A Noninvasive Threat Analyzer for Advanced Metering Infrastructure in Smart Grid

Mohammad Ashiqur Rahman, *Member, IEEE,* Ehab Al-Shaer, *Member, IEEE,*
and Padmalochan Bera, *Member, IEEE*

***Abstract*—Advanced Metering Infrastructure (AMI) is the core component in a smart grid that exhibits a highly complex network configuration. AMI comprises heterogeneous cyber-physical components, which are interconnected through different communication media, protocols, and security measures. They are operated using different data delivery modes and security policies. The inherent complexity and heterogeneity in AMI significantly increases the potential of security threats due to misconfiguration or absence of defense, which may cause devastating damage to AMI. Therefore, there is a need for creating a formal model that can represent the global behavior of AMI configuration in order to verify the potential threats.**

In this paper, we present *SmartAnalyzer*, a security analysis tool, which offers manifold contributions: (i) formal modeling of AMI configuration that includes device configurations, topology, communication properties, interactions among the devices, data flows, and security properties; (ii) formal modeling of AMI invariants and user-driven constraints based on the interdependencies among AMI device configurations, security properties, and security control guidelines; (iii) verifying the AMI configuration's compliance with security constraints using a Satisfiability Modulo Theory (SMT) solver; (iv) reporting of potential security threats based on constraint violations, (v) analyzing the impact of potential threats on the system; and (vi) systematic diagnosing of SMT unsatisfiable traces and providing necessary remediation plans. The accuracy and scalability of the tool are evaluated on an AMI testbed and various synthetic test networks.

***Index Terms*—Advanced metering infrastructure, security analysis, formal verification.**

I. Introduction

Smart grids provide innovative and efficient energy management services, which offer operational reliability and value-added advantages to both customers and energy providers. Advanced metering infrastructure (AMI) in a smart grid provides two-way communication between smart meters and utility servers (headend systems) through intelligent collectors, which allows energy service providers as well as customers to monitor and control power consumption remotely [1], [2]. An AMI network also consists of different kinds of hosts, i.e., the customers' hosts and the energy provider's internal and external users' hosts. These devices communicate with one another using various communication media and protocols, and different security measures, such as firewall based access controls, IPSec based security tunnels, and IDS based payload inspection. Energy usage data is transferred from meter to collector and collector to headend following different modes of data delivery under the control of alternative organizational security policies. Cyber attacks on such networks due to inappropriate or weak configurations have the potential to cause critical damages including power outages and destructions of equipment [3], [4].

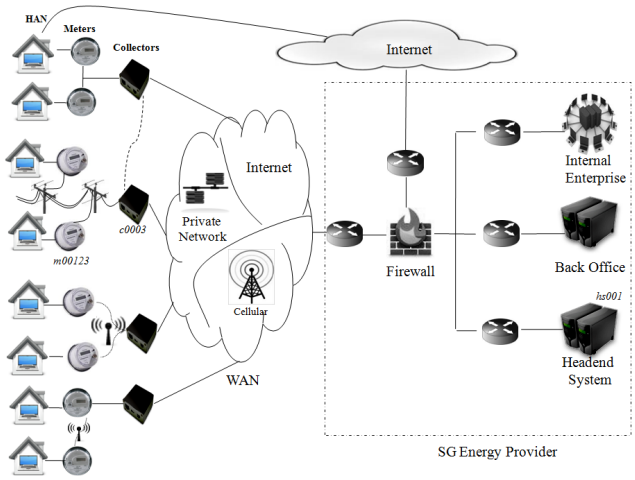

Fig. 1. A typical AMI smart grid network.

The potential misconfigurations and security threats in AMI increase significantly, due to the inherent complexity and interdependencies of device level configurations with various security parameters. There is a pressing need for automated security analysis to identify misconfigurations as well as security threats proactively in AMI. In this paper, we present an automated security analysis tool for AMI, named *SmartAnalyzer*, that creates a logic-based formal model of AMI behavior based on AMI device configurations and verifies the compliance of this model with system invariants and security requirements using constraint satisfaction checking. The tool generates a comprehensive threat report along with a potential remediation plan. We measured the performance of our tool by running it on an AMI testbed and various synthetic test networks. Here, we extend the SmartAnalyzer tool presented in our past work [5] to include a new set of security controls and analytic capabilities. The major extensions are as follows:

1) Our framework provides a formal logic based modeling of various new security constraints, in addition to the constraints specified in [5]. The constraints newly added to the framework are the following:
   - Authenticated communication constraint

Mohammad Ashiqur Rahman and Ehab Al-Shaer are with the Department of Software and Information Systems, University of North Carolina, Charlotte, NC 28223, USA. E-mails: {mrahman4, ealshaer}@uncc.com.

Padmalochan Bera is with Software Engineering and Security Research Group, Infosys Labs, Bangalore, India. Email: bera.padmalochan@gmail.com.

Manuscript received March 31, 2012; revised September 26, 2012.

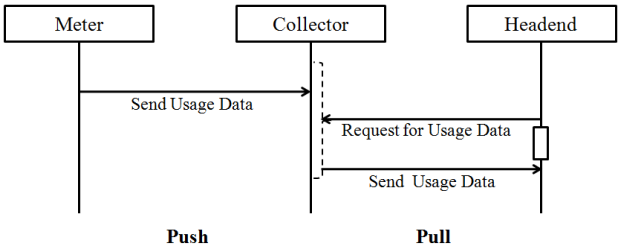

Fig. 2. Push-Pull mode of Data Delivery in Smart Grid

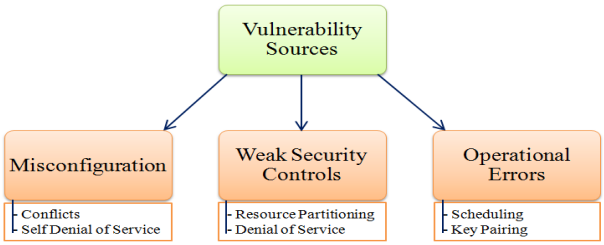

Fig. 3. Root sources of Smart Grid Security Threats

- Secure tunnel constraint
- Priority delivery constraint
- Fault-tolerant constraint
- Domain boundary protection constraint

2) We present a comprehensive diagnosis of misconfigurations or security weaknesses by analyzing constraint violation results (SMT traces) and generating necessary remediation plans. We show an example in order to explain the process of trace analysis.
3) We briefly show an approach for the qualitative impact analysis of a potential security threat by defining three impact factors: (i) the loss due to the threat occurrence, (ii) the easiness of the threat execution, and (iii) the cost for the remediation of the threat.

The rest of the paper is organized as follows. We discuss the background, challenges, and objective of our research in Section II. We describe the architecture of SmartAnalyzer in Section III. We present the formal modeling of AMI topology and the components in Section IV. We present the modeling of various invariant and user-driven security constraints in Section V. In Section VI we describe the investigation of constraint verification results in terms of diagnosis and impact analysis. We present the evaluation results of our tool in Section VII. We briefly discuss about the limitations, extensibility, and deployment of SmartAnalyzer in Section VIII. We describe the related works on AMI security in Section IX. We conclude the paper with the summary in Section X.

II. Background and Challenges

A. AMI Complexity

The general structure of an AMI network is shown in Fig. 1, which usually consists of millions of smart meters, thousands of intelligent data collectors, and one or more headend systems as the main components. A meter establishes a secure connection with a specific collector and reports energy usage data periodically. A collector forwards the data received from a group of meters to a headend over a secure connection. It also forwards control commands and patches from the headend to the meters. A meter may be connected to a collector directly or through another meter. The latter case occurs in a mesh network of meters, where intermediate meters relay the data to the collector. A large number of collectors are connected with a headend usually through a proprietary but third party network. There is one or more firewalls for restricting the access between AMI and the energy provider's network. There are two data delivery modes, which can be used between meter and collector, and between collector and headend: (i) *push-driven mode* (simply, push mode) in which a meter or a collector reports data periodically based on a pre-configured delivery schedule, and (ii) *pull-driven mode* (simply, pull mode) in which a meter or a collector reports data only upon receiving a request. In the pull mode, requests are usually sent periodically following a schedule. In practice, the push mode is used between meter and collector, while the pull mode is used between collector and headend (Fig. 2).

AMI networks are more complex than traditional networks mainly due to the following reasons. First, AMI is a hybrid network consisting of (1) heterogeneous devices (e.g., meters, collectors, firewalls, routers, IPSec gateways, etc.), (2) varieties of links (e.g., power lines, wired, and wireless), and (3) different protocols (e.g., LonTalk protocol [6] between meter and collector, and TCP/IP protocol between collector and headend). Second, an AMI network involves varieties of data stream types (such as power usage data, control commands, software patches), which exhibit different priorities and resource requirements. Third, unlike the policy-based Internet forwarding, data delivery in AMI is either time-driven or request-driven and it follows specific schedules. For the purpose of successful delivery of data, AMI must be configured carefully to synchronize the data delivery without overflowing the network or its devices. Moreover, an AMI network must be accessible from the utility network for different purposes like control and patch management. Energy users from Home Area Networks (HANs) can also access the AMI network via the Internet or smart meters.

B. Potential Threats on AMI

The inherent complexity of integrating multiple heterogeneous systems in AMI significantly increases the potential of security threats, which can cause massive and extremely devastating damage. The root causes of security threats on AMI have been shown in Fig. 3.There are two main causes of threats on AMI [7]. The first is the *misconfiguration* that might cause inconsistency, unreachability, broken secure tunnels, etc. It is well documented that configuration errors cause 50%-80% of vulnerabilities in cyber infrastructure [8]. The second is the *weak or absence of security controls* that can cause cascaded cyber attacks, such as scanning, denial of service (DoS), jamming, etc. Threats might come due to *operational errors* also. For example, if the data delivery is request-driven rather than time-driven, an operational error, such as requesting data from a large number of collectors at the same time may lead to cyber breakdown. We classify the potential threats on AMI into the following categories:

Reachability and Integrity Threats. To achieve successful data delivery, reachability must hold between the sender and the receiver. In addition, data should be delivered satisfying end-to-end integrity. Its violation not only can cause incorrect billing but may also launch malicious control commands towards AMI. Moreover, inconsistencies in authentication and encryption parameters of the communicating devices may cause service disruption. One important aspect of the reachability verification is to check the correctness of any IPSec tunnel existing in the path.

Availability Threats. Improper scheduling of data delivery between meters and collectors can lead to buffer overflow and data loss in the collector side. This can cause delay in data delivery, even data loss at the endpoints due to limited link bandwidth. For example, if UDP protocol is used between collector and headend, improper scheduling may allow a large number of nodes to transfer data to a headend, which can flood links on the path and consequently cause data loss. The use of TCP protocol (TCP based communication) may create traffic congestion, which in turn leads to delay in data reporting and data loss on the sender side (if data rate is higher than the delay). The main purpose of AMI is to deliver clients' power usage data to the provider's side. Hence, these resource availability threats can be very devastating.

Typical Cyber Threats. Typical cyber threats on AMI are endpoint DoS, link flooding, wireless link jamming, etc. For example, a large number of compromised collectors can launch a distributed DoS attack to a headend. It is infeasible to provide protection against cyber threats from any number of compromised collectors. We present constraints, which can limit the possibility of such attacks. These constraints essentially incorporate the use of network limiters.

C. Objective

The correct functioning of AMI stands on consistent and secure execution of tasks in time. The safe security configuration depends not only on the local device parameters but also on the secure interactions and flows of these parameters across the network. There is a significant number of logical constraints on configuration parameters of millions of AMI devices, which need to be satisfied to ensure safe and secure communications among AMI components. These constraints represent system invariants and user-driven (i.e., organizational) security requirements. DHS AMI Task Force [1] and NIST (NISTIR 7628) [7] have developed security guidelines consisting of more than 300 security controls for ensuring trusted path, resource availability, boundary security protection, etc., towards controlling different security threats on AMI. Implementing these security controls in a scalable manner is one of the major challenges in smart grid security modeling. In addition, there is no such formal framework to support the energy providers for analyzing the constraints based on their own organizational and business requirements.

Researchers proposed different security analysis tools for analyzing misconfiguration problems in traditional networks [9], [10], [11]. These tools do not model complex heterogeneous configurations like time-driven data forwarding

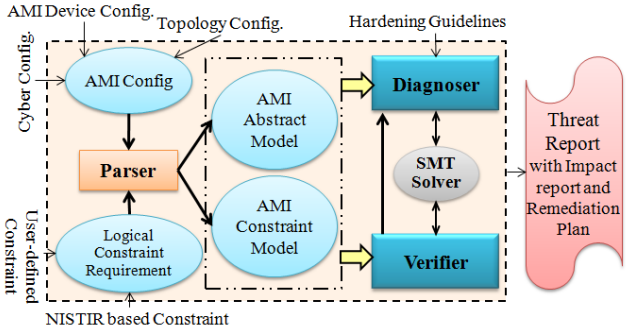

Fig. 4. The architecture of SmartAnalyzer.

and different security controls specific to a smart grid. The objective of this work is to develop an automated tool, SmartAnalyzer that will allow energy providers to objectively assess and investigate AMI security configurations for identifying and mitigating potential security threats and to enforce AMI operational and organizational security requirements. The major technical novelty of the tool lies in its capability of analyzing various safety critical constraints on AMI, such as (i) data overwrite protection, (ii) device scheduling and cyber bandwidth constraint, (iii) assured data delivery, (iv) data freshness, etc. Apart from these, the tool is capable of verifying various basic security properties, such as trusted path, data integrity, confidentiality, etc. Most importantly, the tool uses an SMT based formal analysis engine as the core and provides a proof-based threat report as the outcome, which can be comprehensively used for fixing the errors.

III. SmartAnalyzer Architecture

SmartAnalyzer is an automated security analysis tool for AMI that has the following functionalities:

- Providing an extensible global model abstraction capable of representing millions of AMI device configurations.
- Formal modeling and encoding of various invariant and user-driven constraints into SMT logics.
- Verifying the satisfaction of the constraints with AMI configuration using an SMT solver.
- Identifying potential security threats from the constraint violations and providing remediation plans for security hardening by analyzing the verification results.

SmartAnalyzer architecture is shown in Fig. 4. First, the tool parses a given *AMI configuration template* (e.g., Fig. 5 and Fig. 6) and encodes it into SMT logics. The generic configuration template is presented in a CSV (Comma Separated Value) text file.It consists of the device configurations (based on abstraction), topology, communication between the devices, data delivery schedules in the network, etc.

SmartAnalyzer parses (*parser* module) the configuration template and extracts a formal AMI configuration model. It uses a method of abstraction by exploiting the correlation between different configuration parameters in AMI. For example, a collection of meters with the same properties (e.g. meter type, vendor, network zone, scheduling info, etc.) are grouped in a meter class. Then, the tool models the organizational requirements and various security guidelines

Meter Class	ID	Type	Patch Info	Sampling Info	Reporting Mode (to Collector)	Report Schedule	Auth Property	Encrypt Property	Ports in Service	Comm Protocol				
meter	m00003	ge	pm011, pm115	18,40	push	15,40	auth1	encrypt1	nil	lontalk				
meter	m00123	echelon	pm115	15,30	push	20,30	auth0	encrypt1	nil	lontalk				
meter	m00129	echelon	pm0115	20,30	push	20,60	auth1	encrypt1	nil	lontalk				
Collector Class	**ID**	**Type**	**Patch Info**	**Buffer Info**	**Reporting Mode (to Headend)**	**Schedule (to**	**Pull Schedule (from meter)**	**ConnectedMeters**	**Connected Headend**	**Link (to Meter)**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**
collector	c0003	rev03	pc213, pc012	data, 9000, 1	pull	nil	nil	m00003,5; m00123,4	hs001	powerline1	auth1, auth2	encrypt1, encrypt2	22, 53, 161, 222	lontalk, ip
collector	c0005	rev02	pc012	data, 8000, 1	push	300, 14400	nil	m00003,5; m00129,5	hs001	powerline1	auth1, auth2	encrypt1, encrypt2	22, 53, 161, 222	lontalk, ip
Headend Class	**ID**	**Type**	**OS**	**Patch Info**	**Pull Schedule (from Collector)**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**					
headend	hs001	nil	win2010	p357, p254	180, 2880, c0003	auth2, none	encrypt2, none	22, 53, 161	ip					
Backend Class	**ID**	**OS**	**Patch**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**							
backend	bs001	fedora14	p357	none	none	22	ip							
Home Host Class	**ID**	**OS**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**								
home host	h0001	win_vista	none	none	_	ip								

Fig. 5. An example configuration data of an AMI topology.

(such as NISTIR 7628 standards) respectively as AMI user-driven and invariant constraints. For the purpose of threat verification, both the configuration and constraint model are encoded into SMT Logic. The salient feature of our framework is that the energy provider can flexibly incorporate more invariant constraints as per the smart grid security standards and user-driven constraints as per the organizational need. The combination of invariant and user-driven constraints is assumed to be complete to verify the potential security threats with respect to the smart grid standards and requirements.

The tool reports with empty threat in the case of all the constraints being satisfied in the verification process. On the other hand, the verification engine (an SMT solver [12][13]) provides an *unsat-core* in the case of one or more constraints being unsatisfied. The traces of this unsat-core are analyzed to generate a comprehensive threat report. Finally, the tool (*diagnoser* module) systematically analyzes the unsat-core traces in the threat report and creates a cumulative remediation plan including various possible alternatives. This remediation plan helps the administrators in reconfiguring AMI by directly fixing the configuration values or further incorporating new security alternatives. Most importantly, SmartAnalyzer provides the unsat-core traces as a proof or evidence of constraint violations (i.e., availability of potential threats) in the AMI configuration, which is the highlighted feature of the tool.

IV. Modeling AMI Configuration

An AMI configuration describes the device level configuration parameters, network topology, communication links, security properties, and the set of traffic control rules. Fig. 5 and Fig. 6 present an example of a partial AMI Configuration.

A. Configuration Level Abstraction

An enterprise smart grid network typically consists of millions of smart meters and thousands of collectors distributed over different geographical regions. These devices communicate for delivering data based on device configurations and communication properties. For the purpose of achieving better scalability, we introduce the concept of *abstraction* based on the similarities between the configurations of the devices. In our model, we represent this abstraction by defining classes for each kind of devices A particular class of device shares

Link	Src	Dest	Status	Link Type				
link	zc101	v1	up	gprs				
link	zhs101	r2	up	ethernet1				
link	v1	r1	up	fiber1				
Link Profile	**ID**	**Media**	**Mode**	**Shared Status**	**BW**			
link profile	powerline1	power_line	halfduplex	yes	5			
link profile	wifi1	wireless	fullduplex	no	150			
Auth Profile	**ID**	**Algo**	**Key**					
auth	auth0	sha1	96					
auth	auth1	sha1	160					
auth	auth_ipsec1	sha1	160					
Encrypt Profile	**ID**	**Algorithm**	**Key**					
encrypt	encrypt1	rc4	64					
encrypt	encrypt2	rc4	128					
encrypt	crypt_ipsec1	md5	64					
Fw Policy	**Node**	**Src**	**Src Port**	**Dest**	**Dest Port**	**Protocol**	**Action**	**Limit**
fw policy	f1	150.0.0.0/8	161	172.16.0.0/16	161	udp	allow	_
fw policy	f1	150.0.0.0/8	22	172.16.0.0/16	22	tcp	allow	_

Fig. 6. An example configuration data of an AMI security policy.

the same (physical and logical) configuration properties. We are not limited with this class based concept within meters and collectors. In order to have a common design model, the same concept is applied for all kinds of AMI devices. However, the cyber-physical devices (e.g., routers, firewalls, etc.) are not modeled as classes. As we are describing the AMI devices under the class model, no network identity (i.e., IP address) is possible as a class property. Hence, we introduce the concept of *zone* as a collection of similar (but not only limited to same class) AMI devices and provide network identity (usually subnet based) to the zone. The concept of zone, increases the scalability of the modeling of end-to-end network communication oriented constraints.

B. Modeling of AMI Physical Components

In this subsection, we present the formalizations of different AMI device configurations.

Smart Meter: A meter class is identified by an *Id* and its profile *SM* is represented as a conjunction ($\wedge$) of different parameters as shown in Table I. The vendor type (i.e., *Echelon*, *GE*, etc.) is represented by the parameter *Type*. We represent the sampling information of a meter using *SInfo*, which consists of two components, sampling size (*SSize* in KB) and sampling time (*STime*). A meter can deliver data to a collector in two different modes: *pull* and *push*. In *pull* mode, the meter reports data based on the request from the collector that follows a specific *pull schedule* of the collector. On the other

TABLE I
FORMAL DEFINITION OF AMI METER AND COLLECTOR

Smart Meter:

$SM_i \Rightarrow Type_i \wedge Patch_i \wedge SRate_i \wedge Mode_i \wedge RSche_i \wedge Auth_i \wedge Encr_i \wedge Serv_i \wedge CommProto_i \wedge TRate_i$

$Patch_i \Rightarrow \bigwedge_{j=0...} Patch_{i,j}$

$SRate_i \Rightarrow SSize_i \wedge STime_i$

$RSche_i \Rightarrow RScheBase_i \wedge RScheInt_i$

$Auth_i \Rightarrow \bigwedge_{j=0...}(AAlgo_{i,j} \wedge AKey_{i,j})$

$Encr_i \Rightarrow \bigwedge_{j=0...}(EAlgo_{i,j} \wedge EKey_{i,j})$

$Serv_i \Rightarrow \bigwedge_{j=0...} SPort_{i,j}$

$CommProto_i \Rightarrow \bigwedge_{j=0...} CommProto_{i,j}$

Intelligent Data Collector:

$IC_i \Rightarrow Type_i \wedge Patch_i \wedge BufSize_i \wedge Mode_i \wedge RSche_i \wedge PRSche_i \wedge Auth_i \wedge Encr_i \wedge ConnSM_i \wedge LinkToSM_i \wedge AttachHS_i \wedge Serv_i \wedge CommProto_i \wedge TRate_i$

.....

$PRSche_i \Rightarrow \bigwedge_{j=0...}(PScheBase_{i,j} \wedge PScheInt_{i,j} \wedge RDev_{i,j})$

$ConnSM_i \Rightarrow \bigwedge_{j=0...}(CSMId_{i,j} \wedge CSMNum_{i,j})$

hand, in *push* mode, the meter reports data to the collector (without waiting for request) based on its own *report schedule*. This reporting mode is captured by *Mode*. The reporting time schedule of a meter (in *push* mode) is modeled using *RSche*, which consists of *RScheBase* and *RScheInt*. This indicates that the meter will report periodically in a regular interval of *RScheInt* starting from *RScheBase* after the base time. To achieve end-to-end security, the communicating devices must agree in their authentication and encryption properties. We model the authentication properties of a meter using the parameter *Auth* as conjunction of algorithm (*AAlgo*) and key length (*AKey*). A meter may support multiple authentication properties. Encryption property is modeled similarly as *Encr*. The running services and communication protocols associated with a meter are represented by *Serv* and *CommProto* respectively. The parameter *Patch* denotes the patches that are installed in the meter. The maximum transmission rate (in Mbps) of a meter is denoted by the parameter *TRate*.

Intelligent Data Collector: A collector class profile *IC* is represented as a conjunction of different parameters, which covers the same parameters of meter class profiles except the sampling information. In addition, each collector may have a pull schedule that is represented by the parameter *PRSche*. It has three components: *PScheTime*, *PScheInt*, and *RDev*, which denote that the collector periodically pulls data from reporting device (*RDev*, a meter) starting at *PScheBase* with interval *PScheInt*. A collector has a buffer for storing the report data from different meters. *BufSize* represents the buffer size (in KB). The parameter *ConnSM* represents the conjunction of the connected meters (*CSMId*) to a collector class and their count (*CSMNum*). *LinkToSM* represents the type of the link (i.e., a link profile ID, refer to Section IV-C) that connects the collector to the meter. *AttachHS* represents the headend system to which data is reported by the collector. Table I shows the formalization (partial) of a collector class profile IC_i.

Headend System: A headend system class profile *HS* is a conjunction of the parameters: *Type*, *OS*, *Mode*, *TRate*, *Patch*, *PRSche*, *Auth*, *Encr*, *Serv* and *CommProto*. These properties are modeled as similar to those of a meter/collector.

TABLE II
MODELING OF ZONE AND DOMAIN

Zone:

$Zone_i \Rightarrow ZSn_i \wedge ZMem_i \wedge ZGw_i$

$ZSn_i \Rightarrow Ip_{i,j} \wedge Mask_{i,j}$

$ZMem_{i,j} \Rightarrow ZMId_{i,j} \wedge ZMNum_{i,j}$

.....

$ZMem_i \Rightarrow \bigwedge_{j=0...} ZMem_{i,j}$

Representation of a source:

$(S \Leftrightarrow Id \wedge ZId) \Rightarrow (Id \Leftrightarrow ZMId)$

Domain:

$Domain_i \Rightarrow \bigwedge_{j=0...} DZId_{i,j}$

Host Devices: An AMI network contains different types of hosts, such as (1) hosts of home area network (enterprise clients), (2) enterprise internal hosts, (3) enterprise application servers (backend systems), and (4) external hosts from the Internet. Hosts have considerably less parameters. For example, a class profile of enterprise client hosts has *OS*, *Auth*, *Encr*, *Serv*, *CommProto* and *TRate* parameters only.

C. Modeling of AMI Network Topology

An AMI topology defines the physical and logical connectivity between different AMI devices.

Router, Firewall and IPSec: We model router (*R*), firewall (*F*), and IPSec (*IS*) devices similar to [10]. We only introduce the traffic limiting capability of a firewall in the model using the parameter *FwLim* in its policy (*FwPolicy*), which denotes that whether traffic bandwidth should also be controlled along with access control action (*FwAct*). If a limit is applied on traffic, corresponding limit value is represented by *FwLimV*. Router selects the next-hop (*RNext*) for a particular traffic based on its forwarding policy (*RPolicy*).

Link: A link is identified by an ID (*LId*). Its profile is a conjunction of *NodePair* (i.e., the node-pair connected by the link) and *LinkStatus* (i.e., up or down). *LId* binds the specified link type to the predicate *LinkProp* that represents the properties of that link including *MediaType* (i.e., wireless, ethernet, etc.), *SharedStatus* (i.e., shared or not), *CommMode* (i.e., half-duplex, full-dulex, etc.), *LinkBw* (in Mbps). There can be also some especial properties, for example, *PrioritySlotRatio* for LonTalk type of links that represents the ratio of priority slots for congestion-randomization over normal slots.

Zone: We model logical zone as a collection of similar AMI devices. Each zone has an ID (*ZId*). The profile of a zone (*Zone*) consists of three parameters: *ZSn*, *ZMem* and *ZGw*. The parameter *ZSn* denotes an IP-address (with subnet Mask) that covers all devices in that zone. *ZMem* represents the IDs of different device classes and the number of devices under each class that belong to the zone. *ZGw* denotes the gateway router ID for that zone. The formalization of a zone, $Zone_i$ is represented in Table II. Any source or destination node of the traffic is represented as a conjunction of its *Id* and *ZId*.

The number of traffic sources or destinations depends on the number of zones and the number of classes in the zones. For example, if there are 50 zones and 4 collector classes per zone on average, then there are 200 possible source or destination collectors. In the rest of the paper, when we will refer a source or a destination, especially in the case of a traffic, intuitively it will mean a device class associated with a zone. It is to remember that meters are directly associated with a collector.

Domain: We define a domain as a group of similar kind of network devices. For example, all the smart meters in the AMI network are considered as a domain. Each domain has an ID (*DId*). The profile of a domain (*Domain*) has the list of zones (*DZId*s) that fall under the domain (see in Table II).

V. AMI Threat Analysis Model

The potential causes of security threats on AMI lie in violations of system invariants and security requirements. We classify them as *invariant* and *user-driven* constraints.

A. Invariant Constraints

There are various invariant constraints based on connectivity, data delivery schedule, resource availability, etc. For the purpose of successful communication, these constraints need to be satisfied in AMI. Table III shows the formalizations of different invariant constraints.

Reachability Constraint: The reachability constraint is the basis to achieve the transmission of the data between a pair of devices (if required). For example, at least one collector must be reachable from a meter for delivering the report data. Similarly, there should be reachability from collectors to a headend, so that each collector can forward the data to the headend. This constraint intuitively checks that the communicating devices should have a common protocol to communicate and the links between a pair of devices along with satisfaction of routing/security device policies. The formalization of general reachability constraint is shown in Table III. We first define the constraint *Forward* that checks whether the specific traffic $Tr_{S,D}$ (i.e., from S to D) can be transferred from a node (X) to another node (Y) (like state transition). Then, we define $CanReach$ and the reachability constraint (*ReachableConstr*) on top of this. In the constraint formalization, we also model the maximum possible transmission rate ($CanReachTrR$) by taking the minimum bandwidth of the links across the path along with the limits that may be imposed by a firewall.

Security Pairing Constraint: Consistent security pairing between two AMI devices is required in addition to reachability for successful communication. It states that there should be a matching of the authentication and confidentiality parameters between the communicating devices. That is, if the receiver expects some security requirements, the sender should comply with the requirement. For example, in Fig. 5, although there are 4 meters of class $m000123$ connected with a collector of class $c0003$, they are not allowed to communicate with the collector. This is due to the violation of security pairing (e.g., the mismatch of authentication parameters, $auth0$ and $auth1$). Similarly, a host from HAN will not be able to communicate to a meter, if the host does not comply with the communication protocol (e.g., LonTalk) supported by the meter.

TABLE III
Formalizations of Invariant Constraints)

Reachability Constraint:

$$Forward_{X,Y,Tr_{S,D},TrR} \Rightarrow$$
$$\quad Link_{X,Y} \wedge$$
$$\quad (((X \Leftrightarrow S) \wedge (ZGw_S \Leftrightarrow Y)) \vee (Y \Leftrightarrow D) \vee$$
$$\quad\quad (R_X \wedge RPolicy_{X,Tr_{S,D}} \wedge (RNext_X \Leftrightarrow Y)) \wedge$$
$$\quad ((F_X \Rightarrow FwPolicy_{X,Tr_{S,D}} \wedge FwAct_X \wedge$$
$$\quad\quad (FwLim \Rightarrow (TrR \Leftrightarrow min(LimVal, LinkBw_{X,Y})))) \vee$$
$$\quad\quad (\neg F_X \Rightarrow (TrR \Leftrightarrow LinkBw_{X,Y})))$$
$$CanReach_{A,B,Tr_{S,D}} \Rightarrow$$
$$\quad Forward_{A,B,Tr_{S,D},TrR} \vee$$
$$\quad (\exists C, CanReach_{C,B,Tr_{S,D},TrR2} \wedge Forward_{A,C,Tr_{S,D},TrR1} \wedge$$
$$\quad\quad (CanReachTrR_{A,B,Tr_{S,D}} \Leftrightarrow min(TrR1, TrR2))$$
$$ReachableConstr_{S,D} \Rightarrow CanReach_{S,D,Tr_{S,D}}$$

Pairing Constraint:

$$AuthPairing_{S,D} \Rightarrow$$
$$\quad (AAlgo_S \Leftrightarrow AAlgo_D) \wedge (AKey_S \Leftrightarrow AKey_D)$$
$$EncrPairing_{S,D} \Rightarrow$$
$$\quad (EAlgo_S \Leftrightarrow EAlgo_D) \wedge (EKey_S \Leftrightarrow EKey_{m,D})$$

Schedule Constraints:

$$MeterSampConstr_M \Rightarrow$$
$$\quad SM_M \wedge (Mode_M \Rightarrow ((STime_M \leq RScheInt_M) \wedge$$
$$\quad\quad (RScheBase_M \leq RScheInt_M))) \wedge$$
$$\quad ((SSize_M / STime_M) \leq TRate_M)$$

$$CollectorPullScheConstr_C \Rightarrow$$
$$\quad IC_C \wedge (((M \Leftrightarrow CSMId_C) \wedge \neg Mode_M) \Rightarrow PRSche_C)$$

Resource Constraints:

$$(TotalSData_C \Leftrightarrow \textstyle\sum_M SData_M) \Rightarrow$$
$$\quad (M \Leftrightarrow CSMId_C) \Rightarrow$$
$$\quad\quad (SData_M \Leftrightarrow (SSize_M \times CSMNum_C)))$$
$$CollectorBufConstr_C \Rightarrow$$
$$\quad IC_C \wedge (BufSize_C \geq TotalSData_C)$$

$$(TotalSRate_C \Leftrightarrow \textstyle\sum_M SRate_M) \Rightarrow$$
$$\quad (M \Leftrightarrow CSMId_C) \wedge$$
$$\quad (SRate_M \Leftrightarrow ((SSize_M / STime_M) \times CSMNum_C)))$$
$$CollectorTrRConstr_C \Rightarrow$$
$$\quad IC_C \wedge (TrR_C \geq TotalSRate_C)$$
$$CollectorBwOutConstr_C \Rightarrow$$
$$\quad IC_C \wedge (TotalSRate_C \leq LinkBw_{C,ZGw_C})$$

Authenticated Communication Constraint: The constraint *AuthCommConstr* requires end to end authenticated communication to the AMI core devices (i.e., meters and collectors). That is, the receiver requires the verification that the sender is authorized. This is similar to authentication security protocol pairing, except it ensures that authentication is done. That is, for example, if the receiver has no authentication requirement, *AuthCommConstr* will fail, although the security pairing is still successful. The sender's authentication is crucial for verifying that the data is received from a reliable source. For example, Lontalk, the communication protocol between meter and collector, provides an authentication mechanism using 48-bit key. The key has to be matched between the communication parties.

Report Schedule Constraint: The schedule constraints

ensure the basic correctness of a report or pull schedule. The constraint *MeterSampConstr* states that the sampling time and the reporting base-start time of a meter must be less than (or equal to) its reporting interval, such that no reporting is done without new data. It also verifies that the sampling rate cannot be more than its maximum transmission rate. If a meter follows the push reporting mode ($Mode$ is true), then it should have a reporting schedule. A similar constraint (*CollectorPullScheConstr*) states that if a collector is connected with some meters who follow the pull reporting mode ($Mode$ is false), then the collector should have a pull schedule for them.

Resource Constraint: There are different resource constraints, which are often relate to report schedules. The constraint *CollectorBufConsrt* states that the buffer size of a collector should be greater than or equal to the cumulative sampled data size of all the connected meters to that collector. Otherwise, data loss will occur in collector buffer under any report schedule. Similarly, the constraint *CollectorTrRateConstr* states that the cumulative sampling rate of the connected meters cannot be more than the maximum transmission rate (or the bandwidth of the link from the collector to its gateway) of the collector. Otherwise, no schedule will be possible without data loss.

B. User-driven Constraints

To achieve correct and secure functioning of an AMI network, there may exist different user-driven constraints. Here, we focus on AMI specific constraints, whose formalizations are shown in Table IV. Most of these constraints are modeled as compositions of different invariant constraints.

Assured Data Delivery: We define *assured data delivery* as checking of the end-to-end data delivery (from a meter to a headend through a collector) to satisfy the AMI global functionality. This requirement intuitively implies the satisfaction of all the invariant constraints (compositionality), which are the following: (1) reachability, (2) successful security pairing with ensuring sender authentication, (3) availability of resources (conjunction of all resource constraints including data overwrite constraint), and (4) synchronous reporting without flooding the cyber towards the headend (schedule constraints along with cyber bandwidth constraints across the path). A violation of any one of these constraints can create failure in data delivery.

Data Overwrite Protection Constraint: This constraint states that the aggregate report data of all the meters connected to a specific collector must not flood the collector buffer within the reporting interval. For example, in Fig. 5, a collector of class $c0005$ receives reports from 5 meters of the class $m00129$ (sampling rate: 20 KB per 30 seconds) and 5 meters of the class $m0003$ (sampling rate: 18 KB per 40 seconds). Therefore, the collector will receive 335 KB (in average) of data every 60 seconds, which is to be stored in its buffer. According to the report schedule, the collector pushes the data to the headend every 1440 seconds. Thus, during this period, total 8040 KB of data will be sent to the collector by these meters. This amount of data will flood the collector buffer (size 80000 KB), which will in turn cause data loss, i.e., initial 40 KB of data will be overwritten.

TABLE IV
FORMALIZATIONS OF DIFFERENT USER-DRIVEN CONSTRAINTS

Data Overwrite Protection Constraint:

$(TotalRData_C \Leftrightarrow TotalSRate_C \times Period) \Rightarrow$
$((Mode_C \Rightarrow (Period \Leftrightarrow RSInt_C)) \vee$
$(\neg Mode_C \Rightarrow (H \Leftrightarrow AttachHS_C) \wedge (Period \Leftrightarrow PRSInt_H)))$
$OverwriteProtectConstr_C \Rightarrow$
$IC_C \wedge (BufSize_C \geq TotalRData_C)$

Cyber Bandwidth Constraint:

$(Num_C \Leftrightarrow \sum_Z ZMNum_Z) \Rightarrow (MId_Z \Leftrightarrow C)$
$(TotalRRate_{H,Sche} \Leftrightarrow \sum_C (TotalSRate_C \times Num_C)) \Rightarrow$
$(H \Leftrightarrow AttachHS_C) \wedge Mode_C \wedge (RSche_C \Leftrightarrow Sche)$
$LinkBwConstr_{H,X,Y} \Rightarrow$
$HS_H \wedge (LinkBw_{X,Y} \geq TotalRRate_H)$

Secure Tunnel Constraint:

$SecureTunnelConstr_{M,C,H} \Rightarrow$
$ReachableConstr_{C,H} \wedge$
$InTrafficPath_{C,H,X} \wedge InTrafficPath_{C,H,Y} \wedge$
$SecTunnel_{C,H,X,Y} \wedge (SecKeyLen_{C,H,X,Y} \geq K_T)$

Priority Delivery Constraint:

$TotalEmergeBw_C \Leftrightarrow$
$\sum_{\exists_i M \Leftrightarrow CSMId_{C,i}} EmergeMsgProb_M \times CSMNum_{C,i}$
$LonTalkPriorityConstr_C \Rightarrow$
$(L \Leftrightarrow LinkToSM_C) \wedge MediaType_L \Leftrightarrow' LonTalk' \Rightarrow$
$TotalEmergeBw_C \leq (LinkBw_L \times PrioritySlotRatio_L)$

Quality of Delivery Constraint:

$FreshnessConstr_{M,C,H,T} \Rightarrow$
$AssuredDelivery_{M,C,H} \wedge$
$((Sum_{T1,T2} \leq T) \Rightarrow (((T1 \Leftrightarrow RSInt_M) \wedge Mode_M) \vee$
$((T1 \Leftrightarrow PRSInt_C) \wedge \neg Mode_M)) \wedge$
$(((T2 \Leftrightarrow RSInt_C) \wedge Mode_C) \vee$
$((T2 \Leftrightarrow PRSInt_H) \wedge \neg Mode_C))$

Availability Protection Constraint:

$(MaxTrR_{H,X,Y} \Leftrightarrow \sum_C CanReachTrR_{Tr_{C,H},X,Y} \times Num_C) \Rightarrow$
$Compromise_C \wedge (AttachHS_C \Leftrightarrow H) \wedge CanReach_{Tr_{C,H},X,Y}$
$AvailProtectionConstr_{H,X,Y} \Rightarrow$
$IC_C \wedge (LinkBw_{X,Y} \geq MaxTrR_{H,X,Y})$

Fault-tolerant Constraint:

$LinkFailConstr_{S,D,L} \Rightarrow$
$(L \Leftrightarrow X,Y) \wedge ReachableConstr_{S,D} \wedge FaultyLink_L$

$DevFailConstr_{M,C} \Rightarrow$
$\exists_{\hat{C} \neq C} ReachableConstr_{M,\hat{C}} \wedge FaultyDev_C$

Domain Boundary Protection Constraint:

$DomBoundarySecConstr_{D1,D2} \Rightarrow$
$\forall_X \forall_Y \exists_Z (ZId_X \Leftrightarrow DZId_{D1}) \wedge (ZId_Y \Leftrightarrow DZId_{D2}) \wedge$
$ReachableConstr_{X,Y} \wedge InTrafficPath_{X,Y,Z} \wedge F_Z$

Cyber Bandwidth Constraint: The constraint *LinkBwConstr* conforms that the aggregated report rate of the collectors reporting simultaneously due to matching report schedules should not exceed the bandwidth limitation of the network path (considering a link from X to Y) connecting to the headend. Violation of this constraint will cause link congestion (DoS).

Secure Tunnel Constraint: This constraint checks whether

the data is transmitted from a collector to a headend after satisfying the security (IPSec) tunnel requirement across a designated path. In this case, the type of tunneling may be single, nested, or even both/hybrid. There may exist more constraints for ensuring the quality of the tunnel. For example, this requirement can be defined as the satisfaction of minimum key length (say, 256 bits) and/or the number of nested tunnels (say, 2-level of nested tunnels).

Priority Delivery Constraint: The data delivery in AMI may be realized based on prioritization constraint according to the enterprise policy. For example, the LonTalk protocol allows for priority message slots (to cope with the 4% cases of data delivery failures due to its collision avoidance mechanism). One can define slots in which no other device can use to randomize for collision avoidance. Those priority slots can be reserved for those key messages (control messages) that must have access, no matter how busy the network is. The constraint *LonTalkPriorityConstr* states that the priority messages should have the necessary bandwidth that is required for them to successfully transmit without any collision, that is, the bandwidth dedicated for priority messages should be enough considering the maximum possible priority traffic size. There can be a constraint to establish that a collector should have multiple traffic queues in order to support prioritized message transmission to the headend system, at least one queue for emergency (i.e., alarm) traffic and one queue for normal traffic.

Quality of Delivery Constraint: There are user-driven constraints for ensuring the quality of delivery. For example, the report freshness constraint (*FreshnessConstr*) restricts the delivery of data within a specific time window, say, T. A user can also have constraint on the quality of the trusted path. For example, this requirement can be defined as the satisfaction of end-to-end authentication or encryption based on a key length that the length should be at least 128 bits.

Availability Protection Constraint: This constraint (*AvailProtectionConstr*) ensures that if there are X number (or portion) of AMI devices being compromised, assured data delivery constraint is still preserved. It intuitively verifies that DoS attack is not possible on links or endpoints, when number of compromised nodes is no more than X (say, 5% collectors).

Fault-tolerant Constraint: Users may require to check availability of alternative measures when any AMI component fails. The constraint *LinkFaultTolerantConstr* denotes that if a link is down, the data delivery should not fail. For example, if the link between a collector (in $zc101$ in Fig. 6) to its gateway router ($v1$) fails, the collector is required to send the report data received from its connected meters to a neighbor collector (which might fall under a different zone, e.g., $zc102$), which will forward the data to the headend, which ensures the end-to-end data delivery between the meters and the headend. The constraint *DevFaultTolerantConstr* requires that if an intermediate device fails, the reachability (data delivery) still holds. For example, if a collector is down, then the meters, which were designated to send data to this collector, should be able to communicate with a different collector. A more strict fault-tolerant constraint can be *n-fault tolerance*, which means that if n number of device or link faults happen, still the data delivery is ensured. For example, if smart meters can be deployed as a mesh network, this network is capable of self-healing like routers, if the preferred neighbor meter for transmitting data to collector/access point fails, the sending meter can use another meter to transmit the data. Hence, for an example, a requirement corresponding to each meter can be defined as how many neighboring meters can fail, although data transmission is still possible.

Domain Boundary Protection Constraint: In these group of user-driven constraints, we check whether communication between different boundaries in the smart grid network are protected with security devices (i.e., Firewalls, IDS, IPSec, etc.) and appropriate authentication and data encryption methods. The constraint *DomBoundarySecConstr* requires that there should be at least a security device (typically firewall) between any communication path between two devices of two different domains. Similarly, there are also more constraints for domain boundary protection in order to ensure that, for example, every communication from the devices of a domain to the devices of another domain must be done with proper encryption.

C. NISTIR Security Controls as Constraint Violations

The above mentioned constraints provide building blocks for verifying various important security controls documented in NISTIR 7628 guidelines [7]. For example, data overwrite protection, cyber bandwidth, and availability protection constraints fall under the NISTIR SG.SC-5 (DoS attack protection) control. Similarly, domain boundary protection constraints represent the NISTIR SG.SC-7 control. Authenticated communication and security tunnel constraints collectively ensure NISTIR SG.SC-10 (trusted path) control. In addition, priority delivery and fault-tolerant constraints implement NISTIR SG.SC-6 (resource priority) and SG.CP-11 (fail-safe response) controls respectively.

D. SMT Encoding and Constraint Verification

We use Boolean terms to encode the Boolean configuration parameters. We also use Boolean terms to encode some of the integer configuration parameters, which usually take a very small range of values. The remaining parameters are modeled as integer terms. We normalize the parameters into integers that may take real values (e.g., bandwidth). We use *bit-vector* terms for encoding IP addresses. In some of the computations, we require multiplying/dividing two variables. But, *Yices* SMT solver [13] does not support such non-linear operations. Thus, earlier in [5] we encoded such operations by normalizing one of the variables to a small set of possible values and applying the operation on the other variable with one of those values if it matches with the former variable. Due to this shortcoming of Yices, we have moved to *Z3*, another very well-known SMT solver [12]. This tool supports non-linear operations. We have applied Z3 .NET API to implement SmartAnalyzer. The tool (*parser* module) reads the configuration from the input template file and directly builds the model using the API.

After defining the configuration parameters as SMT variables, we model the configurations associated with the AMI network topology and the AMI components. We encode each constraint under the same formalism. Then SmartAnalyzer

creates a verification query that checks the satisfaction of a constraint with the configuration. If M_{Conf} and M_{Constr} are the AMI configuration and constraint models respectively, the verification query Q is encoded as the following clause:

$$Q \Rightarrow M_{Conf} \wedge M_{Constr}$$

VI. Verification Result Diagnosis, Hardening, and Impact Analysis

The verification engine of SmartAnalyzer (i.e., Z3 SMT Solver) generates the verification results, which is either *sat* (satisfiable) or *unsat* (unsatisfiable). In the case of an unsat, the verification engine provides an *unsat-core* that basically represents the traces of constraint violations in the configuration. Then, SmartAnalyzer (*diagnoser* module) systematically analyzes these violation traces and generates a comprehensive threat report for overall AMI configuration verification. This report includes threat sources, targets, violating rules and reasonings, and a remediation plan showing possible reconfigurations for hardening the problems.

A. Methodology of Unsatisfied-core Generation.

If the SMT solver gives an unsat result, we need to get the unsat-core, which tells about the unsatisfied constraints and the corresponding configurations that the constraint does not support. In order to get the unsat-cores in the case of model failures, we use the concept of *hard and soft clauses* (*assumptions checking* in Z3) for verification. We take the configurations as a group of assumptions and the constraints as a different group of assumptions for verifying the satisfaction of the model. If the model verification fails, the unsat-core shows the list of assumptions, i.e., the constraints and the configurations, which are not satisfied. From the list of the unsatisfied configurations, it is possible to trace the reasoning of failure of a constraint.

B. Methodology of Remediation Plan Synthesis.

In order to get a remediation plan, we consider a policy for the reconfigurations. The policy shows feasible or preferred invariant and user-driven guidelines about possible candidates for reconfigurations. An invariant guideline represents the configurations, which are practically infeasible to modify. The vendor specific device configurations (such as the buffer size of a collector) are usually constant for a device. Hence, changing this property requires replacing the device with a different or newer product that has the required configuration property. The user-driven guidelines represent the organizational priorities or capabilities on the reconfigurations for remediation from threats. For an example, the organization may be fine with deploying many collectors, but a minimum number of meters must be connected to each collector.

In the process of exploring the reconfiguration plan, the diagnoser module continuously checks the satisfaction of the model by releasing the assumptions (soft clauses) of the configurations systematically according to the remediation guidelines until the model verification gives a sat result. Releasing an assumption lets the solver choose the configuration values associated with the assumption that satisfy the hard clauses along with the remaining assumptions. This process is an implementation of *max-sat* [14]. Then, a remediation plan is generated from the max-sat output. It is worth mentioning that we use quantifiers for the purpose of verifying some constraints. In such cases, Z3 may return *unknown* instead of *sat*. This implies that there is no constraint violation found by the solver. Hence, if the result is not unsat, we consider that the model is satisfied with the given constraints.

TABLE V
A Simple Example of Resource Constraint Verification

```
(assert (M 0)) ;; Meter 1 (Id 0)
(assert (= (Id 0) 0))
(assert (= (SSize 0) 25))
(assert (= (SInt 0) 45))

(assert (M 1)) ;; Meter 2 (Id 1)
(assert (= (MId 1) 1))
(assert (= (SSize 1) 15))
(assert (= (SInt 1) 30))

(assert (IC 10)) ;; Collector 1 (Id 10)
(assert (= (Id 10) 10))
(assert (= (BufSize 10) 200))
(assert (=> P0 (= (CSMId 10 0) 1)))
(assert (=> P1 (= (CSMId 10 1) 0)))
(assert (=> P2 (= (CSMNum 10 0) 8)))
(assert (=> P3 (= (CSMNum 10 1) 8)))

(assert (=> PC
   (=> (CollectorBufConstr 10)
      (and (M (CSMId 10 0)) (M (CSMId 10 1))
       (= (SData 10 0) (* (CSMNum 10 0) (SSize (CSMId 10 0))))
       (= (SData 10 1) (* (CSMNum 10 1) (SSize (CSMId 10 1))))
       (>= (BufSize 10) (+ (SData 10 0) (SData 10 1)))))))

(assert (CollectorBufConstr 1))

(check-sat PC P0 P1 P2 P3) ;; Sat
(get-model)
(get-unsat-core) ;; Unsuccessful
```

C. Verification Trace Analysis: An Example

This section describes how verification traces (results) from the SMT solver are analyzed to find the reasons of constraint violations and remediation plans for them. We explain the procedure with an example of constraint verifications.

In our example, we evaluate the collector resource (buffer) constraint (Table III). Table V shows the SMT-LIB encoding of the AMI configuration (required segment only) and the collector resource constraint. To comprehend the verification trace, we consider a tiny AMI configuration with two meters, one collector, and one headend. Here, the constraint verification gives a sat result. This signifies that the AMI configuration satisfies the resource constraint. Thus, there is no unsat-core in this evaluation. However, we get an unsat result when the AMI configuration model is modified by setting the buffer size to a reduced value, 100 KB. The evaluation result is presented in Table VI. It shows that there is no model that satisfies the collector resource constraint. It also shows the unsat-core, i.e., the unsatisfied constraints (assumptions). Then, to find a sat result, we run the max-sat implementation on the configuration model sequentially by intuitively weakening the configuration constraint following the unsat-core. This is done

TABLE VI
AN EXAMPLE OF DIAGNOSIS PROCESS

Modified Model:

```
. . . . . .
(assert (= (BufSize 10) 100))

. . . . . .
(check-sat PC P0 P1 P2 P3) ;; Unsat
(get-model)
(get-unsat-core)
```

Solver Output:

```
unsat
ERROR: model is not available
(P0 P1 P2 P3 PC)
```

Max-SAT Implementation:

```
. . . . . .
(assert (forall ((c Int) (x Int))
   (=> (and (>= c 10) (<= c 10) (>= x 0) (<= x 1))
     (and (>= (CSMId c x) 0) (<= (CSMId c x) 1)))))

(assert (forall ((c Int) (x Int))
   (=> (and (>= c 10) (<= c 10))
     (>= (+ (CSMNum c 0) (CSMNum c 1)) 6)
     (>= (CSMNum c 0) 0) (>= (CSMNum c 1) 0))))

. . . . . .
(check-sat P0 P1 P2 PC) ;; Unsat
(get-model) ;; Unsuccessful
(get-unsat-core) ;; (P0 P1 P2 PC)

. . . . . .
(check-sat P0 P1 PC) ;; Sat
(get-model) ;; Successful
(get-unsat-core) ;; Unsuccessful
```

Satisfied Model:

```
. . . . . .
CSMNum − > {
   10 1 − > 6 ;; The number of type 1 meter is 6
     else − > 0} ;; The number other type (type 0) meter is 0
. . . . . .
```

by removing one predicate (among the predicates of the unsat-core) from the configuration constraint each time and running the model verification until the verification result converges to sat. Obviously, the resultant satisfiable model indicates the configurations that satisfy the resource constraint. Then, we use the immediately preceding unsat trace as the potential trace of the constraint violation. For example, Table VI shows that the configuration predicates P0 and P1 hold the resource constraint. In this case, the number of the type 0 meters and that of the type 1 meters are respectively 6 and 0. The immediately preceding unsat trace reported is "PC P0 P1 P2" which indicates that P2 predicate leads the violation. It can be observed that P2 predicate in the configuration (as shown in Table V) asserts the number of each type of meters as 8, which leads to the unsatisfiability of the resource constraint when the buffer size is 100 KB.

From the unsat-core, it is found that the remediation to the collector resource constraint violation is possible by applying one of the following measures: (i) changing the collector's buffer size, (ii) changing the sampling size or rate of the meter(s), and (iii) changing the number of meters to transmit

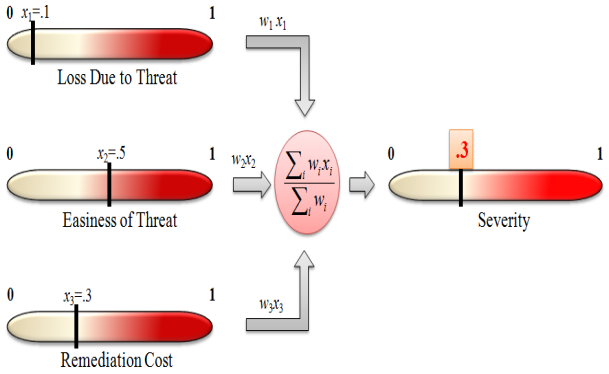

Fig. 7. An example of a graphical impact report for a collector resource constraint violation. Here, AMI consists of 100 meters and 10 collectors, where 10 meters are connected to each collector.

data to the collector. The buffer size of a collector is basically vendor-specific and this is not configurable except by replacing the collector with a different one (having a larger buffer). This is an example of invariant guidelines that the collector's buffer size cannot be considered in the remediation plan. The sampling rate of a meter is also vendor-specific. However, it might be possible to replace a meter with a different one (chosen from the available meters) that has a smaller sampling size or rate. Hence, in our example, we consider the connected meters to the collector as assumptions (P0 and P1). It is easy to change the number of meters connected with the collector. We take the assumptions P2 and P3 corresponding to the numbers of meters. However, in the diagnosis process, we could assume that the organization is using only one vendor-specific type of meter (e.g., the type 0 meter) and presently the organization is not willing to try different kinds. This is an example of user-driven guidelines, when we would not consider the assumptions P2 and P3.

D. Impact Analysis

We analyze the impact of a potential threat, i.e., a constraint violation, by considering three factors: (i) loss due to the threat occurrence, (ii) easiness of the threat execution, and (iii) remediation cost of the threat. We compute the *severity* of a threat from the weighted sum of these factors. The user specifies the weight for each factor. If the weights are not specified, all factors are weighted the same.

As the main goal of an AMI network is to enable communication between meters and the energy provider's headend system, we compute the loss due to a threat occurrence as the ratio of the number of affected meters to the total number of meters in the AMI network. Hence, the loss value ranges from 0 to 1. The number of meters affected by a specific threat is identified in the threat report. In addition, a meter can be impacted indirectly by a threat occurrence on a collector (e.g., a collector resource constraint violation). In this case, the meters associated with this collector will fail to send data to the headend system. Therefore, in such cases, the total number of affected meters is computed from the affected collectors and the association of meters with these collectors (Section IV-B).

The easiness of a threat execution is measured by the access capabilities required for executing the threat. For example, if

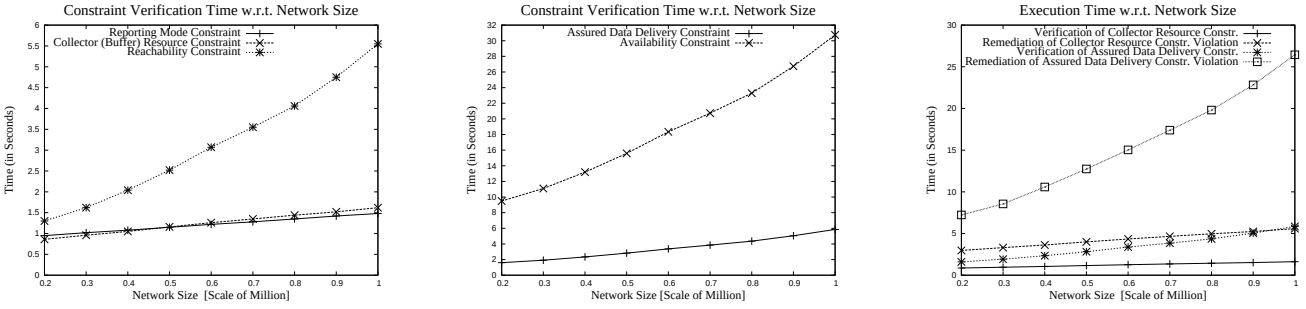

Fig. 8. Impact of the network size on (a) the invariant constraint verification time, (b) the user-driven constraint verification time, and (c) the remediation plan finding time in case of constraint violation.

a threat occurs due to an Internet access, the execution of this threat is easier compared with the internal user's access. In our implementation, we consider three different types of access capabilities: (i) Internet, (ii) home area network, and (iii) internal enterprise network, in order to classify the easiness of a specific threat as (i) high (from $1.0 - 0.7$), (ii) medium (from $0.7 - 0.4$), and (iii) low (from $0.4 - 0.1$), respectively. However, users can also create further classifications based on fine-grain access capabilities.

The cost for the remediation of a threat is computed from the remediation plan, which provides the reconfigurations that are required to eliminate this threat. The remediation cost is computed as the ratio of the total reconfiguration cost to the maximum affordable cost. The reconfiguration cost for each individual reconfiguration as well as the maximum affordable cost are specified by the user. In Fig. 7, we show an example of the impact analysis considering a threat on a collector, which occurs due to a resource constraint violation. In this example, since 10 meters are connected to the collector, all of them will be affected because of this threat occurrence. The figure shows the above mentioned three factors and the associated severity.

VII. Evaluation

In this section, we evaluate SmartAnalyzer in terms of *accuracy*, *usability*, and *scalability*.

A. Accuracy

Firstly, the accuracy of our tool is ensured by the use of a formal constraint satisfaction checking method. In addition, we evaluate our tool with ground truth scenarios by deploying it in a small AMI testbed created in our university [15]. The testbed setup typically represents a small subset of the network shown in Fig. 1. We analyze some of the security constraints, especially, data overwrite protection and cyber bandwidth constraints. The results of our tool are cross-validated with the real scenario. For the purpose of analyzing the constraints, we slide the values of different configuration parameters, such as (i) taking very low and high pull schedule intervals for the headend, and (ii) changing the bandwidth of the links from high to very low. We find some constraint violations that lead to link flooding and data loss. In addition, we inject high amounts of data through the simulation (by adding multiple simulated collectors in the testbed) to observe its effect on cyber bandwidth constraint. After observing the constraint violations, we reconfigure the setup according to the remediation plan and reevaluate the constraint to see the effect. For example, in the case of a cyber bandwidth constraint violation, we add traffic limit in firewall rules and observe the resolving of link flooding. These tests significantly help us in verifying the accuracy of the tool.

B. Usability

The usability of SmartAnalyzer is evaluated by providing it to different real-life experienced users and considering their feedback. The main usability of our tool lies in the operational efficiency. It allows users (i) to evaluate new AMI configurations, and (ii) to modify and reevaluate the configuration within a few seconds (particularly in 5-7 mouse clicks). The input (configuration template) and output (threat report) of the tool are simple to understand and are easy to use. In addition, remediation instructions in the threat report allow a user to reconfigure the data accordingly.

C. Scalability

We evaluate the scalability of SmartAnalyzer by analyzing the *time* and *space* required in constraint verification by varying the AMI network size. We consider the network size as the total number of collectors in AMI (the number of meters are proportional to the number of collectors). The number of collectors depends on the number of collector zones and their sizes. We consider only a single headend zone (10 headends of two headend classes) in the network. We take 100 and 50 meter and collector classes respectively, while each collector is connected with 10 meters (of 2 random meter classes) on average. Each collector zone consists of around 1000 collectors (of 5 random classes). We keep the values of these parameters fixed in most of the experiments, except those cases where their impacts on the scalability are analyzed.

Impact of network size: Fig. 8(a) and Fig. 8(b) show the constraint verification time with respect to the network size. We show the verification time for different invariant constraints (i.e., reporting mode, collector resource, and reachability) and user-driven constraints (i.e., assured data delivery and availability protection constraint). A significant part of the constraint analysis time is covered by the modeling (SMT logic encoding) time, which is almost linearly dependent on the network size that varies with number of zones. Verifications of some constraints involves all (or a portion of) possible

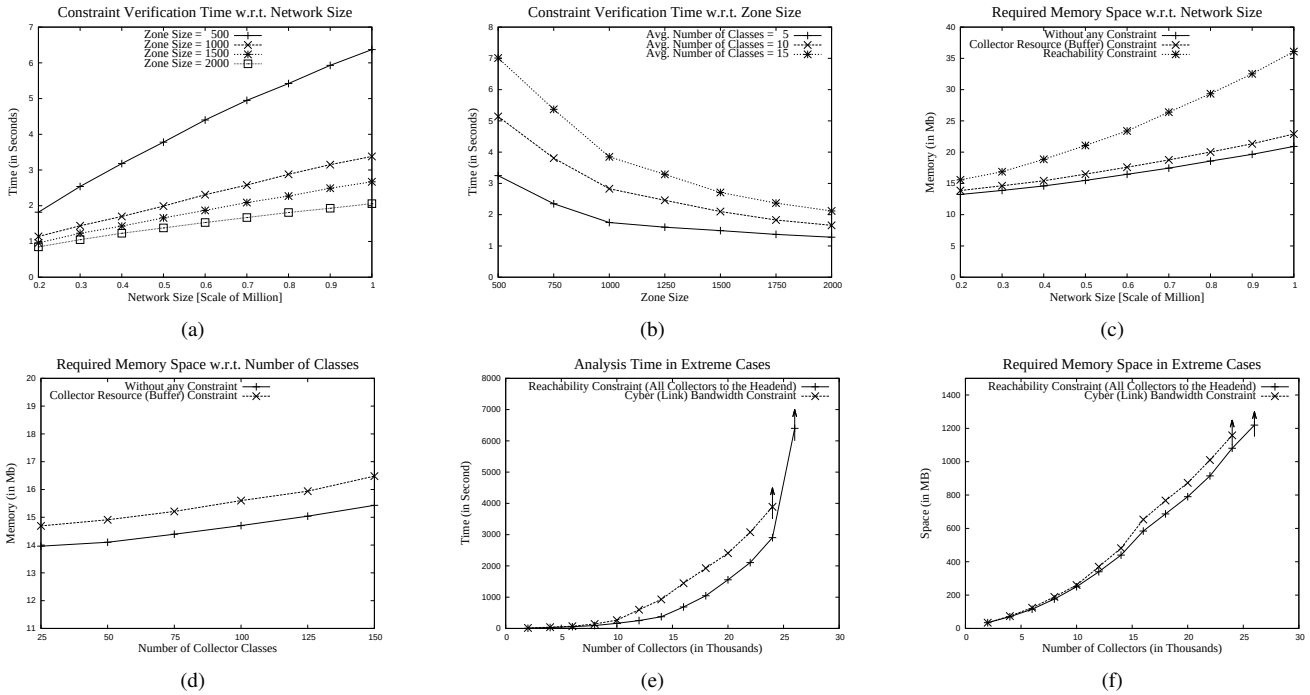

Fig. 9. (a) Impact of the zone size on the constraint verification time, (b) impact of the number of collector classes per zone, (c) impact of the network size on the memory requirement, (d) impact of the number of classes on the memory requirement, (e) - (f) impact of the AMI network size on the analysis time and the memory requirement for cyber bandwidth constraint verification, which show that the analysis of this constraint fails when the number of collectors increases more than some certain labels.

source or target nodes (Section IV-C), which increase with the number of zones. Thus, the verification time of such kind of constraints (e.g., reachability) increases more with the size of the network than that of the constraints (e.g., collector resource), which are involved with the class size only. Usually, the user-driven constraint analysis time is more than the invariant constraint analysis time (see Fig. 8(b)), since most of the former constraints subsume the later constraints. We also evaluate the impact of the network size on finding the remediation plan in case of constraint violations. Fig. 8(c) shows the evaluation results of finding remediation plans in two cases: (i) a collector resource constraint violation and (ii) an assured data delivery constraint violation. In both of the cases, we observe that the time for finding a remediation plan is longer than that for the verification only. The reason is as follows. The remediation plan is required when a violation is occurred, i.e., the solver gives an unsat result. In an unsat case, the solver needs to search the whole potential solution space to conclude that there is no model that satisfies the constraint. Hence, it takes a longer time compared to a sat case. To find a remediation plan, we execute the max-sat process, which requires another one or more verifications of the model (Section VI-C). Therefore, the execution time for finding a remediation plan is significantly long.

Impact of zone size and member classes: We evaluate constraint verification time with respect to different network zone sizes. This analysis is shown in Fig. 9(a) considering the reachability constraint. We observe that the analysis time significantly reduces with the increase in the number of collectors in the zone. This is due to the fact that the number of zones decreases as the zone size increases, which in turn decreases overall model size and the potential targets. Fig. 9(b) shows the constraint verification time taking a fixed zone size and varying the number of average classes per zone. We find that the time increases, if variation of classes increases.

SMT space requirement: The space requirement (memory used) of the SMT solver [12] is evaluated by changing the network size (i.e., number of zones) and the number of classes. Such analysis results are shown in Fig. 9(c) and Fig. 9(d). We observe that the space requirement increases linearly with the network size. Similar to the analysis time, the space for constraint verification is the sum of the space for modeling the AMI configuration and that for modeling a constraint. The figures justify this by showing that less space is required when no constraint is verified. The constraints involving more quantifiers require larger memory space for encoding. Fig. 9(c) shows such a comparison between collector resource and reachability constraints.

Extreme cases when SMT solver fails : If the model size increases significantly, SmartAnalyzer fails to give a solution. Increase of the model size depends not only on the AMI network size (especially, the AMI components) but also on the constraint type. We present this event in Fig. 9(e) and Fig. 9(f). In the figures, we show the time and space requirements of all reachability (*ReachableConstr*) satisfaction (all collectors to the headend) as well as the cyber bandwidth constraint *LinkBwConstr* satisfaction. The modeling of *LinkBwConstr* also requires knowing all the traffics (and traffic size) from collectors to the headend passing through a link at a particular time (according the reporting schedules). Due to the modeling of all possible traffics between the collectors and the headend, the model size becomes very large. The figures show that if the number of collectors increases more than 23 thousand (arrow sign in the figures), the cyber-bandwidth constraint verification

fails. Similarly, all reachability verification fails if the number of collectors is higher than 26 thousand. These failures happen due to the *out-of-memory-exception* given by the SMT solver. Fig. 9(f) shows the memory space consumed by the model.

VIII. Discussion

In this section we discuss the issues about the limitation, extensibility and deployment of SmartAnalyzer.

A. Limitation

SmartAnalyzer can successfully identify possible threats on AMI by constraint satisfaction checking. It is highly scalable with the network size. However, there is a couple of limitations of the tool. First, we have used device and property level abstraction for achieving scalability under large scale smart grid configuration, which in turn may not provide fine-grain attack paths. Second, an SMT solver is used as the core analysis engine, different normalizations are considered for real valued calculations. Moreover, the tool does not provide the functionality for analyzing some of the inherent smart grid security properties, such as LonTalk protocol configuration.

B. Extensibility

Our proposed model is flexible to model any AMI network consisting of smart meters, intelligent collectors, and headend systems. All existing AMI systems follow this general structure with variability in connectivity structures, technologies and configuration parameters, which are possible to define in a flexible way in our model. For example, communication parameters (i.e., link types, their bandwidths, transmission protocols, etc.) and device configurations (i.e., meter sampling rate, collector buffer size, meter to collector assignment, etc.) are formalized as variables in our model. As a result, our model can adapt to various existing AMI systems. Moreover, to incorporate a completely new configuration capability, new constructs need to be added to the model and used in the query interface and the analysis engine.

C. Deployment Issues

SmartAnalyzer takes inputs from a template (Fig. 5 and Fig. 6) and generates a threat report with necessary remediation plans as outputs. The deployment of SmartAnalyzer can be fully-automated using SNMP-based configuration management tools. These tools can remotely extract the configurations of the AMI devices. This configuration data can be parsed to generate the AMI input template. The remediation plan generated by our tool can be executed using the configuration management tool by sending, e.g., SNMP SET commands. This is possible mainly in the cases of the reconfiguration-based remediation. In other cases, the execution of a remediation plan may require manual interventions.

IX. Related Work

Throughout the last decade, the security policy misconfiguration and its verification have been studies extensively in [16], [9], [10], [11]. In these approaches, the formal definition of configuration anomalies and safe deployment of single or multiple security devices have been proposed and algorithms were presented to discover configuration inconsistency. There is also a number of works on risk-based security configuration analysis. Risk analysis using attack graphs has been proposed in [17], [18]. The attack graphs are used in In [17] to predict the various possible ways of penetrating a network to reach critical assets. The authors in [18] model the problem of selecting a set of security hardening measures to minimize the residual damages in a predefined attack graph within a budget. Other works [19], [20] have proposed to find optimal deployment of security devices using attack graphs in order to block all attack scenarios. However, all these above mentioned security analysis tools are proposed for analyzing misconfiguration problems in traditional networks. These tools do not model time-driven data forwarding and different operational and security controls specific to a smart grid.

Last few years a significant number of works [21], [7], [1] have been initiated on describing the interoperability among heterogeneous smart grid components including security issues based on different attack scenarios. These works also describe the operational functionalities of AMI components and the corresponding energy provider's internal system with guidelines for secured communication between them. They advice that the utilities should not be trusted to be ensured that proper security is implemented. McDaniel et al. [3], [22] present the security and privacy challenges in smart grid networks. The works report that the customers work closely with the utility to manage energy usage in the smart grid, requiring that they share more information about how they use energy and thus exposing them to privacy invasions. Energy use information stored at the meter and distributed thereafter acts as an information-rich side channel, exposing customer behaviors. Wang et al. [23] present an artificial intelligent based approach for analyzing risks in smart grid networks. However, in their analysis, they do not consider network link capacity, bandwidth and different modes of communications between the smart grid components. Anwar et al. propose a couple of frameworks [24], [25] for modeling power grid and its control elements using first order logic. These frameworks are capable of evaluating power flows, overloading violations in smart grid. Liu et. al. [26] present a study on false data injection attacks in power grid. McLaughlin et. al. [4] present an approach for penetration testing on AMI systems. They develop archetypal and concrete attack trees for energy fraud, denial of service and targeted disconnect attacks. However, these works do not analyze various misconfiguration problems and security controls on power grid networks.

The survey reveals that no significant research has been done on formal modeling of the complex AMI configuration and analyzing various security constraints on the configuration. Therefore, SmartAnalyzer is a novel and useful tool for provably analyzing operational consistency and security controls in

AMI. Moreover, the tool provides possible remediation plans for the constraint violations, which are useful for reconfiguration planning towards security hardening.

X. CONCLUSION

In this paper, we present an automated AMI configuration verification, diagnosis and repair technique that is implemented in a tool called SmartAnalyzer. We define various AMI system invariants and constraints that are important for protecting AMI from classes of security threats. Using these constraints and the AMI configuration, we create a logic-based formal model of AMI and we then use SMT to solve the constraint satisfaction problem. Our implemented tool performs static configuration analysis in order to determine potential threats due to violations to the AMI security requirements, such as data overwrite protection, correct device scheduling with respect to limited resource and cyber bandwidth, authenticated and trusted communication, prioritized data delivery, fail-safe response, etc. The accuracy and usability of our presented tool were evaluated using an AMI testbed. We evaluate the scalability of SmartAnalyzer in different test configurations. We achieve significantly high scalability by applying the property level abstractions in the model. We observe that the running time of the constraint verification is within 20 seconds for a network of one million collectors.

REFERENCES

[1] (2008) Ami system security requirements. AMI-SEC Task Force. [Online]. Available: http://osgug.ucaiug.org/utilisec/amisec/

[2] (2008) The smart grid: An introduction. U.S. Department of Energy. [Online]. Available: http://energy.gov/oe/downloads/smart-grid-introduction-0

[3] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security Privacy*, vol. 7, no. 3, pp. 75–77, 2009.

[4] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel, "Multi-vendor penetration testing in the advanced metering infrastructure," in *26th Annual Computer Security Applications Conference (ACSAC)*, USA, 2010, pp. 107–116.

[5] M. A. Rahman, E. Al-Shaer, and P. Bera, "Smartanalyzer: A noninvasive security threat analyzer for ami smart grid," in *31st IEEE International Conference on Computer Communications (INFOCOM)*, 2012, pp. 2255–2263.

[6] Lontalk protocol specification. [Online]. Available: http://www.enerlon.com/JobAids/Lontalk%20Protocol%20Spec.pdf

[7] (2010) Nistir 7628: Guidelines for smart grid cyber security. Smart Grid Interoperability Panel- Cyber Security Working Group. [Online]. Available: http://www.nist.gov/smartgrid/upload/nistir-7628_total.pdf

[8] R. Alimi, Y. Wang, and Y. R. Yang, "Shadow configuration as a network management primitive," in *ACM SIGCOMM Conference on Data communication*, 2008, pp. 111–122.

[9] X. Ou, S. Govindavajhala, and A. Appel, "Mulval: A logic-based network security analyzer," in *14th USENIX Security Symposium*, 2005, pp. 113–128.

[10] E. Al-Shaer, W. Marrero, A. El-Atawy, and K. Elbadawi, "Network configuration in a box: Towards end-to-end verification of network reachability and security," in *IEEE International Conference on Network Protocols (ICNP)*, NY, USA, 2009, pp. 107–116.

[11] P. Bera, S. Ghosh, and P. Dasgupta, "Policy based security analysis in enterprise networks: A formal approach," *IEEE Transactions on Network and Service Management*, vol. 7, no. 4, pp. 231–243, 2010.

[12] L. de Moura and N. Bjorner, "Z3: An efficient smt solver," in *14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.

[13] B. Dutertre and L. D. Moura. (2006) The yices smt solver. [Online]. Available: http://yices.csl.sri.com/tool-paper.pdf

[14] R. Nieuwenhuis and A. Oliveras, "On sat modulo theories and optimization problems," in *Theory and Applications of Satisfiability Testing (SAT)*, ser. Lecture Notes in Computer Science. Springer, 2006, vol. 4121, pp. 156–169.

[15] Ami smart grid testbed at unc charlotte. Http://www.cyberdna.uncc.edu/events.php.

[16] E. Al-Shaer and H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *23rd IEEE International Conference on Computer Communications (INFOCOM)*, 2004, pp. 2605–2616.

[17] S. Noel and S. Jajodia, "Attack graphs for sensor placement, alert prioritization, and attack response," in *Cyberspace Research Workshop of Air Force Cyberspace Symposium*, Shreveport, Louisiana, 2007.

[18] R. Dewri, N. Poolsappsi, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in *14th ACM conference on Computer and Communications Security (CCS)*, 2007, pp. 204–213.

[19] I. Kotenko and M. Stepashkin, "Attack graph based evaluation of network security," in *10th IFIP International Conference on Communications and Multimedia Security*, 2006, pp. 216–227.

[20] J. Homer and X. Ou, "Sat-solving approaches to context-aware enterprise network security management," vol. 27, no. 3, pp. 315–322, 2009.

[21] (2009) Security in the smart grid. ABB Group. [Online]. Available: http://www02.abb.com/db/db0003/db002698.nsf/0/832c29e54746dd0fc12576400024ef16/$file/paper_Security+in+the+Smart+Grid+(Sept+09)_docnum.pdf

[22] S. McLaughlin, D. Podkuiko, and P. McDaniel, "Energy theft in the advanced metering infrastructure," in *4th International Workshop on Critical Information Infrastructure Security*, 2009, pp. 176–187.

[23] Y. Wang, D. Ruan, J. Xu, M. Wen, and L. Deng, "Computational intelligence algorithms analysis for smart grid cyber security," in *Advances in Swarm Intelligence*, ser. Lecture Notes in Computer Science, 2010, vol. 6146, pp. 77–84.

[24] Z. Anwar, R. Shankesi, and R. H. Campbell, "Automatic security assessment of critical cyber-infrastructures," in *IEEE/IFIP International Conference on Dependable Systems and Networks*, 2008, pp. 366–375.

[25] Z. Anwar and R. H. Campbell, "Automated assessment of critical infrastructures for compliance to cip best practices," in *2nd IFIP WG 11.10 International Conference on Critical Infrastructure Protection*, Arlington, Virginia, 2008.

[26] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electrical power grids," in *ACM Conference on Computer and Communications Security (CCS)*, 2009, pp. 21–32.

Mohammad Ashiqur Rahman received his BSc and MSc in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2004 and 2007, respectively. He is currently pursuing his Ph.D. degree in Computing and Information Systems at the University of North Carolina at Charlotte (UNCC), USA. He is associated with the CyberDNA Research Center, UNCC as a research assistant. Before that, he was working as a Lecturer, Dept. of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. His primary research area includes information and network security, security management, distributed computing. Currently he is more focused on cyber-physical system security and performance optimization.

Ehab Al-Shaer is a Professor and the Director of the Cyber Defense and Network Assurability (CyberDNA) Center in the College of Computing and Informatics at University of North Carolina Charlotte. He received his MSc and Ph.D. in Computer Science from the Northeastern University (Boston, MA) and Old Dominion University (Norfolk, VA) in 1998 and 1994 respectively. His primary research areas are network security, security management, fault diagnosis, and network assurability. Prof. Al-Shaer edited/co-edited more than 10 books and book chapters, and published about 150 refereed journals and conferences papers in his area. Prof. Al-Shaer is the General Chair of ACM CCS 2009-2010 and NSF Workshop in Assurable and Usable Security Configuration, August 2008. Prof. Al-Shaer also served as a Workshop Chair and Program Co-chair for a number of well-established conferences/workshops in his area including IM 2007, POLICY 2008, ANM-INFOCOM 2008, ACM CCS 2010. He is the founder and co-chair of SafeConfig for many years (09 - 11). He also served as a member in the technical programs and organization committees for many IEEE and ACM conferences.

Padmalochan Bera has graduated with Ph.D. degree from Indian Institute of Technology, Kharagpur, India in the year 2011. He received BE and ME in Computer Science & Engineering respectively from Jadavpur University and West Bengal University of Technology, Kolkata, India, respectively. He is currently working as a Senior Research Scientist in Software Engineering and Security Research Group, Infosys Labs, Bangalore, India. Before that he was associated with the CyberDNA Research Center, University of North Carolina at Charlotte, USA as a post-doctoral Researcher. His current research area includes software test automation, preventive maintenance, functional security testing and verification of safety and security properties in control software. He has already published more than 15 research articles in reputed journals and top-tier conferences.