not stable (meaning any assumption of the former would cause false alarms).

6.3 Parameters sensitivity

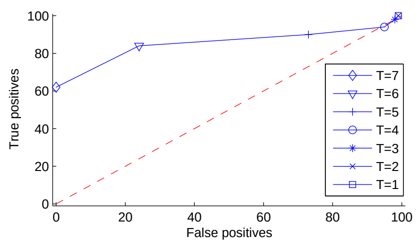
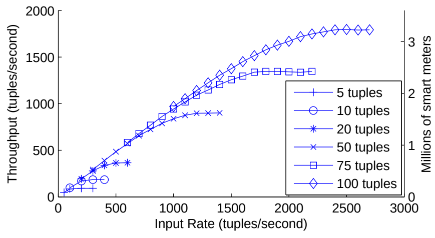


Fig. 8: True Positive and False Positive rates for varying thresholds T.

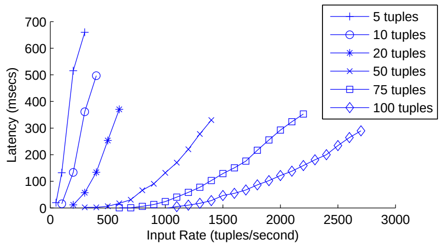
For a given configuration of the *Pattern Matcher*'s parameters PM_{WS}, PM_{WA} and GB, the number of attacks detected by the former depends on the threshold T (i.e., it depends on the number of days during which suspicious messages should be observed in order to raise an alarm). In this section, we present how the true positive and false positive detection rates are affected by varying the values of the threshold T. Since the *Pattern Matcher*'s Aggregate window size (PM_{WS}) is set to seven days, the experiments are run for $T = 1, \ldots, 7$. As presented in Figure 8, the minimum true positive rate is achieved when parameter T is equal to seven. In this case, no false positive alarms are raised by the *Pattern Matcher*. It can be noticed that the true positive rate increases to more than 80% when $T \leq 6$, while it grows to more than 90% when $T \leq 4$.

6.4 Processing capacity

As shown above, *METIS* is able to detect the majority of the energy exfiltration attacks we injected. In this section, we show it can also cope with the large



(a) Throughput (tuples/second).



(b) Latency (milliseconds).

Fig. 9: Throughput and latency for increasing input rates and batch sizes.

volume of events produced in a typical AMI. For that reason, we evaluate the processing capacity of *METIS* when running on a server that could be deployed at the utility's head-end, an Intel-based workstation with two sockets of 8-core Xeon E5-2650 processors and 64 GB DDR3 memory.

Among the different parameters that could influence the processing capacity of the query, the *batch* size plays a fundamental role in this context. While processing messages, a trade-off exists between the rate at which such messages can be processed and the latency imposed by the processing itself. In high-throughput systems, it is common to group tuples together in batches (of thousands or tens of thousands of tuples) in order to achieve higher throughput. Nevertheless, this is not an option in our scenario. Each pair of devices exchanges a small number of messages per hour (in the order of tens). If the analysis relies on big batches (e.g., thousands of messages), devices might not be able to log incoming and outgoing messages for the resulting large periods of time and possible attacks would thus not be detected.

Figure 9a presents the processing throughput for different batch sizes, from 5 to 100 tuples. As expected, increasing the batch size results in higher processing throughput. For a batch size of 100 tuples, the server is able to process approximately 2,000 messages per second. Based on our data, each pair of MCU and SM exchanges one ECReq and one ECResp message each time energy consumption is retrieved. If the 2,000 messages processed every second refer to the exchange of 1,000 pairs of MCUs and SMs, the processing capacity of our prototype would enable the monitoring of more than three millions pairs of MCU and SM every hour. Figure 9b presents the corresponding latency (in milliseconds) for the different batch sizes. While a common pattern is observed for all batch sizes (the latency starts increasing when the throughput gets closer to its maximum), it can be noticed that the highest measured latency is of approximately 0.7 seconds. This means that the latency in the detection of an attack would depend on the frequency with which energy consumption readings are retrieved rather than the (negligible) processing time introduced by *METIS*' analysis.

7 Related Work

Despite their recent deployment, a considerable number of potential attacks against AMIs has already been discussed in literature where some have even been seen in the wild [13]. The attacks range from energy theft [16], stealing of users' information [2], up to physical damage of the infrastructure [4].

As outlined in [14], traditional IDSs cannot be used effectively in these environments without major modifications. However, even though there exists a large literature on intrusion detection in general, very few systems have been developed specifically for AMIs. Several papers motivate the need for security in smart grids (where [5] is such an example); others go one step further and discuss detection mechanisms but often concentrating on other parts of the smart grid (such as attack detection in SCADA networks [3], or for process control [12]). Berthier et al. [2] discuss requirements with an outline of a possible intrusion detection architecture suitable for AMIs. To the best of our knowledge, specification-based IDSs are the main defense mechanism proposed so far for AMIs [1, 18].

One advantage with our approach is that several detection mechanisms can be used as a sensor in the first tier (the *Interaction Modeler*), meaning that the previously suggested specification-based approaches for AMIs could also be integrated into our framework. However, in this paper we instead suggested Bayesian inference in the first tier. Using Bayesian networks to model attacks merges the best properties of the signature-based approach with the learning characteristics of anomaly detection [26]. A specification-based IDS would require manual labor to tune the system to a specific installation, where a Bayesian attack model would be (relatively) easy to create for the system expert with the added benefit that we automatically can parallelize it in METIS by relying on the data streaming paradigm. Specification-based systems work best in very stable environments; in AMIs it is expected that the traffic will be more dynamic and less deterministic in the future with demand-side networks, as described in [14].

Using several tiers of sensors and analysis engines to improve the detection has been used in traditional IDSs such as [25, 21]. Our motivation for having different tiers is that they allow for the implementation of the *Interaction Modeler* to be isolated from the overall event processing. As mentioned above, this approach makes the design and implementation of the attack models easier. The second tier manages the scalability of the approach to allow for the analysis of the underlying traffic in real time.

As discussed in [2, 10], the coexistence of distinct networks within the same AMI demands for distributed traffic analysis, either by relying on the devices themselves (as recently investigated in [20]) or by relying on dedicated sensing infrastructures. To this end, the data streaming processing paradigm [23] is an optimal candidate for AMIs traffic analysis, as explored in [22, 27, 6]. The latter is the closer to our approach, but their evaluation is not based on data from realistic AMIs.[3]

[3] They use the KDD Cup 99 dataset, with known problems (`http://www.kdnuggets.com/news/2007/n18/4i.html`) as well as lacking realistic AMI attacks.

8 Conclusions

This work proposed *METIS*, a two-tier defense framework that eases the modeling of possible adversary goals and allows for a scalable traffic analysis by employing the data streaming processing paradigm. The proposed architecture allows for modular functionality and configurable deployment, allowing also for complementary intrusion detection systems to be integrated in the framework (e.g., by replacing the first tier). In the paper, besides describing and analyzing its design and implementation, we showed how it is possible for a system expert to model the detection of energy exfiltration attacks, a challenging adversarial goal. Moreover, through the evaluation of the use-case based on big volumes of data extracted from a real world AMI, we showed that *METIS*' analysis can achieve high detection rates, with low false alarm numbers, even when relying on commodity hardware.

It is worth pointing out that the possibility for distributed deployment of *METIS* enables for the detection of a variety of scenarios, including those whose detection is only possible through distributed evidence. The latter opens a path for new research in detecting and mitigating adversarial actions in AMIs, where for scalability and privacy purposes it can be imperative to detect unwanted situations close to the data sources, without the need to store the original data.

References

1. R. Berthier and W. H. Sanders. Specification-based intrusion detection for advanced metering infrastructures. In *IEEE 17th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2011.
2. R. Berthier, W. H. Sanders, and H. Khurana. Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. In *Smart Grid Communications (SmartGridComm), First IEEE International Conference on*, 2010.
3. S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes. Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA security scientific symposium*, 2007.
4. M. Costache, V. Tudor, M. Almgren, M. Papatriantafilou, and C. Saunders. Remote control of smart meters: friend or foe? In *Computer Network Defense (EC2ND), Seventh European Conference on*, 2011.
5. G. N. Ericsson. Cyber security and power system communicationessential parts of a smart grid infrastructure. *Power Delivery, IEEE Transactions on*, 2010.
6. M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez. Securing advanced metering infrastructure using intrusion detection system with data stream mining. In *Intelligence and Security Informatics*. Springer, 2012.
7. N. Falliere, L. O. Murchu, and E. Chien. W32. stuxnet dossier. Technical report, Symantec Corporation, 2011.
8. FORWARD Consortium, White book: Emerging ICT threats. `http://www.ict-forward.eu/media/publications/forward-whitebook.pdf`.
9. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 1997.

10. D. Grochocki, J. H. Huh, R. Berthier, R. Bobba, W. H. Sanders, A. A. Cárdenas, and J. G. Jetcheva. AMI threats, intrusion detection requirements and deployment recommendations. In *Smart Grid Communications (SmartGridComm), IEEE Third International Conference on*, 2012.
11. V. Gulisano, R. Jimenez-Peris, M. Patino-Martinez, C. Soriente, and P. Valduriez. Streamcloud: An elastic and scalable data streaming system. *Parallel and Distributed Systems, IEEE Transactions on*, 2012.
12. D. Hadiosmanovic, D. Bolzoni, P. Hartel, and S. Etalle. MELISSA: Towards Automated Detection of Undesirable User Actions in Critical Infrastructures. In *Computer Network Defense (EC2ND), Seventh European Conference on*, 2011.
13. KrebsonSecurity. FBI: Smart Meter Hacks Likely to Spread. `http://krebsonsecurity.com/2012/04/fbi-smart-meter-hacks-likely-to-spread/`, April 2012.
14. N. Kush, E. Foo, E. Ahmed, I. Ahmed, and A. Clark. Gap analysis of intrusion detection in smart grids. In *Proceedings of the 2nd International Cyber Resilience Conference*, 2011.
15. R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer networks*, 2000.
16. S. McLaughlin, D. Podkuiko, and P. McDaniel. Energy theft in the advanced metering infrastructure. In *Critical Information Infrastructures Security*. Springer, 2010.
17. S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel. Multi-vendor penetration testing in the advanced metering infrastructure. In *Proceedings of the 26th Annual Computer Security Applications Conference*, 2010.
18. R. Mitchell and I.-R. Chen. Behavior-Rule Based Intrusion Detection Systems for Safety Critical Smart Grid Applications. *Smart Grid, IEEE Transactions on*, 2013.
19. A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, 2010.
20. M. Raciti and S. Nadjm-Tehrani. Embedded cyber-physical anomaly detection in smart meters. In *Critical Information Infrastructures Security*. Springer, 2013.
21. S. A. Razak, S. Furnell, N. Clarke, and P. Brooke. A Two-Tier Intrusion Detection System for Mobile Ad Hoc Networks A Friend Approach. In *Intelligence and Security Informatics*, Lecture Notes in Computer Science. Springer, 2006.
22. Y. Simmhan, B. Cao, M. Giakkoupis, and V. K. Prasanna. Adaptive rate stream processing for smart grid applications on clouds. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, 2011.
23. M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, 2005.
24. Storm project. `http://storm.incubator.apache.org/`. Accessed: 2014-03-10.
25. E. Tombini, H. Debar, L. Mé, and M. Ducassé. A serial combination of anomaly and misuse IDSes applied to HTTP traffic. In *Computer Security Applications Conference. 20th Annual*, 2004.
26. A. Valdes and K. Skinner. Adaptive, Model-Based Monitoring for Cyber Attack Detection. In *Recent Advances in Intrusion Detection*, Lecture Notes in Computer Science. Springer, 2000.
27. D. Zinn, Q. Hart, T. McPhillips, B. Ludascher, Y. Simmhan, M. Giakkoupis, and V. K. Prasanna. Towards reliable, performant workflows for streaming-applications on cloud platforms. In *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2011.