A Response Cost Model for Advanced Metering Infrastructures

Ahmed Fawaz, *Student Member, IEEE,* Robin Berthier, *Member, IEEE,* and William H. Sanders, *Fellow, IEEE*

***Abstract*—The smart grid is driving the deployment of networked components within the traditional power grid to achieve greater reliability and control. However, those cyber components are creating new security vulnerabilities that could lead to significant disruptions if exploited maliciously. We propose a comprehensive cost model designed to assist in the selection of responses to malicious attacks on an Advanced Metering Infrastructure (AMI). The cost model reflects the costs of response actions due to outages in the distribution grid, disruptions of the electricity market, and violations of contractual agreements among stakeholders. We also present a realistic implementation of the cost model using ArcGIS for topology generation and GridLAB-D for grid simulation. The cost model is a step forward towards achievement of automated resilience in the smart grid. It also provides comprehensive modeling for system dynamics and operation costs, while relieving utilities from the burden of manually assigning costs for response actions and modeling attacker steps.**

***Index Terms*—Security; Security assessment tools; Intrusion tolerance; Critical infrastructures**

I. Introduction

The worldwide deployment of smart grids, in which communication technologies are integrated within power delivery infrastructures, has fueled many research efforts on the topic of cybersecurity. The majority of those efforts are dedicated to designing either new protective measures, such as better encryption and authentication algorithms or stronger network isolation systems, or new detection solutions, such as specification-based intrusion detection systems or large-scale analytical algorithms that report incidents quickly and accurately. However, research efforts have been limited with respect to automated response to cyber incidents. Initial studies have explored use of techniques such as game theory [1], but the path between theoretical results and practical deployment has still not been bridged. That may appear surprising, given the criticality of automated response to cyber incidents in large cyberphysical systems and the tendency of the power grid community to rely on automated protection devices. In fact, the acts of tolerating failures and protecting against cyber intrusions are quite different because of the nature (accidental vs. malicious) of the causes of the underlying impairments. Indeed, designing a comprehensive response system that can take into account alerts from both cyber and power sensors to automatically decide on optimal recovery actions is a significant challenge. A major element of the challenge is the requirement for a cost model that can accurately predict the costs of possible response actions.

The authors are with the University of Illinois at Urbana-Champaign, Urbana, IL, 61801 USA e-mail: (see http://www.perform.illinois.edu/).

The goal of this paper is to make progress towards intrusion tolerance by introducing a practical cost model in the context of Advanced Metering Infrastructures (AMIs). An AMI serves as the communication backbone for smart meters and enables new applications, such as frequent remote meter reading or demand response features. Cyber threats against an AMI have the potential to penetrate the utility network or to exploit meters to disrupt the power grid. The significant number of devices in a typical AMI makes manual response to incidents nearly impossible. As a result, it is necessary to develop an accurate cost model that allows systems to support automated decision-making.

In particular, we propose a service-oriented cost model that uses a dependency graph to model the system. It computes the cost of a response action using the deterioration of quality of services and possible attack consequences. We describe the quality of service using the traditional security attributes: confidentiality, integrity, and availability (CIA). Finally, we convert these to a financial cost. In some cases, service-level agreements (SLAs) can be used to directly map degradation in a security attribute to financial cost. For example, loss of availability can be penalized. However, some attributes (like integrity) are not likely to be covered by an SLA; we compute the consequences on such losses on the business process. The consequences are system-dependent and could include possible attacks, and loss of production, among other effects.

This paper makes the following contributions:

- We introduce a practical cost model workflow that can translate system logs and cyber alerts into response costs.
- We propose a method to model the service level of services using dependency graph and low-level components.
- We provide a method to generate a dependency graph from an AMI routing topology, and we explore the impact of the change of security state in the context of an attack.
- We show case the cost model using response selection during an attack scenario, and we assess the potential disruption that a given number of compromised meters could cause.

II. Related Work

To the best of our knowledge, we are the first to study the concept of a response cost model in the context of the smart grid. [1] has introduced the Response and Recovery Engine (RRE) and shown how RRE can be used in a power grid environment, but the concept of cost is left as a parameter in the model that needs to be provided by operators. In the general context of traditional IT systems, that idea of using static costs has been studied by [2], [3], [4], [5], [6], [7], where

TABLE I
TERMS AND NOTATIONS

AMI	Advanced Metering Infrastructure
ANSI	American National Standards Institute
CIA	Confidentiality-Integrity-Availability
CR	Collecting Router
EAX	Block Cipher Mode
GIS	Geographic Information System
HE	Head-End
IDS	Intrusion Detection System
RRE	Response and Recovery Engine
SAIDI	System Average Interruption Duration Index
SLA	Service-Level Agreement
TCP/IP	Transmission Control Protocol/Internet Protocol
WAN	Wide Area Network

the approach consists of generating a taxonomy of possible response actions and then assigning a static cost value to each action. The main limitations of that approach are that it requires operators to estimate a large number of parameters, and it does not capture dynamic costs due to system changes.

Other researchers [8], [9] went beyond static costs by decomposing the cost of actions into different sub-costs to better capture the impact of actions on the system. Luo et al. [10] describe a solution for linearly increasing static costs over time, while Anuar et al. [11] leverage an analytical hierarchy process to compute impact factors in addition to using static costs. Those approaches reduce the inflexibility of previous static techniques, but they still suffer from requiring a large number of parameters that are not updated based on the dynamic state of the system.

One approach to that challenge has been to explore dependency graphs. The authors of [12] dynamically update the availability of the system by propagating in the graph the impact of nodes' becoming unavailable because of response actions. The authors of [13] extended that approach by using three separate graphs to cover confidentiality, integrity, and availability properties. Kheir et al. [14] combined those graphs into a unified formalism by labeling nodes with a vector and by using a matrix to capture how the different security properties depend on each other.

We build on top of those solutions by focusing on their main drawback, namely their need for significant operator involvement to define the many parameters. In addition, our work differs from related studies on dynamic cost models by providing output that can be directly used by an automated reasoning system. We describe a complete workflow that has been tailored for the Advanced Metering Infrastructure, which is one of the largest cyber-physical systems.

Table I defines the acronyms we use throughout the paper.

III. RESPONSE COST MODEL

When accidental failures or malicious events occur, operators of large infrastructures such as AMIs have to make critical decisions in order to recover the system and to keep it operating. A diverse set of sensors provides multiple feeds of input raw data that need to be converted into information that a human operator can comprehend. The objective of a cost model is to perform that conversion task. At a high level, a cost model takes static input, such as system configurations and system topologies, and dynamic input, such as live feeds of alerts and system logs, and identifies the state of the system, from which the financial impact of possible system changes can be computed. System changes include failures, attacks, and security responses. We are particularly interested in the responses; do we automatically provide quantified decision-making support to operators who are responding to cybersecurity events?

When a response action is taken to change the system state, the notion of response cost can be divided into an operational cost (required to perform the action) and an impact cost (as a result of the action), which measures the consequences of an attack. Indeed, any action requires a preparation phase and leads to a consequence phase, and costs can be estimated for both of those phases. Mathematically, the cost of an action is computed with:

$$C_{Action} = C_{Operation} + C_{Impact} \tag{1}$$

The evaluation of those costs requires an in-depth understanding of the system to enable accurate predictions of system behaviors. To gain predictive capabilities, one needs first to design a dependency model from which state changes of a set of components can be precisely trickled down to other components. Several models can be used to define dependency relationships; we chose to use a dependency graph.

Figure 1 presents a high-level framework to tag the dependency graph with input data and cost conversions for the special case of an Advanced Metering Infrastructure.

An important feature of an AMI is that it combines a power infrastructure and a communication infrastructure. We assume that the configuration and topology for both of those infrastructures is static and provided to the model. The AMI routing information captures the communication network topology and is used along with system logs and IDS sensor alerts to generate a dependency graph. From the dependency graph, the security state of the system can be computed and divided into impacts on CIA of the different components. Understanding the security posture of the cybercomponents enables the model to simulate possible failures in the power distribution infrastructure. The simulations are used to compute outage metrics. Finally, the impact on CIA and the magnitude of outages are converted into financial values that can be presented to an operator or to an automated response system as a response cost.

IV. DEPENDENCY GRAPH GENERATION

Our cost model uses a dependency graph of an AMI to evaluate the effect of a response action on the security properties of services in an AMI. The dependency graph, $G = (P, E)$, models the dependency relations between the different components $\mathcal{C}$ and services $\mathcal{S}$ in a system, where $P = \mathcal{C} \cup \mathcal{S}$. The edges in the graph E model the dependency relation between the components and services. Each studied system has its own set of services and components. Moreover, the relationship between the services and components relates to the system configuration. We define the general method for

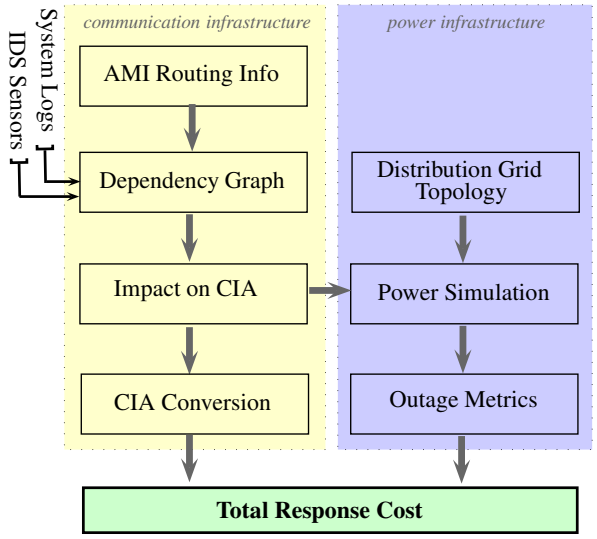

Fig. 1. Overview of the proposed cost model workflow

generating the dependency graphs and also specify the method for creating them for AMIs.

A. Services

In a general computing system, services in a system are implemented via processes running on hosts. For example, in a traditional IT system, those processes could be HTTP services, Kerberos, or an encryption service. Those processes offer the service over the network to other hosts or users.

In our case study of an AMI, we consider the following services: a *meter-reading* service, a *real-time pricing information* service, and a *remote commands* service, $\mathcal{S} = \{\mathcal{S}_{mr}, \mathcal{S}_{pi}, \mathcal{S}_{rc}\}$. We chose those services as a representative set of operations typically sent in AMIs. We note that this list is not exhaustive. Our focus is to propose a framework for response cost that can be easily extended to cover other services.

Figure 2 shows a sample dependency graph containing the meter-reading service. More end-to-end services are performed in an AMI, but we focus on the listed subset in $\mathcal{S}$, as they provide a major portion of AMI traffic.

B. Components

Components are elements of the system that enable a service to function. We define 4 classes of components to understand how they relate to each other: data sources and sinks, cipher elements, and routing elements. We use $D_{(\text{location,service})}$ to describe the data elements. For example $D_{(\text{ID,mr})}$ is the data element for the meter reading service in a meter, where ID is an identifier used to specify a unique meter. The identification of the components of each service defines the dependency relationship in the graph.

Smart meters implement ANSI C12.19 for data storage and ANSI C12.22 [15], [16] for communication. ANSI C12.19 is the table structure used in smart meters to store information such as configurations, meter readings, keys, and function calls. ANSI C12.22 is an application-layer protocol used to transport C12.19 table data over a network; it defines session handling, security modes, and message exchanges. C12.22 data could be transported over TCP/IP [17] or IEEE 802.15.4. All services in an AMI are implemented as reads and writes to the C12.19 tables and flows of C12.22 messages between a meter and the head-end. The head-end is an application that manages meter communication and membership. It connects to the meters via routing nodes; this connection could be over WAN or private lines. The head-end is typically located in the headquarter of a distribution utility. For example, to disconnect a meter, a C12.22 disconnect message is sent. The message is a full write message that sets a specific entry in a C12.19 table to `0x00`. In our AMI case study, the components, $\mathcal{C}$, are the meters, table entries, messages/links, and head-end components used by each service. Since data entries have different semantics, we differentiate among the table entries and define them as separate components in the dependency graph, as shown in Figure 2.

- **A meter-reading service** in an AMI generates metering data due to usage, stores the data in a table entry $D_{(\text{m,mr})}$, and sends the data to the head-end using C12.22 messages. The data are stored at the head-end's database $D_{(\text{he,mr})}$.
- **A real-time pricing service** in an AMI writes real-time pricing information received from the head-end's database $D_{(\text{he,pi})}$ through C12.22 messages to a table entry $D_{(\text{m,pi})}$.
- **A remote commands service** in an AMI writes special values in the table entry $D_{(\text{m,dr})}$ when specific C12.22 write messages are received from the head-end. That launches special functions that control appliances and turn services on and off remotely.

Finally, some services use the cipher components for encryption and authentication of the messages exchanged between the meters and head-end. The basic building components of those operations are keys, ciphers, and hash functions. In our AMI study case, the cipher operations are defined in the C12.22 standard security mechanism [15]. Standard C12.22 security uses the EAX mode for authentication and protects the privacy of the message using a shared *key*. Our dependency graph models the security mode through a set of keys $\mathcal{K}$ and cipher $E_{k_i}(m)$. A key $k_i \in \mathcal{K}$ is shared between a meter M_i and the head-end. The head-end has the following high-level components: 1) a database $D_{(\text{he},S(m_i))}$, which stores the data for each service for each meter; 2) meter keys k_i's; 3) a utility key (pk, sk); and 4) an encryption-decryption engine ED_{eng}, which performs the cipher. It is necessary to describe the key management and generation process, because it affects the state of the cipher components. Key management is usually proprietary, with each vendor designing its own scheme. However, any key management scheme, whether based on symmetric or asymmetric keys has the same functions. They include key generation, distribution, key revocation, and key replacement. For our purposes, we are not concerned with the exact implementations for those functions, as long as they perform their intended functionality.

C. Graph Generation

The dependency relation $\mathbb{D} = P \times P$ defines the transitive relation among the components and services in G. That is, $(a, b) \in \mathbb{D}$ means "a needs b." $(a, b) \in \mathbb{D}$ is defined as an edge $(a, b) \in E$. In the following we provide a method to construct the dependency graph using routing information, the information flow, and the abstracted service definition discussed in section IV-B. For each service, the graph is generated by tracing service messages from the hosts to the users. For example, in an AMI, a message originates at the head-end, is sent to a meter M_l through a WAN connection to the collector (CR_k), and then is routed using a wireless mesh network. The message is verified through use of the key and the cipher $D_{k_l}(m)$ and is written to the C12.19 table entry. Thus, a service in an AMI depends on the state of the wireless channel that transmitted the message, the set of meters that relayed the message, the different components in the originating meter, and the state of the head-end.

We start with the set of all routes $\mathcal{R}$ in the system, those are the paths from each meter back to the head-end. A meter M_i could connect directly or through a set of meters (using a wireless network) to the collector CR_j, we represent those paths using $M_i \rightsquigarrow CR_j$. The collectors connect using a WAN or direct link to the head end. The total set of paths is $M_i \rightsquigarrow CR_j \xrightarrow{WAN} Headend$. We create the routing graph, which represents the backbone of our dependency graph. Then we attach our services and components to the respective hosts. The last step is to connect the dependent services and components. Each meter node contains the services in $\mathcal{S}$ and the dependent C12.22/C12.19 entities we identified.

Each node in the graph is tagged with a vector $V[C, I, A] \in [0, 1]^3$ that represents the current security level of the node in terms of confidentiality, integrity, and availability, with 0 being the least secure state and 1 being the most secure state. The index is interpreted as the current link capacity in the case of availability. For example, if 50% of traffic is dropped through a link, then $V_{link}[A] = 0.5$. However, confidentiality and integrity are binary, with 1 for the secure state and 0 representing the insecure state.

D. Weight Matrix

Each edge in G is tagged with a function, $\mathbb{F} : [0, 1]^3 \rightarrow [0, 1]^3$. The function is a mapping between the security properties (CIA) of a component and its dependent. That is, $\mathbb{F}$ defines the effects of a compromise of any of the security properties of a component on its dependent node in G. We implement the function $\mathbb{F}$ as a 3×3 weight dependency matrix $\mathcal{W}$.

1) Data Nodes: If the CIA of stored data is affected, then the respective security property of the dependent component is lost. For example, if meter-reading data D_{mr} is not available, then the billing service S_{mr} for the meter is not available. The weight matrix is a diagonal matrix with 1's on the diagonal and 0's elsewhere.

2) Cipher Nodes: The loss of confidentiality of a key leads to loss of confidentiality of the message, and loss of integrity. It does not, however, lead to loss of availability for the dependents. If the integrity of a key is compromised, then the service is using a mismatching key for the security mechanisms. Thus, availability and confidentiality are lost for the end services. If A is compromised, then either C is compromised because data are sent unencrypted, or A is compromised because the data are not being pushed through.

3) Communication Nodes: The security state of the communication link has a direct effect on the messages passing through it. That is, if the link loses availability, the service loses availability; this also holds for confidentiality and integrity. The weight matrix is thus a simple diagonal matrix.

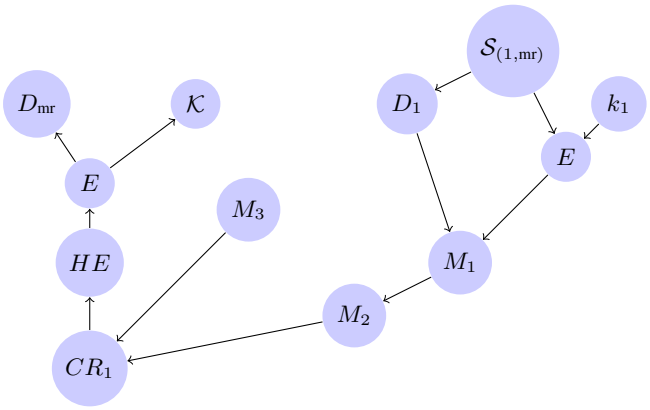

Fig. 2. Sample dependency graph of an AMI for meter-reading service $\mathcal{S}_{mr}$

E. Example

We want to generate the dependency graph of an AMI deployment. Our AMI has 3 smart meters $\{m_1, m_2, m_3\}$, a collector CR_1, and the head-end HE. The meters have the following topology: $M_1 \rightarrow M_2$, $M_2 \rightarrow CR_1$, $M_3 \rightarrow CR_1$ and $CR_1 \xrightarrow{WAN} HE$. We want to study the meter reading service in this example specified $\mathcal{S}_{mr}$. We only picked on service to make the example visually simple. For this service each meter has a data component $D_{(\text{i,mr})}$, a key component k_i to encrypt the readings, and a cipher E that uses the key. The head-end has a copy of the key components $\{k_1, k_2, k_3\}$, a data component to store the readings $D_{(\text{he,mr})}$ for the meter reading service, and a cipher component E to process the messages. The collector is merely a routing device that does not alter the content of the messages between the meters and the head-end. We generated the dependency for our AMI deployment shown in figure 2. Figure 2 shows a sample dependency graph in an AMI. In this example, the meter-reading service $\mathcal{S}_{(1,\text{mr})}$ of meter M_1 depends directly on the data component D_1 and the cipher E. The existence of the previous components depends on the meter M_1. The communication depends on the path between the M_1 and the head-end (HE) that includes meter M_2 and the collection relay CR_1.

V. Security State Computation

The cost model uses the dependency graph to compute the consequences of a response on system dynamics. The model computes a vector $V_{S_i}[CIA]$, which is the security vector of all services in an AMI. We begin by initializing the dependency graph with the intrinsic security status of the

Require: $EventSet \leftarrow Logs \cup Alerts$
Require: $G = (P, E)$
1: **for all** $event$ in $EventSet$ **do**
2: $V_{intrinsic,ID}[CIA] \leftarrow Result(event)$
3: **end for**

Fig. 3. Initialization algorithm

Require: $G = (P, E)$
Require: $R \leftarrow$ Response Action
1: **for all** S_i in $ServiceSet(P)$ **do**
2: $DependencyPath \leftarrow DependencySet(S_i)$
3: **for all** (D', E, D) in $DependencyPath$ **do**
4: $V_D = \min(\frac{1}{3}\mathcal{W}[E] \cdot V_{D'}, V_{intrinsic,D})$
5: **end for**
6: $V_{S_i}[\cdot] \leftarrow \min_{D \in DependencySet(S_i)}\{V_D[\cdot]\}$
7: **end for**
8: **for all** S in $ServiceSet$ **do**
9: $V[CIA] \leftarrow \frac{1}{N}\sum_i V_{S_i}[CIA]$
10: **end for**

Fig. 4. Update algorithm

graph. The intrinsic state is due to the previous state and new alerts and events that take place in the AMI. The events could be, for example a disconnected meter or a compromised key. Figure 3 shows the initialization algorithm. This algorithm requires the set of alerts and logs collected in an AMI. The alerts are mapped to the dependency graph through updating of the intrinsic security vector of all nodes. The function $Result(.)$, in line 2, changes $V_{intrinsic,ID}[CIA]$ based on the event. The result can be implemented as a look-up table for possible alerts. For example, if meter m_i lost its connections then $V_{intrinsic,m_i}[A] = 0$. The next step is to inject a response and evaluate the effect of the response on services. That is described in the update algorithm, Figure 4. First, we inject the response R, much as in to the $Result(.)$ function in the initialization algorithm. For example, if an action is a block, then we have to change the availability of the affected parties. The second step is to propagate the new vector values and compute the per-service security indices. We loop over the services in the graph; $ServiceSet(.)$ returns all the vertices that are services. We start with a service S_i and find the end-to-end dependencies of the service, going from the meter to which the service belongs to the head-end. We then loop over all sets of (D', E, D); this is a single hop in the graph. V_D is the security state due to the dependency. We update V_D by propagating the security state in its predecessor, line 4, where $\mathcal{W}$ is the weight matrix in section IV-D. We compute the $V[CIA]$ for S_i as the minimum security state along the path, line 6. If a dependent component of a service S_i loses confidentiality or integrity, then the service loses that property. The same argument applies to availability, because the link with the smallest availability acts as the bottleneck in the path. Finally, we compute the total $V[CIA]$ for each service in the AMI as the fraction of meters that were affected in line 9, where N is the number of meters in the AMI.

VI. CIA Conversion

The third step of the dependency analysis is to convert the security vector $V[CIA]$ into the financial cost incurred by a utility as described in section III. The cost breakdown for loss of security in AMI services is highlighted in Table II. The conversion method accounts for the possible consequences of the responses by modeling the effects of an attack on the power distribution system without explicitly enumerating the possible attack steps. For example, the table shows that the cost of loss of integrity and availability of pricing information and remote commands is due to outages caused by a malicious increase in power demand. The other factors we are considering are energy theft (electricity market) and SLAs that penalize degradation of service.

TABLE II
Attack/Response Action Cost Breakdown

	Integrity	**Availability**	**Confidentiality**
RT pricing	Outages		No Cost
Remote Cmnds.	Outages	SLA	Privacy Cost
Meter Readings	Electricity Market	SLA	

A. Meter-Reading Compromise

Compromising the integrity of meter-reading data could result in receipt of inaccurate metering data by a utility. The meter data are used to bill the customer for power usage; thus, any inaccuracies mean that either the customer is being overcharged, or the utility is losing revenue. Energy theft occurs when meter data reflect a value less than the actual one, causing revenue loss for the utility. The cost for the utility is shown in equation 2.

$$C_{mr} = \Delta T \times \text{estimatedUsageDiff} \times \$/\text{kWh} \times (1 - V_{S_{mr}}[I]) \quad (2)$$

ΔT is the duration of the period in which the response is enabled, and $\$/\text{kWh}$ is the cost of power during that period per kWh per unit of time. *EstimatedUsageDiff*, computed in equation 3, estimates the value of the power taken during the theft.

$$\textit{estimatedUsageDiff} = \gamma \times maxUsage \quad (3)$$

$\gamma \in [0, 1]$ is an index that models the attacker's aggressiveness; more usage is to be expected from a powerful attacker. Other methods could be used to estimate the amount of energy theft possible; those methods include using probabilistic attack models to estimate energy theft based on the sophistication of the attacker, the defenses deployed, and the success of the attack [18], [19]. However, the accuracy of those models is debatable; they require knowledge about the attacks and their prevalence, that is largely unknown to the system administrator. Moreover, such techniques are computationally intensive, and would make online use of the cost model infeasible. We believe that the linear model for the effect of the attack can help system administrator determine the cost of a response effectively.

If meter readings become unavailable, then the utility would miss a ΔT amount of readings. Such a compromise would not

lead to a cost for either the utility or the customer, since the data are already stored within the meter and can be transmitted after the AMI has been restored to a secure state. However, an SLA penalty would be incurred because of the unavailability of the service.

B. Real-time Pricing Information Compromise

Pricing information in an AMI is used for real-time billing. The price is sent to the meter, and the customer pays for the energy used based on that price. If the pricing information is unavailable, the customer uses electricity in the "blind". This means that he or she cannot adjust usage if the price increases. So we assume that it is only fair for the utility to switch back to the flat rate pricing scheme, by which customers pay a constant rate per kWh. The utility loses revenue if the flat rate is less than the current price of energy, because the utility is covering the difference in cost. If the flat rate is more than the actual rate, then the customer is overpaying for energy. The cost conversion algorithm estimates the usage of customers using historic data within a region to compute the revenue loss for the utility.

$$C_{pi} = \Delta T \times \text{historicUsage} \times \Delta\$/\text{kWh} \tag{4}$$

If the integrity of pricing information is breached, then we risk a sudden increase in demand as the low prices drive demand. More details on the cost of power demand increases are presented in the next section. Pricing information is public, so there are no concerns about privacy compromise.

C. Remote Commands Compromise

Service commands remotely control customer service and appliances. That enables demand-response, which controls prices by varying demand during critical times. The loss of integrity of remote commands allows an adversary to arbitrarily increase power demand in the power distribution system. The demand increase has two cost repercussions for a utility if the new demand is a significant fraction of the total demand on the transmission grid. The first direct cost is the fact that the increase of demand will require the utility to purchase generation power at a high cost. The indirect cost is incurred by local outages due to severe and persistent demand hikes. Outages are caused when components of the distribution grid (such as transformers, fuses, and capacitor banks) are overloaded for long periods of time. It is important to note that the distribution grid is designed to handle the maximum possible demand for a short period of time. Engineers measure the peak demand over 15-minute periods for customers. The demand factor, the ratio of the maximum demand over total load, is around 0.2. Then designers compute the maximum 15-minute peak demand for the set of customers to be handled by the electric equipment. Finally, the rating on the equipment is selected to handle the compute maximum peak demand for a short period of time. Thus, if the demand is increased above the expected value for more than 15 minutes, equipment will get damaged [20]. Modeling of the distribution grid is used to predict the costs in the case of increased demand. The value of the increased demand is found using equation 5.

$$Demand = \gamma \times maxPower \times V[I] \tag{5}$$

γ is the same parameter as in equation 3; it models the effect of the attack. It can be set to depend on the criticality of the location or on the level of pessimism the utility is exercising. Outages require that trucks be rolled out to respond and repair damage. They also cause losses for customers; the cost for customers is highlighted in [21] based on an empirical study.

D. Cost of Privacy

Data within an AMI contain private information about the appliances used by the customer and the time a customer uses those appliances. This information can be used to assess the wealth of a person, for advertising purposes by ad agencies, or for criminal activity. Protecting the confidentiality of this information means that the data should not be accessed by unauthorized entities during transit in the AMI network or while stored at the head-end in the utility. There are two types of data that could lead to load identification: the actual power usage information (meter readings), and the demand-response mechanisms that control specific appliances. Several papers propose methods for identification of loads that use usage data [22], [23], [24] and through demand-response systems [25]. Other researchers were able to identify what was being viewed on televisions [26]. Researchers have proposed methods to preserve the privacy of AMI users through special protocols or by installing home batteries. The protocols use homomorphic encryption to transmit and store data without revealing the content, the meter readings, or commands. On the other hand, the battery, when installed, will serve as a buffer, so that it stores power that is used by appliances; that way, power usage will be constant [27]. Although it is not easy to put a price on privacy, there are several methods for doing so. New laws related to smart girds are getting passed: for example, California's smart grid laws require utilities to protect usage information. So when privacy is lost utilities will be vulnerable to lawsuits, which typically result in major financial losses. Moreover, a penalty could be enforced, similar to NERC CIP violation penalties. Further, recurring privacy compromises might lead customers to lose confidence and change service providers, thus leading to long-term cost to the utility.

VII. Implementation and Study Case

We now provide lower-level details about the implementation. We start with the details of the dependency graph implementation. We then specify the method for injecting the responses and computing the security vector $V_i[CIA]$. Finally, we convert the security vector to the financial cost incurred by the utility due to the distribution grid.

A. Dependency Graph

In a practical deployment setting, the dependency graph is generated from a grid topology and meter network information. The meter network information is the list of neighbors reachable by each meter. We wrote a script that takes

geographic information system (GIS) data as the sole input and automatically infers an AMI deployment configuration. To test the script implementation, we obtained GIS data from the publicly available parcel tax data of a small town (with a population of 7,282 inhabitants in 2011). We assumed a single meter per parcel, and we located a number of cell relays to cover the GIS region. We generated the topology of the deployment. Then the cell relays (CRs) were added to the network to cover the deployed meters, allowing 3,000 meters per relay. We formed the shortest routes between the meters and the closest CR in the area. Finally, the dependency graph was generated using the algorithm specified in section IV. All security vectors were initialized to the secure state of the network, that is, $V_i = [1, 1, 1]$, where i represents indices of nodes in the system.

B. Execution Model

The proposed cost model starts by updating the security state of the system by updating the dependency graph. The update changes the security vectors for affected nodes in the system, and then the new values are propagated throughout the dependency graph. A response $\mathcal{R}$ has to be injected into the dependency graph in order to be evaluated; section VII-C explains the process of injecting a response. After the response has been injected, the update algorithm is run again in order to update the security vectors in the graph. Finally, we have to convert the new values into the financial values using the process described in section VI. During the operation of the system the topology of the AMI will change due to the addition/removal of new meters and collectors. In case of a topology change, we need to update the dependency graph to reflect the new changes. We add or remove the affected nodes and re-run the shortest path algorithm in the previous section to update the graph. The state of the new nodes and links are initialized to the secure state. We recompute the security state of the system using the algorithms in section V. This process might seem costly because we have to recompute links in the dependency graph. However, adding a single meter affects only a small subset of meters that are within the same geographic vicinity. Implementation details on the financial conversion process when the distribution grid is affected are discussed in section VII-D.

C. Response Injection

Each action induces changes in the dependency graph; the security vector $V[CIA]$ is updated based on the type of action and the expected effects on the system.

The taxonomy of response actions described in [28] has two kinds of actions, *learning* actions and *modifying* actions. Learning actions do not change the topology of the network; instead, they add traffic to the network in order to gain more information about an event. When a learning action is injected, the availability of a meter or a link is affected because of the traffic needed to get the data. On the other hand, modifying actions can either block or limit an attack, or recover from an insecure state. A limiting response action that blocks a set of meters results in voiding of the availability of the meters, so $V[A] = 0$. Reboot actions cause unavailability for the duration of the reboot process. Recovery actions generally reset the security vector V to $[1, 1, 1]$. However, some actions, such as flashing of a new firmware or verification of routes and meters, temporarily cause reduction in availability. The last step in the communication-side implementation is to update the security vectors. The update algorithm computes the new security vectors in the network. We then convert the CIA indices to the financial cost based on the expected consequences of the action and the actual cost of implementing the action.

D. Distribution Grid Simulation

Parts of the conversion process require prediction of outage frequency and duration. We assume that an outage consists of prolonged overloading of parts of the distribution grid. Localized outages require the utility to dispatch crews to repair and restore service, thus inducing cost to the utility. GridLAB-D [29] is used to simulate the grid while the power demand of residential loads is being increased. GridLAB-D is an open-source, agent-based distribution grid simulator developed and maintained by PNNL. We feed the new security vectors to the GridLAB-D, which determines the amount of extra demand for power that might occur because of injected response actions. That models the notion of consequences enabled by the action after an attacker executes a particular attack.

GridLAB-D's reliability module models faults in the grid that result from short-circuit faults, such as Single-Line-Ground and Line-Line faults. GridLAB-D allows the user to set the target, timings, and frequency of a fault. GridLAB-D does not model faults or consequences due to overloaded components in the grid; however, it notes the overload. For our purposes, we run the simulation twice; first, we increase demand and record the lines with power flow over the rated level. Then, we inject the faults to model possible sagging of lines, destroyed fuses, or damaged transformers. GridLAB-D simulates the faults and reports outage frequency and sizes and some relevant reliability metrics. The cost for a utility to fix the fault is then estimated using an assumption about the cost to repair an outage, SAIDI $\times$ $repairCost(t)$, where SAIDI stands for the system average interruption duration index, which shows the average duration of a sustained interruption for a customer during the reporting period.

E. Performance

The cost model of response actions is part of a response engine decision algorithm. The response engine works in an online fashion to decide on a suitable response during a security incident. The cost model should be efficient in terms of memory usage and runtime in order to get used in an online response system.

1) Dependency Graph: We implemented the dependency graph using an adjacency list to store the state of the different nodes in the system. Each meter node has 8 components, including 3 services, 4 data objects, and 1 cryptographic key, and is tagged with a security vector that holds three floating values (for confidentiality, integrity, and availability). As a result, the memory footprint of the graph is $2N \times (3 \text{ services} + 4 \text{ data} +$

1 key + 1 meter) × (20 bytes). A medium size utility with $1,000,000$ customers will require of around 1 GB of storage. Moreover, we can estimate the number of relationships in the dependency graph. In an AMI, a meter is 4 hops away, on average, from a CR. The number of relationships added by a meter, accounting for 3 services, is on average 20. The number is due to dependencies between the services in the meter and the meter resources, the number of hops between the meter and a CR, and the connection between the CR and HE. Thus we have 20 million relations for a 1 million meter deployment. We implemented a small version of the dependency graph using Neo4j. Neo4j is an open-source graph database. Neo4j is capable of supporting around 34 billion relationships and nodes [30]. Our implementation specified the queries that find the service sets and dependency path. Moreover, the algorithm in Figure 4 has a worst complexity of $O(|P||E|)$; however, the update algorithm is optimized for the AMI study case. The algorithm updates the security vector per service by traversing the path from the service to the head-end. Since the average path length in an AMI is relatively small, the algorithm has an average-complexity running time, $O(|P|)$.

2) Grid Simulation: Our current GridLAB-D is running on a single core; the authors in [31] show that a 200,000-bus node requires 67.64 sec to simulate. They reached a 62x speedup using an FPGA implementation.

VIII. Case Study

In this section, we show how the cost model works in a case study. First, we investigate the level of power consumption needed to drive the distribution grid to a critical state. Then, we use the parameters computed in the first step to study the costs of responding to an attack scenario that compromises the demand-response mechanism in an AMI.

A. Threat Model

The adversary in this case study wants to overload demand in the distribution system and cause outages by damaging equipment (such as lines and transformers). We assume that the adversary is eavesdropping on and storing AMI traffic. We assume that the adversary has the capability to inject and replay the stored packets. For example, if the adversary replays a remote control message that switches an appliance ON, the appliance will be switched on. We assume that the attacker is exploiting a vulnerability in the meters that allows packets to be replayed. Please note that we do not assume that the attacker have compromised the encryption keys; they just collect the packets and replay them without modifying them.

The resources of the adversary in this scenario are related to how many physical sniffing devices can they deploy. If they are amateurs with limited resources. can they cover a small area of the AMI and thus have a minimal effect on the distribution system. On the other hand, if the adversaries are resourceful, they can cover a large area of the AMI and increase the demand significantly, causing outages and equipment damage. We model the resourcefulness of the adversary and the intensity of the threat using the parameter $\gamma \in [0, 1]$.

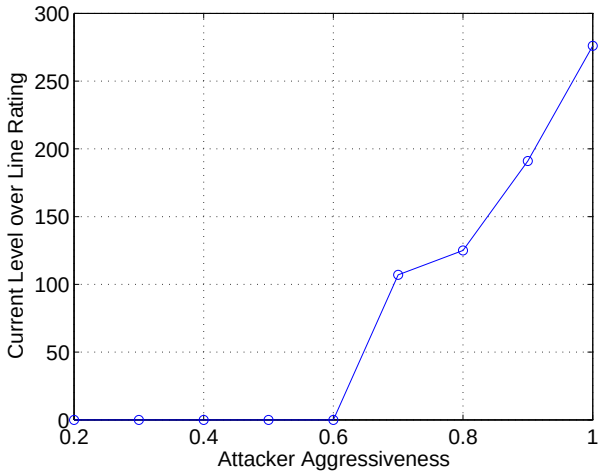

Fig. 5. Current level with varying power demand

B. Demand Increase

We simulated the IEEE 37-node test feeder [32], a standard distribution grid topology that has 37 nodes used for testing numerical algorithms, and varied the power consumption of loads in the topology. The simulation verified the power supply could be disrupted by an increase in demand. Moreover, the simulation found the minimum value of γ to be used in the cost model. We assumed that all loads in the topology were residential loads. Typical residential loads reflect a set of common high-power-demanding household appliances that are capable of generating 40 kW of demand. Table III shows the ratings of some common high-power-demanding appliances. Assume that demand-response systems typically control heavy loads in order to curtail demand during high-demand periods. The maximum possible power injection due to the 37 residential loads is $P_{max} = 1.5$ MW.

TABLE III
Typical Power Consumption [33]

Appliance	Oven	AC	Water Boiler	Dryer
Rating	12 kW	9.2 kW	4.4 kW	3 kW

We simulated the surge of demand that would result from increasing residential demand by $\gamma \times 40$ kW. Figure 5 shows the percentage of current level over the current rating in the main feeder as we increase the attacker aggressiveness γ. When the attacker aggressiveness increases, more meters are controlled, and more appliances can be controlled. By controlling more appliances, the attacker can increase power demand in the distribution system. The simulation confirms that increasing demand causes the current levels to increase beyond the rated levels of the feeders. The results show that it is possible to drive the distribution grid to a critical state with a minimum value of $\gamma = 0.6$.

C. AMI Scenario

We wanted to showcase our cost model for response actions using an AMI under attack. We defined an AMI deployment

with the assumptions given in section IV. The AMI serves 37 customers, and we placed the meters according to the method suggested in section VII. We considered a specific attack scenario based on the threat model, and defined two different response actions that would block the attack and recover to a secure state. The cost model computed the cost of the responses using the parameters shown in Table IV. Historic usage profiles and price forecasts were used to estimate the flat and peak demand rates, and SLAs were in place to penalize for loss of availability of some AMI services. Finally, the customers were connected to the distribution grid through use of the IEEE 37-node test feeder topology.

In our scenario, the adversary exploits a vulnerability that allows them to replay packets. The packets are assumed to be remote control packets for residential appliances. The stored packets are encrypted, so the attacker cannot modify them, they only collect and replay them. We assume that the exploit cannot be detected by the meters, that is the meter cannot detect the replayed packets.

TABLE IV
CASE STUDY PARAMETERS

Parameter	Value	Parameter	Value
Node loc.	GIS	repair cost	\$100/hr
Total nodes	36	T_{patch}	1 day
P_{max}	1.5 MW	ΔT	5 hrs
γ	0.6	Power surge	3 kWh
T_{trans}	18.5 sec	Flat rate	\$4
SLA DR	\$3,000/hr	Peak power cost	\$4.50
SLA MR	\$500/hr	Failure rate	2×10^{-5}

1) Response Actions: We consider two possible response actions. The first response, $\mathcal{R}1$, is to re-key the meters. After the meters are re-keyed, the meter will not be able to decrypt the packets stored by the adversary, as they were encrypted with an older key. The meters are frequently re-keyed at a rate of ΔT, until a patched firmware is sent to all meters. Meters are re-keyed because the attacker will store a new set of messages, encrypted with the new key, to restore the attack. Then, after T_{patch} time, the meters are patched, and the re-keying stops. The second response, $\mathcal{R}2$, re-keys all the meters once during the response strategy, and then blocks the remote command service. Then, after T_{patch}, the meters are patched, and service is restored. The first response does not completely block the remote control service, but instead allows the remote service to run while it changes the AMI keys periodically. The response also risks the possibility of replaying some demand-response messages between the re-keying periods. The second response does not involve the risk of replaying demand-response messages, but it completely blocks the remote command service until a firmware fix is available.

2) Response Cost: The cost of $\mathcal{R}1$ is shown in equation 6. The re-keying process generates new meter keys at the head-end, and then sends them to the meters. The periodic key change causes short temporary unavailability of all services during the roll-out time. After the meters have been re-keyed, the attacker can reinitiate the attack and thus cause an outage in the distribution grid. The first part of the equation ($SLA(All)$) is the SLA penalty due to loss of availability of the AMI service. The second part of the equation reflects the cost of loss of availability of real-time pricing information due to the re-keying process, shown in equation 4. The last part of the cost is due to possible outages, as explained in section VII-D.

$$\begin{aligned} C(\mathcal{R}1) = {} & SAIDI(\gamma) \times \Delta T \times repairCost \\ & + [V_{dr}[I](N \times 3kWh \times 0.5\$/kWh + SLA(All)) \times \\ & \times \frac{T_{patch}}{\Delta T} \times T_{trans}] \end{aligned} \tag{6}$$

where T_{trans} is the time needed to roll out the keys during the re-keying process. The ratio $T_{patch}/\Delta T$ is the number of times meters are re-keyed. The cost of $\mathcal{R}2$ is shown in equation 7. The cost reflects the SLA penalty due to the unavailability of the remote control service of the AMI.

$$C(\mathcal{R}2) = SLA(\mathcal{S}_{rc}) \times T_{patch} \tag{7}$$

D. Numerical Analysis

In this section, we compute the actual costs of the response actions $\mathcal{R}1$ and $\mathcal{R}2$ using the parameters in Table IV.

In those parameters, the SLA penalty for blocking the remote service is high, which is expected because the remote service provides demand-response, which offers major cost relief to the utility. The parameter *repairCost* is also relatively high, because it includes labor hours and rolling of trucks. The time needed to roll out the keys during the re-keying process is estimated as $T_{trans} = N \times 500$ ms.

We first computed the SAIDI due to $\mathcal{R}1$. Since $\gamma = 0.6$, which represents the minimum value of demand when the distribution grid is in a critical state, we increase the failure rate to find the lowest failure rate at which the distribution grid begins to show sustainable outages. Our study showed that at a failure rate of 2×10^{-5}, the SAIDI was about 6.

The cost of $\mathcal{R}1$ is $\$3,000$, while the computed cost of $\mathcal{R}2$ is $\$72,000$. Even though $\mathcal{R}1$ has SLA penalties for all services, the penalties are applied for such a short period of time that the actual cost diminishes. If ΔT were longer, then the cost to repair outages could have been higher than the SLA penalty for the remote command service. As a result, based on the value of the parameters, the system recommends that $\mathcal{R}1$ be applied if there is an attack against the demand-response system.

IX. CONCLUSION

This paper presented a comprehensive approach for cost modeling of response actions. The approach accounts for cost due to the operation of the action as well as the cost due to the action's consequences for the system. We proposed an extended dependency graph to model any system, and used it for AMI. We then defined methods to convert the response's impact on an AMI to a financial cost to the utility. The methods mainly account for losses due to changes in the physical system, in the form of outages and price fluctuations. Finally, we proposed using a simulator, GridLAB-D, to predict outages in the distribution grid due to power demand increases. The

main contribution of this work is that it offers a way to model response actions that requires a minimum of subjective input from the utility, as it uses the costs due to the physical grid to judge the importance of services, instead of requiring an administrator to set such a critical parameter manually.

ACKNOWLEDGMENTS

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000097. Additionally, we would like to thank Jenny Applequist for her constructive revisions.

REFERENCES

[1] S. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "RRE: A game-theoretic intrusion response and recovery engine," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2009, pp. 439–448.

[2] S. Tanachaiwiwat, K. Hwang, and Y. Chen, "Adaptive intrusion response to minimize risk over multiple network attacks," *ACM Trans on Information and System Security*, vol. 19, pp. 1–30, 2002.

[3] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, and S. Dubus, "Risk-aware framework for activating and deactivating policy-based response," in *Proceedings of the 4th International Conference on Network and System Security (NSS)*, Sept. 2010, pp. 207–215.

[4] Y. Wu and S. Liu, "A cost-sensitive method for distributed intrusion response," in *Proceedings of the 12th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, April 2008, pp. 760–764.

[5] Z. Zhang, X. Lin, and P.-H. Ho, "Measuring intrusion impacts for rational response: A state-based approach," in *Proceedings of the Second International Conference on Communications and Networking in China (CHINACOM)*, Aug. 2007, pp. 317–321.

[6] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E. Spafford, "Adepts: Adaptive intrusion response using attack graphs in an e-commerce environment," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2005, pp. 508–517.

[7] W. Lee, W. Fan, M. Miller, S. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *Journal of Computer Security*, vol. 10, no. 1-2, pp. 5–22, 2002.

[8] Z. Zhang, P.-H. Ho, and L. He, "Measuring IDS-estimated attack impacts for rational incident response: A decision theoretic approach," *Computer Security*, vol. 28, no. 7, pp. 605–614, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404809000273

[9] C. Strasburg, N. Stakhanova, S. Basu, and J. Wong, "The methodology for evaluating response cost for intrusion response systems," Iowa State University, Tech. Rep. 08-12, 2008.

[10] Y. Luo, F. Szidarovszky, Y. Al-Nashif, and S. Hariri, "A game theory based risk and impact analysis method for intrusion defense systems," in *Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, May 2009, pp. 975–982.

[11] N. Anuar, S. Furnell, M. Papadaki, and N. Clarke, "A risk index model for security incident prioritisation," in *Proceedings of the 9th Australian Information Security Management Conference (AISM)*, 2011, pp. 25–39.

[12] T. Toth and C. Kruegel, "Evaluating the impact of automated intrusion response mechanisms," in *Proceedings of the 18th Annual Computer Security Applications Conference*, 2002, pp. 301–310.

[13] M. Jahnke, C. Thul, and P. Martini, "Graph based metrics for intrusion response measures in computer networks," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN)*, Oct. 2007, pp. 1035–1042.

[14] N. Kheir, H. Debar, N. Cuppens-Boulahia, F. Cuppens, and J. Viinikka, "Cost evaluation for intrusion response using dependency graphs," in *Proceedings of the International Conference on Network and Service Security (N2S)*, June 2009, pp. 1–6.

[15] *C12.22 Protocol Specification For Interfacing to Data Communication Networks*, American National Standard Institute, January 2009. [Online]. Available: http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+C12.22-2008

[16] R. Berthier and W. H. Sanders, "Specification-based intrusion detection for advanced metering infrastructures," in *Proceedings of the 17th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2011)*, Dec 2011, pp. 184–193.

[17] *RFC 6142 C12.22 ANSI C12.22, IEEE 1703, and MC12.22 Transport Over IP*, Internet Engineering Task Force, 2011. [Online]. Available: http://tools.ietf.org/html/rfc6142

[18] T. Chen, J. Sanchez-Aarnoutse, and J. Buford, "Petri net modeling of cyber-physical attacks on smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 741–749, Dec. 2011.

[19] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke, "Model-based security metrics using ADversary VIew Security Evaluation (ADVISE)," in *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST)*, Sept. 2011, pp. 191–200.

[20] W. H. Kerting, *Distribution system modeling and analysis*. Florida, USA: CRC Press, 2002.

[21] A. Chowdhury, T. Mielnik, L. Lawion, M. Sullivan, and A. Katz, "Reliability worth assessment in electric power delivery systems," in *IEEE Power Engineering Society General Meeting 2004*, June 2004, pp. 654–660.

[22] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, Dec. 1992.

[23] H.-H. Chang and C.-L. Lin, "A new method for load identification of nonintrusive energy management system in smart homes," in *Proceedings of the 7th IEEE International conference on e-Business Engineering (ICEBE 2010)*, Nov. 2010, pp. 351–357.

[24] M. Pipattanasomporn, M. Kuzlu, S. Rahman, and Y. Teklu, "Load profiles of selected major household appliances and their demand response opportunities," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 742–750, March 2014.

[25] M. Lisovich, D. Mulligan, and S. Wicker, "Inferring personal information from demand-response systems," *IEEE Security & Privacy*, vol. 8, no. 1, pp. 11–20, Jan. 2010.

[26] U. Greveler, P. Glosekotter, B. Justusy, and D. Loehry, "Multimedia content identification through smart meter power usage profiles," *Computers, Privacy and Data Protection*, 2012.

[27] D. Varodayan and A. Khisti, "Smart meter privacy using a rechargeable battery: Minimizing the rate of information leakage," in *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 1932–1935.

[28] A. Fawaz, R. Berthier, and W. H. Sanders, "Cost modeling of response actions for automated response and recovery in AMI," in *Proceedings of the 3rd IEEE International Conference on Smart Grid Communications 2012 (SmartGridComm 2012)*. IEEE, Nov. 2012, pp. 348–353.

[29] (2014). [Online]. Available: http://www.gridlabd.org/

[30] (2014). [Online]. Available: http://www.neo4j.org/

[31] J.-H. Byun, A. Ravindran, A. Mukherjee, B. Joshi, and D. Chassin, "Accelerating the Gauss-Seidel power flow solver on a high performance reconfigurable computer," in *Proceedings of the 17th IEEE Symposium on Field Programmable Custom Computing Machines (FCCM'09)*, april 2009, pp. 227 –230.

[32] W. Kersting, "Radial distribution test feeders," in *Proceedings of the IEEE Power Engineering Society Winter Meeting, 2001.*, vol. 2, 2001, pp. 908–912.

[33] D. of Energy. (2012) Estimating appliance and home electronic energy use. [Online]. Available: http://energy.gov/energysaver/articles/estimating-appliance-and-home-electronic-energy-use

Ahmed Fawaz is a Ph.D candidate at the Coordinated Science Laboratory (CSL), University of Illinois at Urbana Champaign. He received his B.E. in Electrical and Computer Engineering in 2011 from the American University of Beirut. Currently, he is working on trust issues in monitoring data during cyber incidents and intrusion resilience in the future smart grid through automated response and recovery using control theory, game theory, hybrid systems and machine learning.

Robin Berthier is a Research Scientist in the Information Trust Institute (ITI) at the University of Illinois at Urbana-Champaign, and a co-founder of Network Perception. He received his Ph.D. in the Reliability Engineering program at the University of Maryland in 2009 under the supervision of Michel Cukier. His research projects include Amilyzer, a specification-based intrusion detection system for advanced metering infrastructures, and NP-View, a firewall compliance checking tool for critical infrastructures. NP-View is the commercial release of the NetAPT tool that has been transferred to Network Perception, a startup incubated at the University of Illinois research park.

William H. Sanders William H. Sanders is a Donald Biggar Willett Professor of Engineering and the Head of the Department of Electrical and Computer Engineering (www.ece.illinois.edu) at the University of Illinois at Urbana-Champaign (illinois.edu). He is a professor in the Department of Electrical and Computer Engineering and Affiliate Professor in the Department of Computer Science. He is a Fellow of the IEEE, the ACM, and the AAAS; a past Chair of the IEEE Technical Committee on Fault-Tolerant Computing; and past Vice-Chair of the IFIP Working Group 10.4 on Dependable Computing. He was the founding Director of the Information Trust Institute (www.iti.illinois.edu) at Illinois (2004-2011), and served as Director of the Coordinated Science Laboratory (www.csl.illinois.edu) at Illinois from 2010 to 2014.

Dr. Sanders's research interests include secure and dependable computing and security and dependability metrics and evaluation, with a focus on critical infrastructures. He has published more than 200 technical papers in those areas. He is currently the Director and PI of the DOE/DHS Trustworthy Cyber Infrastructure for the Power Grid (TCIPG) Center (www.tcipg.org), which is at the forefront of national efforts to make the U.S. power grid smart and resilient.

He is also co-developer of three tools for assessing computer-based systems: METASAN, UltraSAN, and Mbius. Mbius and UltraSAN have been distributed widely to industry and academia; more than 1,400 licenses for the tools have been issued to universities, companies, and NASA for evaluating the performance, dependability, and security of a variety of systems. He is also a co-developer of the Loki distributed system fault injector, the AQuA/ITUA middlewares for providing dependability/security to distributed and networked applications, and the NetAPT (Network Access Policy Tool) for assessing the security of networked systems.