Abstract—Smart grids provide bi-directional communication between smart meters at user premises and utility provider for the purpose of efficient energy management through Advanced Metering Infrastructure (AMI). Recent studies have shown that the potential threats targeting AMI are significant. Despite the need of developing intrusion detection systems (IDS) tailored for the smart grid [4], very limited progress has been made in this area so far. Unlike traditional networks, smart grid has its unique challenges, such as limited computational power devices and potentially high deployment cost, which restrict the deployment options of intrusion detectors. However, smart grid exhibits behavior that can be accurately modeled based on its configuration, which can be exploited to design efficient intrusion detectors. In this paper, we show that AMI behavior can be modeled using event logs collected at smart collectors, which in turn can be verified using the specifications invariant generated from the configurations of the AMI devices. We model the AMI behavior using the fourth order Markov chain and the stochastic model is then probabilistically verified using specifications written in Linear Temporal Logic. Our model is capable of detecting malicious behavior in the AMI network due to intrusions or device malfunctioning. We validate our approach on a real-world dataset of more than two thousand meters collected at the AMI of a leading utility provider.

I. INTRODUCTION

Smart grid deployment initiatives have been witnessed in the recent past. The basic premise of moving to smart grid infrastructure is to manage energy efficiently. An important core network in a smart grid is Advanced Metering Infrastructure (AMI). AMI provides bi-directional communication between end devices at customers’ premises like smart meters and headend at utility provider’s office. Headend system can remotely configure, upgrade, and request meter reading etc. using the AMI. This inherent criticality and availability of the AMI makes it a high potential target for the large-scale attacks that can potentially cause major regional blackout.

Despite these facts, limited progress has been made so far in order to monitor and detect malicious behavior in the AMI. Recent studies, including those by the federal government, have shown that AMI is facing immense potential threats [4], [7], [8], [12], [13], which could affect the deployment growth of smart grid. These vulnerabilities were exploited to penetrate in the AMI. Thus, it enables an attacker to gain control of a number of nodes for nefarious purposes. To address the security challenges faced by the AMI, some efforts have been made to prevent these threats [2], [3], [5].

In AMI, meters communicate with smart collectors using various mediums and smart collectors communicate with headend system (and vice versa) using public networks. Unlike traditional networks, AMI has its own requirements which pose significant challenges for monitoring and intrusion detection because it may require capturing network traffic. First, sensor deployment in the meters is practically infeasible due to the limited computational power and space resources at this node [11]. Second, although some researchers have suggested the meter-based sensors [2], [5], [14], smart grid providers as well as vendors firmly avoid this option due to prohibitive cost increase associated with the large number of meter deployments. Therefore, most IDS proposals for AMI lack practical feasibility.

Since it is important to monitor both the meter-collector and collector-headend communication, deploying detection modules at the smart collector nodes seem to be the most sensible option, particularly if sufficient computational power and space is assumed. Moreover, AMI communication activity is by default logged at the smart collector thus it does not pose any extra burden. Although device configuration and log’s integrity is protected using headend-collector key-pairing, this AMI feature was never exploited for monitoring and characterizing the AMI behavior. AMI is a special purpose network and its traffic dynamics are often very low since it supports a limited number of protocols and it is configuration-driven. Moreover, similar devices from limited vendors are usually deployed. To exploit the limited behavior, simple specification-based intrusion detection techniques are proposed in the recent literature [2].

In this paper, we present a novel device configuration based stochastic model checking-based intrusion detection technique. We model the AMI infrastructure behavior using the logs generated at the smart collector. Specifications written in Linear Temporal Logic (LTL) are automatically generated from the a-priori known configurations of the AMI devices (smart meters and collectors). These LTL specifications, which represent the system behavior invariants, are then probabilistically verified using the stochastic model generated from the smart collector’s logs. The proposed technique exhibits high accuracy (detection rate > 95% and false alarms < 0.2%) and it can be easily deployed in the existing AMI of a smart grid. For experimentation and evaluation we use a real-world dataset of more than two thousand meters obtained as a result...
of our collaboration with a leading smart grid based utility provider.

The rest of the paper is organized as follows. Dataset is described in Section II. Section III discusses the statistical analysis of the logs collected at the smart collectors. Based on the analysis, a stochastic model is proposed in Section IV. Properties are specified using LTL language which is described in Section V. Attack model and evaluation of the proposed approach is shown in Section VI. Section VII concludes the work.

II. DATASET

In AMI, the real-world dataset is important since all of the contemporary work uses simulation which may not necessarily reflect the true behavior of the system under consideration. Therefore, it is very important to analyze a real-world dataset in order to reveal its true behavior. To this end, we use real-world dataset collected at an AMI of a leading smart grid based utility provider. A smart grid testbed is also established by the utility provider at our institute in order to conduct experiments locally in a controlled environment. It includes a monitoring/management node which is capable of monitoring and configuring the nodes in an AMI, also referred as Network Management System (NMS). The testbed also includes multiple smart meters and smart collectors, both of which are capable of bi-directional communication.

Since it is not practically feasible to capture network traffic, we work with the event logs generated at the smart collector. Since event logging is an inherent capability of a smart collector, it does not impose any extra burden on the smart collector. These logs are saved at smart collector for a while and then deleted in a cyclic manner. We collected logs of multiple smart collectors for a total period of two weeks in two sessions of one week each denoted by Log-1 and Log-2. Approximately more than two thousand meters were communicating with the smart collectors. The AMI infrastructure used devices from multiple vendors. Sample and anonymized basic configurations of the meters are shown in Table I. In the AMI under consideration, reporting mode for the meters was ‘push’ i.e., the meter will send scheduled readings by itself at the interval specified in the ‘Reporting Time’. ‘Usage Sampling’ tells the number of seconds after which a sample is taken. Smart collector is aware of its neighboring meters and the link through which they are connected.

Sample of log entries of a smart collector are shown in Table II. These entries are simplified and anonymized to show only the information (variables) required in this work. Each log entry has the event time stamp. Time stamp is followed by the source and destination id of the nodes involved in the communication event. Size of the communication is also logged in kilo bytes (KB). Lastly, the type of event is also logged. For example, 0 represents that it was a meter reading report. 1, 2 and 3 represents disconnect/reconnect, upgrade and load management communication, respectively.

III. STATISTICAL ANALYSIS

We analyzed a number of statistical properties of the log entries. One relevant property that provided us with interesting insights was the analysis of their temporal dependence. It can be intuitively argued that, as long as the log entries are produced by benign events, the log entries observed should exhibit a certain level of temporal dependence. In case of malicious behavior, perturbations in this dependence structure are flagged as anomalous. Therefore, the level of temporal dependence can serve as an important metric for modeling the log entries.

Autocorrelation measures the on-average temporal dependence between the random variables in a stochastic process at different points in time. For a given lag \( k \), the autocorrelation function of a stochastic process \( X_n \) (where \( n \) is the time index) is defined as:

\[
\rho[k] = \frac{E\{X_0X_k\} - E\{X_0\}E\{X_k\}}{\sigma_{X_0}\sigma_{X_k}},
\]

where \( E\{\cdot\} \) represents the expectation operation and \( \sigma_{X_k} \) is the standard deviation of the random variable at time lag \( k \).

Figure 1(a) shows the autocorrelation function plotted for log entries. For both the logs, a certain level of temporal dependence can be easily observed at small lags. This correlation decays in time and eventually drops down to a negligible value. Temporal dependence is present for two reasons. First,
meters respond to the smart collector requests in a short period of time. Second, regular requests and reports are seen thus justifying the homogeneity.

It is well-known that a decaying temporal dependence structure can be accurately modeled using markov chains [9]. To identify the order of correlation presence in the log entries random process, we define a markov chain based stochastic model as follows: Let the log entry tuple at discrete time instance $n$ represents the realization of a random variable derived from a stochastic process $X_n$. This process is a markov chain if it satisfies the markov property i.e., probability of choosing a next state is only dependent on the current state of the markov chain.

In the present context, we can define a markov chain model $X_n$ for log by treating each unique log entry tuple individually and assigning them to multiple non-overlapping bins. Here we assign each unique tuple to each bin. Therefore, the number of bins will be dependent on the number of unique log entry tuples. Each bin then represents a state of the markov chain, while the set of all bin indices $\psi$ is its state space. Based on this state representation, we can define a 1-st order markov chain, $X_n^{[1]}$, in which each bin represents a state of the random process. Similarly, an $l$-th order markov chain, $X_n^{[l]}$, can be defined in which each state is an $l$-tuple $(i_0, i_1, \ldots, i_{l-1})$ representing the values taken by the random process in the last $l$ time instances. In this case the occurrences of $l$-tuple will be counted for calculating the probability. This will increase the size of state space $\psi$ since different combinations of $l$-tuple can be observed.

To explore it further, we use the conditional entropy based measure [9] for the log entries for different markov chain orders. This measure tells us the order of markov chain which provides most of the information regarding the next time instance. Conditional entropy, $H(B|A)$, of two discrete random variables $A$ and $B$ characterizes the information remaining in $B$ when $A$ is already known. If $A$ and $B$ are highly correlated, most of the information about $B$ is communicated through $A$ and $H(B|A)$ is small. Otherwise, $H(B|A)$ assumes a high value, which means that most of the information about $B$ is not given by $A$.

The transition probability matrix of the 1-st order markov chain $P^{[1]}$ can be computed by counting the number of times the state $i$ is followed by state $j$. The resulting $[\psi^{[1]}]$ histograms can be normalized to obtain the state-wise transition probability mass functions as the rows of $P^{[1]}$.

We can find the conditional probability of the 1-st order markov chain as:

$$H^{[1]} = - \sum_{i \in \psi^{[1]}} \pi_i^{[1]} \sum_{j \in \psi^{[1]}} p_{ij}^{[1]} \log_2 \left( p_{ij}^{[1]} \right),$$

where $\pi_i^{[1]}$ is the average probability of being in state $i$ which is computed by counting the total number of times each state is visited and then normalizing the frequency histogram.

The measure $H^{[1]}$ defines how much average information is remaining in log entry $X_n$ when it is calculated using log entry $X_{n-1}$. It is easy to observe that $H^{[1]} = H^{[2]} = \ldots = H^{[l]}$, as each older entry can either be independent of, or provide some information about the present entry. The number of previous entries required to accurately predict the next entry can then be determined by plotting $H^{[1]}$ as a function of the markov chain order, $l = 1, 2, \ldots$. The order at which the conditional entropy saturates defines the total number of previous entries which have conveyed as much information of the present entry as possible.

It can be clearly seen in Figure 1(b) that log entries exhibit a decaying trend over higher order markov chain. It can be seen that it exhibits an exponential decay trend until the 4-th order and drops to a negligible value i.e., $\leq 0.2$. Therefore, we model using the fourth order markov chain since it gives enough information and increases predictability.

IV. Model

Since our model is built on the logs of smart collectors, we first look at the format of a log entry. It can be represented as:

$$t, s_{id} , d_{id} , s_z , t_y$$

where $t$ represents the time stamp at which the event was observed for which the entry is logged. $s_{id}$ and $d_{id}$ refers to the source and destination, respectively, of the communication observed. It can either be the meter or smart collector. Size of the communication is denoted by $s_z$. The type of communication is defined by $t_y$. Therefore, in this model we use the above information. We can encode the state of the network with the following characteristic function:

$$\sigma : s_{id} \land d_{id} \land s_z \land t_y \rightarrow \{true, false\}$$

The function $\sigma$ encodes the state of the network by evaluating to true whenever the parameters used as input to the function correspond to the log entry in the smart collector. If the AMI observes 5 different log entries, then exactly 5 different assignments to $\sigma$ function will result to true. Since each smart collector is independent, we learn the markov model for each smart collector separately.

Suppose we have sequence $S = \sigma_1, \sigma_2, \ldots, \sigma_n$, where $\sigma$ represents a state as shown in Equation 4. Since our statistical analysis showed that conditional entropy is negligible at 4-th order, therefore, we use 4-th order markov chain. A finite state machine having directed graph can be learned from the given sequence $S$. Each state in the graph at time $i$ will be represented by a tuple of 4 i.e., $(\sigma_{i-3}, \sigma_{i-2}, \ldots, \sigma_i)$. Therefore, it can be realized as $s_i$ in the finite state machine.

Algorithm 1 explains the learning of a markov model from the given sequence of log entries. It initializes an empty graph then a directed edge from $s_i$ to $s_j$ is created, if the directed edge does not exist already. However, if $s_i$ does not exist in graph, then a node is also created for $s_i$. This process
Algorithm 1: Learn Markov Model

Data: Sequence $S$
Result: Finite state machine based on 4-th order Markov Model

Initialize empty Graph;
$S = \{\sigma_i | \forall \sigma_i \in \Sigma \}$;
$\forall \sigma_i, \Pr(\sigma_i) > 0$;
while $S \neq \phi$ do
    Slide window by one $\sigma$ at instance $i$;
    Pick $s_i \in S$;
    if $s_i \in$ Graph then
        Make directed edge from $s_{i-1}$ to $s_i$;
    else
        Create node $s_i$ in Graph;
        Make directed edge from $s_{i-1}$ to $s_i$;
    end
end

keeps repeating until $S$ is empty. Once the state machine is created, it is easy to calculate the transition probability matrix. For each state $s_i$ in graph, $\sum_{\forall \sigma_i \in \Sigma} \tau(s_i, \sigma_i) = 1$.

Since a log entry $\sigma$ is a conjunction of different variables, total possible combinations can exceed and may require a lot of processing power. However, it can be calculated for each network under consideration. In our case study, 10 bits were assigned to $s_{id}$ and $d_{id}$, 8 bits for $sz$ and 3 bits for type $ty$ of communication. Therefore, the possible number of $\sigma$ are $2^{10} \times 2^{10} \times 2^8 \times 2^3$ which is a relatively large number. Since the model treats each smart collector’s log separately, either source or destination of each log entry will be fixed to the id of that particular smart collector. Moreover, a smart collector can only be connected to its neighboring meters/devices. In our case study, the smart collector was connected to 8 other devices. Therefore, the number of $\sigma$ reduces to $1 \times 8 \times 2^8 \times 2^3$, which is relatively smaller. Since 4-th order markov model is being used, possible combinations of four $\sigma$ can yield to a lot of states. To this end, the proposed algorithm only takes the combinations which are observed in the sequence $S$ and only keeps the edges which are observed since all the combinations are not possible due to configuration, thus reducing the size of transition probability matrix.

V. Properties Specification for Model Checking

Since the proposed model is based on markov chain and exhibits a temporal dependence, we define properties in Linear time Temporal Logic (LTL) [1]. Unlike traditional model checking, stochastic model checking allows you to check that with what probability the property is satisfied by the model. These probabilities can be thresholded in order to accommodate the unseen behavior up to a certain extent.

Let $\phi$ be the LTL formula over the alphabets $\Sigma$. An LTL formula can be satisfied for a sequence of alphabets $s$ which is a state definition in our case having $s = \sigma_1, \sigma_2, \ldots, \sigma_n$ where $\sigma_i \in \Sigma$. Therefore, the probabilistic LTL can be defined as:

$$\phi ::= P_{\text{exp}}(\varphi), \quad \varphi \in \{\geq, >, \leq, <, \gamma, =\};$$

$$p \in [0, 1]; \quad \varphi \in \text{LTL}$$

LTL properties can be verified with the markov chain model built in the earlier section. For example, if a configuration parameter defines the sampling rate and report size, a property can be written that whenever a report request is received the reply should have this particular size. Temporally it can be stated that given the system is in ‘request’ state, the next expected state is ‘reply_with_size_h’. We wrote a small parser which reads the configuration and generates the properties in LTL format for the tool which can be verified against the model using PRISM model checker tool [10].

The properties can be derived from the configurations and the security control guidelines such as NISTIR 7628 by NIST. Since the configurations shown are related to reading report, below we show some examples of the properties derived from the configuration. Let $\gamma$ be the number of meters associated with a smart collector. One basic example is that whenever a report reading request is generated, a meter should respond with a report. It can be formulated as:

$$P_{\geq d_1}(\square(rreq_i \rightarrow \bigcirc rrep_i))$$

where $rreq_i$ and $rrep_i$ represents the reading request and reading report, respectively, for meter $i$. $\square$ and $\bigcirc$ represents the always and next operator, respectively. $d_1$ is used as a probability threshold that this property should be satisfied with the probability greater than or equal to $d_1$.

Similarly, it can also be defined that the report generated should have a size with in the limits defined since sampling rate is fixed. It can be formulated as:

$$P_{\geq d_2}(rrep_i \rightarrow rsz_i), \quad 1 \leq i \leq \gamma$$

where $rsz_i$ denotes the report size for meter $i$. However, $rsz_i \in sz_i$ which is a valid report size set for meter $i$. Moreover, Equation 5 and 6 can be combined to show the temporal behavior that whenever a reading request is generated, it is followed by the reading reply which has a valid size.

Moreover, a meter should not send the reading report twice in the next $T_1$ consecutive time periods. It can be formulated as:

$$P_{\leq d_3}(\square rrep_i \rightarrow (\neg rrep_i \cup t_i \geq T_1))$$

where $d_3$ is thresholded with $\leq$ that the probability of seeing such a behavior should be less than $d_3$, $t_1$ is a counter which observe values in the range $\{1, 2, \ldots, T_1\}$.

To avoid flooding the smart collector with reports from multiple meters at the same time, the associated meters were configured to have different reporting intervals. Therefore, smart collector will not receive consecutive reports from multiple meters in the consecutive $T_2$ time periods. It can be formulated as:

$$P_{\leq d_4}(\square rrep_i \rightarrow (\neg rrep_j \cup t_2 \geq T_2), \forall j=1$$

this prevents the multiple meters from sending the reports after each other for consecutive $T_2$ time periods.
VI. EXPERIMENTATION & EVALUATION

Before discussing the experimentation and evaluation, we first discuss the attack model.

A. Attack Model

Our focus is on the large scale attacks which include compromising a large number of meters to cause a major blackout in the area. These attacks are, but not limited to, spoofing, denial of service, distributed denial of service, scanning, penetration, evasion, mimicry etc. Since the infrastructure exhibits a deterministic behavior and is homogenous in nature; spoofing, mimicry and evasion techniques can inject similar traffic without being detected.

Traditional attacks like denial of service will be detected by the property 7 since it will create multiple entries in a time window shorter than $T1$. Similarly, distributed denial of service will cause property 8 to be invalid since it will cause multiple sources to create log entries in a time window shorter than $T2$. To countermeasure mimicry and evasion attacks, we propose a simple randomization module which introduces a notion of randomness in the meter behavior while staying deterministic for the smart collector. It is discussed in the subsequent section.

To evaluate the proposed scheme, we generate attacks in a controlled environment in the smart grid testbed. We launch scanning, DoS, evasion, mimicry and data injection traffic. For scanning and DoS, we generate low rate attacks i.e., 0.1 pkts/sec to 1 pkts/sec. For evasion, mimicry and spoofing, we placed a switch in between the meters and collector; and attached an attack machine to it. We wrote a simple program which uses the same configuration as of a meter and generate similar reading reports in the same format. In order to be evasive, less than 5% of the total generated traffic by the machine was the injected traffic. Injected traffic includes malicious commands like random file uploads, requesting reports at irregular intervals (requesting report itself is not an attack), administrative commands without proper authorization and failed authorization attempts. Malicious logs were mixed into the real-world logs collected at an AMI of the utility provider for the purpose of the accuracy evaluation.

B. Robustness against Evasion and Mimicry Attacks

Since evasion and mimicry attacks leverage the known behavior of the network, we randomize one of the configuration parameters which makes it difficult for the attacker to guess. Scheduled readings have a specific time interval specified in the configuration. We randomize the reporting interval parameter. Let $int_i$ denote the set containing all the possible valid configuration values for the report interval for meter $i$, where $int_i^k$ is the randomly selected configuration value at time $t$ for the meter $i$. To introduce a certain level of randomization, a meter should have enough possible valid configuration values to pick from. It can be formulated as:

$$|int_i| \geq \theta$$  \hspace{1cm} (9)

The greater the $\theta$, the higher the randomization. To select a value randomly from the given set, we use a simple hash function since it is effective and computationally inexpensive too.

$$int(t + 1) = H(k_i, t, int_i^k) \mod l + 1 \hspace{1cm} (10)$$

where $k_i$ denotes the pre-shared key between the meter $i$ and the smart collector, $t$ represents the time and $l$ is the size of possible values that is used to keep the value in the given range. The hash function takes the key and the previously selected reporting interval to introduce unpredictability for the attacker and predictability for the smart collector. Reporting interval will be chosen every time before sending the report. Smart collector will do the same computation in order to verify if the report was expected at the given time or not. Calculated time interval will be used in the property 7 to identify any non-compliance behavior. In our experiment, we implemented this simple module on the NMS machine which was capable of remotely configuring the meters. The module was written in the C language using the NMS API to configure the time interval for each report.

C. Accuracy Evaluation

The basic premise of our work is to develop a model using the logs of smart collector and verify the configuration properties written in LTL. If a property is verified against the model with probability $p > d$, where $d$ is set as a threshold, we say that it is normal, otherwise, it is anomalous. Threshold $d$ was learnt separately for each property by noting the property verification probability using the benign logs model (without attack). First we analyze that how well the model represents the system behavior. Please note that this temporal behavioral analysis was done without the randomization module and attack traffic. The model checking was conducted using a probabilistic symbolic model checker tool PRISM [10].

1) Model Accuracy: In order to determine whether the model actually represents the logs or not, we conduct a model accuracy experiment as shown in Figure 2. It does not include the attack traffic or property verification. We divide the benign logs into two halves. The first half was used to build a model using the Algorithm 1. Second half was used as the test dataset and the model was applied on it for the prediction. For each step, the current state was learned and based on the current
state, the next state was predicted using the model i.e., states having higher probabilities in probability mass function (pmf) were predicted. Figure 2 shows the prediction using the fourth order model. The red cross marker denotes the actual states observed in the test log. However, blue dashed line represents the prediction of the model. It can be observed that the model predicts the future states with high accuracy though few false predictions (less than 2%) were encountered as well. These false predictions were observed as a result of the unseen behavior since the benign log was divided into two halves. As a result, few lower probability states were not observed after a certain state s (i.e., tuple history of order four) present in the first half, thus yielding to false prediction.

2) Detection Accuracy: Mixed data was used to calculate the detection accuracy which had attack and benign logs both. Since the attack logs were generated separately in a controlled environment, the time stamps of the attack logs were adjusted by a fixed constant to have the same time window as of normal/benign logs. Allowed communication type ty values were used for the attack logs since other communication types do not exist in the model and can be detected easily. For attack detection, model learning was done in a continuous online learning fashion using a sliding window approach. The size of the sliding window was kept to one hour and the sliding window interval was set to one minute. Model was learnt separately for each smart collector in the dataset and detection results were averaged. Figure 3(a) shows the detection accuracy achieved by the presented model. It can be seen that high detection rate of more than 95% was achieved with a negligible false alarm rate of approximately 0.2%. Please note that these rates are on the entire dataset collected. Receiver Operating Characteristics (ROC) curve is generated by changing the verification probability (threshold) of configuration-based LTL properties which were verified against the model built using the mixed data logs. It is intuitive that the attack activity does not follow the state transitions as allowed in the temporal properties. Therefore, the higher volume of attack activity as compared to the benign activity can be easily detected even with the loose probability verification threshold. On the other hand, the lower volume of attack activity will be detected by the strict probability verification threshold. Figure 3(b) shows the effect of verification probability for multiple degree of DoS attacks. It can be observed that the higher attack rate is detectable even with a lax verification probability threshold.

Algorithm 1 was implemented in Java. The run-time complexity is in hundreds of milliseconds (~300ms) with a memory requirement of few kilobytes (~20KB). The complexity was measured using HPROF [6] tool.

VII. CONCLUSION

In this work, we present a fourth order markov chain based model for intrusion detection since it incurs lower conditional entropy and higher prediction accuracy. The novelty of the model lies in the configuration-based stochastic modeling of the AMI using the logs collected at smart collector and device configuration. Moreover, considering the challenges of AMI network, the proposed approach is practical since it does not require high computation power and memory in the field. However, it can also work offline in the substation by pulling the smart collector logs from the area. It also does not need to be trained, unlike traditional intrusion detectors.

ACKNOWLEDGMENTS

This work is a result of joint collaborative effort with a leading smart grid based utility provider. The work was done as part of the urban city project initiated by the utility provider which monitors several buildings for data analysis.

REFERENCES