IEICE TRANS. INF. & SYST., VOL.E100–D, NO.8 AUGUST 2017

PAPER Special Section on Information and Communication System Security

A Novel RNN-GBRBM Based Feature Decoder for Anomaly Detection Technology in Industrial Control Network

Hua ZHANG[†a)], ***Member***, **Shixiang ZHU**[†], **Xiao MA**[††], **Jun ZHAO**[†††], ***and*** **Zeng SHOU**[††††], ***Nonmembers***

SUMMARY As advances in networking technology help to connect industrial control networks with the Internet, the threat from spammers, attackers and criminal enterprises has also grown accordingly. However, traditional Network Intrusion Detection System makes significant use of pattern matching to identify malicious behaviors and have bad performance on detecting zero-day exploits in which a new attack is employed. In this paper, a novel method of anomaly detection in industrial control network is proposed based on RNN-GBRBM feature decoder. The method employ network packets and extract high-quality features from raw features which is selected manually. A modified RNN-RBM is trained using the normal traffic in order to learn feature patterns of the normal network behaviors. Then the test traffic is analyzed against the learned normal feature pattern by using osPCA to measure the extent to which the test traffic resembles the learned feature pattern. Moreover, we design a semi-supervised incremental updating algorithm in order to improve the performance of the model continuously. Experiments show that our method is more efficient in anomaly detection than other traditional approaches for industrial control network.

key words: *anomaly detection, industrial control network, GBRBM, RNN-GBRBM, osPCA, semi-supervised*

1. Introduction

In recent decades, industrial control systems have been well researched and extensively developed extensively with a high rate, and widely applied in various fields, such as oil, water, traffic and even nuclear. However, the development of network intrusion technology has surpassed the security study in industrial control systems at a terrifying pace. For achieving the decentralized management and the remote control, more and more control systems connect to open networks. But most of these industrial control networks (ICNs) were not designed with security constraints in the primary system design [1], and are vulnerable to network attacks nowadays. The critical infrastructures in an industrial control network, especially control systems, are the most potential targets of network attacks from cyber-terrorist, malicious hacker and disgruntled employees [2]. Disruption from any of them would cause a tremendous loss of production cost if we do not act on these in time, and even sometimes it may cause serious environmental damage and endanger the public safety.

In response to the threats of anomalous behaviors in industrial control network, anomaly detection techniques are becoming a hot field in industrial network security research. In general, there are two basic modeling theories for building a Network Intrusion Detection System (NIDS). One is signature based or rule based NIDS, which detects intrusion by observing events in the system and applying a set of rules that lead to the decision regarding whether a given pattern of activity is suspicious or not. The other one is anomaly based NIDS, which compares events against an established baseline, and the baseline will identify what is "normal" for this network [3]. Signature and anomaly based NIDS are similar in terms of conceptual operation and composition. The main difference between these methodologies is inherent in the concepts of "attack" and "anomaly". An attack can be defined as "a sequence of operations that puts the security of a system at risk". An anomaly is just "an event that is suspicious from the perspective of security". Based on this distinction, we point out the main advantages and disadvantages of each NIDS type as follows.

Signature based NIDS (S-NIDS) provides very reliable detection results for specified well known attacks with a low false positive rate (or FP, events erroneously classified as attacks). However, S-NIDS is not capable of detecting new, unfamiliar intrusion, even if it is built as minimum variant of already known attacks. On the contrary, the most powerful ability of anomaly based NIDS is detecting unknown intrusion events as well as "zero day" attacks, because they model the normal operation of a network and detect deviation from them. However, the high false positive rate in anomaly based NIDS is a serious problem [4].

Given the promising capabilities of anomaly based network intrusion detection system (A-NIDS), this approach is currently a principal focus of research and development in the field of industrial control network intrusion detection. There are a lot of effective techniques have been proposed for anomaly based detection, which can be divided into three types: statistical anomaly detection, machine

Manuscript received August 23, 2016.
Manuscript revised January 5, 2017.
Manuscript publicized May 18, 2017.

[†]The authors are with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, 100876 China.
[††]The author is with Data Network of power dispatch and safety protection of electric power supervisory control system, State Gird Jiangsu Electric Power Company, Shanghai Road No 215 Nanjing Jiangsu 210024, China.
[†††]The author is with Safety protection of electric power supervisory control system, NARI Group Corporation, No. 19 Chengxin Rd, Jiangning, Nanjing Jiangsu 211000, China.
[††††]The author is with Dispatching Data Network and Secondary Safty Protection, Electric Power Science Research Institute of State Gird Jiangsu Electric Power Company, Jiangning Paweier Road No 1 Nanjing Jiangsu 211103, China.
a) E-mail: zhanghua_288@bupt.edu.cn (Corresponding author)
DOI: 10.1587/transinf.2016ICP0005

learning based anomaly detection and data mining based anomaly detection [3]. Lots of real NIDSs based on these techniques had a good performance in the past decades, such as Next-Generation Intrusion Detection Expert System (NGIDES) [5] and Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) [6], which were both developed by Stanford Research Institute (SRI). However, an increasing number of undetected attacks have arisen in industrial control network as of late, there are three factors leading to the situation [7]–[9]:

- Attacks have become more sophisticated than before. The behavior of attack is interconnected with multiple protocol fields and network environment.
- With multi-strategy sequential attacks being more and more frequently, recognizing anomaly pattern within a single network packet becomes unrealistic. It is too hard to detect intrusion by inspecting single packet content for a conventional A-NIDS.
- With the development of network technology, attacks are updating at an ever faster speed. Conventional A-NIDSs performed poorer with unseen attacks. We need keep training models with latest network data in order to maintain the performance of detection.

Focusing on above factors, we proposed a modified architecture of machine learning based A-NIDS, including a novel RNN-GBRBM based feature decoder. The rest of this paper is organized as follows. Section 2 gives a brief overview of models, which are involved in the feature decoder, including Restricted Boltzmann Machines (RBMs) and RNN-RBM. The architecture that system used are shown in Sect. 3. Sections 4 and 5 elaborate on the decoder, classifier we have adopted, and a novel semi-supervised incremental updating algorithm. Section 6 is dedicated to the description of experiments and discussion of their results. At the end, we give our summary in Sect. 7.

2. Overview of Models

2.1 Restricted Boltzmann Machine

Restricted Boltzmann machine (RBM) is a stochastic neural network, which is powerful enough to represent complicated distribution [10]. Mostly, RBMs have been used as generative models of many types high-dimensional data including labeled or unlabeled images that represent speech, bags of words that represent documents, and user ratings of movies. Also, recently some NIDS researches has been dipping their toe into RBMs [11].

As shown in Fig. 1, the standard type of RBMs has n binary-valued hidden units and m binary-valued visible units, and consists of a matrix of weights $W = (w_{ij})$ associated with the connection between hidden unit h_j and visible unit v_i, as well as bias weights (offsets) b_{h_j} for the hidden units and b_{v_i} for the visible units.

A joint configuration (v, h), the visible and hidden units, has an energy given by Eq. (1).

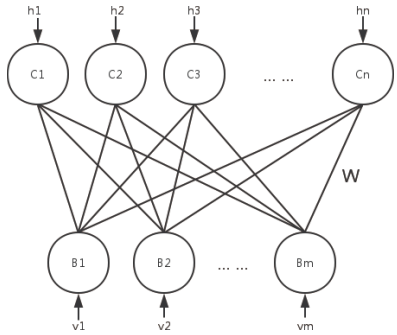


Fig. 1 Structure of a restricted Boltzmann machine

$$E(v,h) = \sum_{j=1}^{n}\sum_{i=1}^{m} w_{ij}h_j v_i - \sum_{i=1}^{m} b_{v_i}v_i - \sum_{j=1}^{n} b_{h_j}h_j \tag{1}$$

The network assigns a probability to every possible pair of a visible unit and a hidden vector via a energy function as shown in Eq. (2).

$$p(v,h) = \frac{1}{Z}e^{-E(v,h)} \tag{2}$$

Where Z denotes the "partition function" which is given by summing over all possible pairs of visible and hidden vectors, as shown in Eq. (3).

$$Z = \sum\nolimits_{v,h} e^{-E(v,h)} \tag{3}$$

Because of the specific structure of RBMs, visible and hidden units are conditionally independent given one-another, as shown in Eqs. (4) and (5).

$$p(v\,|\,h) = \prod\nolimits_{i=1} p(v_i\,|\,h) \tag{4}$$

$$p(h\,|\,v) = \prod\nolimits_{j=1} p(h_j\,|\,v) \tag{5}$$

In the commonly studied case of using binary units, where $v_j,\ h_i \in \{0,1\}$, we obtain a probabilistic version of the usual neuron activation function [12], as shown in Eqs. (6) and (7), from Eqs. (1), (2).

$$p(v_i = 1\,|\,h) = \sigma\left(b_{v_i} + \sum\nolimits_{j=1} h_j w_{ij}\right) \tag{6}$$

$$p(h_j = 1\,|\,v) = \sigma\left(b_{h_j} + \sum\nolimits_{i=1} v_i w_{ij}\right) \tag{7}$$

Where $\sigma(\bullet)$ is the logistic sigmoid activation function $1/(1+e^{-x})$. The independence between the variables in visible or hidden layer makes Gibbs sampling especially easy: Instead of sampling new values for all variables subsequently, the states of all variables in either layer can be sampled jointly. Thus, Gibbs sampling can be performed in just two steps: sampling a new state h for the hidden units based on $p(h\,|\,v)$ and sampling a state v for the visible layer based on $p(v\,|\,h)$ [10]. This is also referred to as block Gibbs sampling.

The parameters of RBMs, which include Θ =

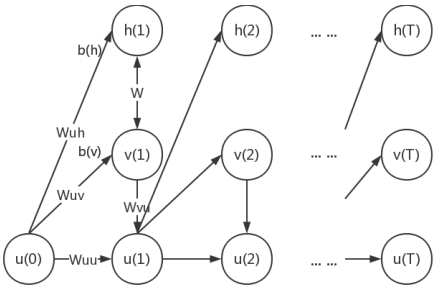


Fig. 2 Structure of a RNN-RBM

(W, b_h, b_v), are obtained by training. Training means finding the values of these parameters that correspond to desirable values for the energy, usually such that the energy is minimized. Thus, a possible training strategy may aim at minimizing the log-likelihood of the training data that is estimating its gradient with respect to the model parameters. While an exact computation is intractable, the gradient can be estimated using a method called contrastive divergence (CD) [11]. CD learning is highly successful and is becoming the standard learning, only runs block Gibbs sampling for k (usually $k = 1$ [13]) steps to approximate the second term in the log-likelihood gradient from a sample from the RBM distribution [14].

2.2 RNN-RBM

The RNN-RBM is an energy-based model generalized from RTRBM [15] for density estimation of temporal sequences, where the feature vector $v^{(t)}$ at time window t may be high-dimensionality. The structure of RNN-RBM is shown in Fig. 2. It allows to describe multimodal conditional distribution of $v^{(t)} \mid \mathrm{A}^{(t)}$ where $\mathrm{A}^{(t)} \equiv \{v_\tau \mid \tau < t\}$ denotes the sequence history at time t, via a series of conditional RBMs whose parameters $b_v^{(t)}$, $b_h^{(t)}$ (as shown in Eq. (8) and (9)) depend on the output of a deterministic RNN with hidden units $u^{(t)}$.

$$b_v^{(t)} = b_v + W_{uv} u^{(t-1)} \tag{8}$$

$$b_h^{(t)} = b_h + W_{uh} u^{(t-1)} \tag{9}$$

For simplicity, we denote the RBM parameters as W, $b_v^{(t)}$, $b_h^{(t)}$ and a single-layer RNN whose hidden units $u^{(t)}$ are only connected to their direct predecessor $u^{(t-1)}$ and to $v^{(t)}$ Eq. (10) [16].

$$u^{(t)} = \tanh(b_u + W_{uu} u^{(t-1)} + W_{vu} u^{(t)}) \tag{10}$$

In the Sect. 5, we will introduce the training algorithm of a modified RNN-RBM in our system in details.

3. System Architecture

As mentioned above, Our NIDS architecture is based on machine learning based A-NIDS schemes. More specifically, the architecture is based on clustering and outlier detection strategy, which works by grouping the observed data into clusters, according to a given similarity or distance measure. The process most commonly used for this strategy includes following four steps.

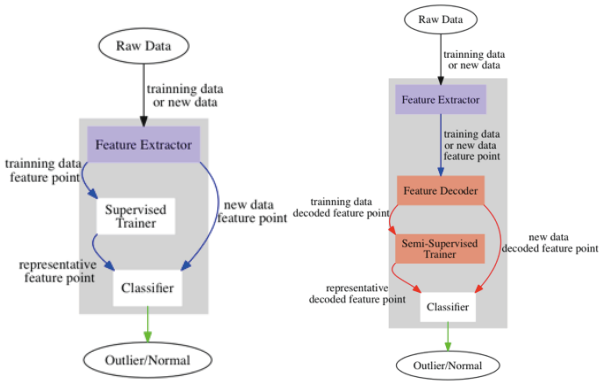


Fig. 3 (a) The architecture of conventional clustering & outlier strategy NIDS. (b) The modified architecture of the NIDS

- Features are extracted from the acquired data.
- Selecting a representative feature vector for the cluster of normal data.
- The feature vector of each new data is classified as belonging to the given cluster according to the proximity to the corresponding representative feature vector.
- If the vector doesn't belong to the cluster, then it is the outlier and represents an anomaly in the detection process.

To solve the problems mentioned in Sect. 1, we made two modifications on the architecture of conventional clustering & outlier strategy NIDS. First, a novel feature decoder is introduced. In order to analyze continuous time series of network data with highly complex structure, the RNN-GBRBM (modified RNN-RBM) is adopted. Combining the desirable characteristics of RNNs and RBMs have proven to be non-trivial [16] because RNN enables the network to have a simple version of memory with very minimal overhead and allows more freedom to describe the temporal dependencies involved [17], as well as because RBM can capture complicated, high-order correlations between the activities of hidden features [18] and provide a closed-form representation of the distribution underlying the observations [10]. Moreover, a semi-supervised incremental updating algorithm, which is appropriate for training the decoder and updating the parameter of classifier, is proposed. The algorithm provides equivalent detection performance as the conventional supervised training method while decreasing reliance on manual inspection. In addition, the algorithm also dynamically updates the profiles of normal network data. This is a very important property, since normal network data profiles may change over time.

The comparison between the architecture of conventional clustering & outlier strategy NIDS and the modified architecture of the NIDS is shown in Fig. 3. The arrows represent different types of streaming of network data. The black arrows indicate raw data, the blue arrows indicate

feature vectors generated by Feature Extractor, the red arrows indicate decoded feature vectors which are processed from feature vectors, and the green arrows indicate the output result.

In this paper, given a sequential industrial control network data, denoted as $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(T)}\}$, we suppose to build a model λ when $\lambda(x_{test} \leq \tau)$. Considering x_{test} as an outlier or anomaly, we define that x_{test} is unseen network data and τ is a preset threshold. According to the designed architecture, λ becomes to Eq. (11), Where ν is the model of classifier, and μ is the model of feature decoder (i.e. RNN-GBRBM), $\mu(x)$ denotes the decoded feature vectors.

$$\lambda(x) = \nu(\mu(x)) \tag{11}$$

4. Feature Decoder

4.1 Feature Extraction

In order to construct the input of trainer and classifier, feature extraction is necessary. Feature extraction starts from an initial set of measured raw network data and builds derived values (features) in a unified form that is intended to be informative and non-redundant, facilitating the subsequent training, decoding and classification. The usual way of feature extraction is picking or combining some of the protocol field values or network parameters as features (vectors) via human analysts [2]–[6], [11], [19]–[21].

We define the extracted feature of raw network data as Eq. (12). Where x_t is the t-th network data in a network data sequence, d_{raw} is the dimensionality of the feature vector $\vec{v}(x_t)$, and $v_k(x_t)$ is a real value which indicates the k-th feature of x_t.

$$\vec{v}(x_t) = [v_1(x_t), v_2(x_t), \ldots, v_{d_{raw}}(x_t),] \tag{12}$$

However, it is sometimes better to increase the dimensionality of features with another technique, this still happens when data is extremely complex. Simple extracted features are not enough to represent the information of raw data, and to derive good results. We proposed a novel idea of using RNN-GBRBM as feature decoder, so as to make features more informative and convey more information to the classifier. For concreteness, thanks to some traits of GBRBM, the feature decoder increases the dimensionality of extracted features v and converts features from real-valued vectors into binary vectors.

4.2 Gaussian-Bernoulli Boltzmann Machine (GBRBM)

As described in Sect. 3, we need make some modification on the original RNN-RBM to joint our settings. The most major modification on RNN-RBM is reforming RBM to Gaussian-Bernoulli RBM (GBRBM). As shown in Eqs. (5) and (6), the random binary variables are assumed for the inputs of the RBM [10]. This assumption becomes a critical issue in considering real applications such as network data. Reference [22] proposed that RBMs can extend harmoniums into the exponential family, such as Gaussian, which could make them much more widely applicable. Hence, RBM can be extended to GBRBM which can take real-valued variables as inputs in the visible units, and obtain an efficient classification feature (decoded feature) as outputs in the hidden units [14].

The GBRBM has visible units with real-valued variables and binary hidden units. The energy function of the GBRBM is defined as Ref. [12], and is shown in Eq. (13). Where b_{v_i} and b_{h_j} are bias corresponding respectively to visible and hidden units, w_{ij} is the connecting weights between the visible and hidden units and σ_j is the standard deviation associated with Gaussian visible units v_i. Conditional probabilities for visible and hidden units are shown in Eq. (14) and Eq. (15). Where $\mathrm{N}(\bullet \,|\, \mu, \sigma^2)$ denotes the Gaussian probability density function with mean μ and standard deviation σ.

$$E(v,h) = \sum_{j=1}^{n}\sum_{i=1}^{m} w_{ij}h_j\frac{v_i}{\sigma_j} - \sum_{i=1}^{m}\frac{(v_i - b_{v_i})^2}{2\sigma^2} - \sum_{j=1}^{n} b_{h_j}h_j \tag{13}$$

$$p(v_i = v \,|\, \vec{h}) = N\left(\vec{v} \,\middle|\, b_{v_i} + \sum\nolimits_j h_j w_{ij}, \sigma_i^2\right) \tag{14}$$

$$p(h_j = h \,|\, \vec{v}) = f\left(b_{h_j} + \sum\nolimits_i w_{ij}\frac{v_i}{\sigma_i^2}\right) \tag{15}$$

In this case, the GBRBM takes an extracted feature $\vec{v}$ as input in the visible units and obtains a decoded feature $\vec{h}$ in the hidden units. $\vec{h}$ is defined as Eq. (16). Where x_t is the t-th network data in a network data sequence, $d_{decoded}$ is the dimensionality of the decoded feature vector $\vec{h}(x_t)$, and $h_k(x_t)$ is a binary value which indicates the k-th code of decoded features.

$$\vec{h}(x_t) = [h_1(x_t), h_2(x_t), \ldots, h_{d_{decoded}}(x_t)] \tag{16}$$

This procedure of transforming extracted features into decoded features, called "decoding" in this paper, is basically a half step of block Gibbs sampling, i.e. sampling a new state h as a decoded feature for the hidden units based on $p(h \,|\, v)$ with a well-trained GBRBM. On the contrary, the other half step of block Gibbs sampling, i.e. sampling a state v for the visible layer based on $p(v \,|\, h)$ [10] called "encoding".

In addition, Ref. [18] presented that stacking multiple hidden layers in a GBRBM structure (i.e. Deep Boltzmann Machines, DBMs) can obtain a tighter variational lower bound on the log-probability of the test data and have the potential of learning internal representations that become increasingly complex. This however, would be computationally very expensive, since we adopt only one hidden layer in this paper.

4.3 RNN-GBRBM

As mentioned before, we will rarely be interested in decoding an individual extracted feature vector by a GBRBM.

Instead, consider extracting information of temporal dependencies from a series of extracted features and decoding these vectors into a fixed length decoded feature vector by joining multiple GBRBMs with a RNN structure, i.e. RNN-GBRBM.

Intuitively, we use RNN to encode the decoded features outputted from GBRBM into a more compact and informative form. Consider the output of RNN-GBRBM as an ultimate decoded feature $\vec{s}$. $\vec{s}$ is defined as Eq. (17). Where x_t is the t-th network data in a network data sequence, $d_{decoded}$ is the dimensionality of the decoded feature vector $\vec{s}(x_t, \Delta t)$, $s_k(x_t)$ is a binary value which indicates the k-th code of decoded features, and Δt is the number of the hidden units in RNN which indicates $\vec{s}$ is encoded with Δt decoded features $\vec{h}(x_t), \vec{h}(x_{t+1}), \ldots, \vec{h}(x_{t+\Delta t-1})$.

$$\vec{s}(x_t, \Delta t) = [s_1(x_t), s_2(x_t), \ldots, s_{d_{decoded}}(x_t)] \tag{17}$$

The training algorithm of RNN-GBRBM need estimate following parameters: $u^{(0)}$, b_v, b_h, W_{uh}, W_{uv}, W_{uu}, W_{vu}. An iteration of training is based on the following general scheme [16]:

- Propagate the current values of the hidden units $u^{(t)}$ in the RNN portion of the graph using Eq. (10).
- Calculate the RBM parameters that depend on the $u^{(t)}$ (Eqs. (8) and (9)) and generate the negative particles $v^{(t)*}$ using k-step block Gibbs sampling.
- Use CD (contrastive divergence) to estimate the log-likelihood gradient (Eq. (6)) with respect to W, $b_v^{(t)}$, $b_h^{(t)}$.
- Propagate the estimated gradient with respect to $b_v^{(t)}$, $b_h^{(t)}$ backward through time (BPTT) [23] to obtain the estimated gradient with respect to the RNN parameters.

5. Anomaly Classifier

Given a set of normal industrial control network data $X_{train} = \{x^{(1)}, x^{(2)}, \ldots, x^{(T)}\}$ and a set of unseen network data $X_{unknown} = \{x^{(t_u)}, x^{(t_{u+1})}, \ldots\}$. We use X_{train} as a training dataset to train the decoder μ. With this well-trained decoder, we decode the data in X_{train} and $X_{unknown}$ into decoded features S_{train} and $S_{unknown}$, defined as Eqs. (18) and (19).

$$S_{train} = \{\vec{s}(x_t, \Delta t)\}, \quad t = 0, 1, 2, \ldots, T - \Delta t + 1 \tag{18}$$

$$S_{unknown} = \{\vec{s}(x_t, \Delta t)\}, \quad t = t_u, t_{u+1}, \ldots \tag{19}$$

In this section, we propose the model of classifier ν and a semi-supervised incremental updating algorithm. The network data x_{test} will be a normal data if $\nu(x_{test}) \leq \tau$, $x_{test} \in X_{unknown}$. Otherwise x_{test} will be an outlier or anomaly.

5.1 Oversampling Principal Component Analysis

In the past, many outlier detection methods have been proposed. Typically, these existing approaches can be divided into three categories: distribution (statistical), distance and density-based methods [24]. It is worth noting that the above methods are typically implemented in batch mode, and thus they cannot be easily extended to anomaly detection problems with streaming data or online settings. While some online or incremental-based anomaly detection methods have been recently proposed, it turns out that their computational cost or memory requirements might not always satisfy online detection scenarios [25], [26].

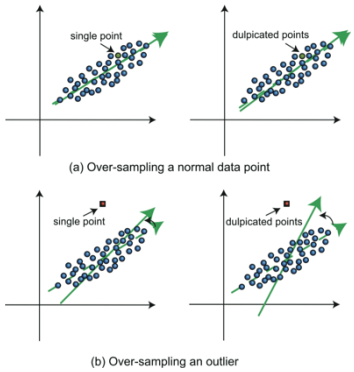


Fig. 4 The effects of adding/removing an outlier or a normal data instance on the principal directions.

Reference [24] proposed an online anomaly detection technique, named oversampling Principal Component Analysis (osPCA) in order to solve the above problems. Most importantly, it is a good algorithm to process binary vectors like decoded feature vectors since there is no other efficient algorithm for classifying or clustering bit arrays data.

Principal Component Analysis (PCA) is a well-known unsupervised dimension reduction method, which determines the principal directions of the data distribution. To obtain these principal directions, one needs to construct the data covariance matrix and calculate its dominant eigenvectors. These eigenvectors will be the most informative among the vectors in the original data space, and are thus considered as the principal directions. The osPCA consider the size of the dataset is typically large for practical anomaly detection problems, and thus it might not be easy to observe the variation of principal directions caused by the presence of a single outlier. As shown in Fig. 4, the proposed osPCA scheme will duplicate the target instance multiple times, and the idea is to amplify the effect of outlier rather than that of normal data so as to detect the outlier [24].

In this paper, we use osPCA as our classification algorithm. Firstly, sticking with previous definitions at start of this section, calculate the dominant eigenvectors of a specific dataset S_{train} mentioned before, and consider it as the principal direction u_t of S_{train}:

$$\Sigma_{S_{\text{unknown}}} u_t = \lambda u_t \tag{20}$$

Then, duplicate the unseen network data s_{test}, $s_{test} \in S_{unknown}$ r times, make them and S_{train} merge into a new set $S_{deviated}$, defined as:

$$S_{\text{deviated}} = S_{\text{train}} \cup \{s_u, \ldots, s_u\} \tag{21}$$

Likewise, calculate the principal direction u_d of $S_{deviated}$:

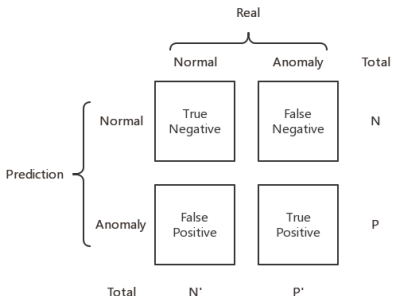


Fig. 5 The confusion matrix

$$\Sigma_{S_{\text{deviated}}} u_d = \lambda u_d \tag{22}$$

Once these eigenvectors are obtained, we use the absolute value of cosine similarity to measure the variation of the principal directions, i.e.

$$\theta = 1 - \left| \frac{\langle u_t, u_d \rangle}{||u_t|| \, ||u_d||} \right| \tag{23}$$

The θ can be considered as a "score of outlierness", which indicates the anomaly of target network data. We note that θ can be also viewed as the influence of the target network data in the resulting principal direction, and a higher θ score (closer to 1) means that the target network data is more likely to be an outlier. For a target network data x_{test}, if its θ is above the threshold τ, we then identify x_{test} as an outlier.

In this paper, we define the entities that are identified as anomaly as "positive", and the entities that are identified as normal as "negative". Additionally, we also define the correct prediction as "true", and wrong prediction as "false". To determine the parameter τ, we stick with reducing the false positives (maximizing the recall of anomaly detection) on the condition of minimum the false negatives (optimal precision). Mainly because 1. Every single false negative (shown in Fig. 5) is fatal to an industrial system, we cannot allow any underlying intrusion to penetrate into the system. 2. The normal data in the industrial control network is much more than the anomaly, we can sort the normal data out of the false positives (shown in Fig. 5) by human analyst at a very low cost, although, it is nearly impossible to sort the anomaly out of the false negatives.

We consider the parameter τ as the tolerance of anomaly. The larger τ is, the more tolerance the model has, and the less anomaly the model can find out. If the amount of false positives is much more than false negatives, then τ is underestimated, otherwise τ is overestimated. Therefore, in order to avoid unnecessary losses as much as possible, we initialize τ to 0 at first, and keep τ increasing until the condition that a minimum amount of false positives with 0 false negative occurs.

5.2 Semi-Supervised Incremental Updating Algorithm

We would also like to point out that the proposed techniques mentioned before might not enough in practical anomaly detection scenarios due to the problem of cool start and the continuous updates of intrusions. When a large amount of brand new network data appear in the network, we need update the decoder of NIDS, training it with the new types of normal data but discarding the outliers.

Table 1 The general framework for the semi-supervised incremental updating algorithm.

Algorithm 1 Semi-supervised Incremental Updating

$i \leftarrow 0$
$\tau_0 \leftarrow 0$
unknown datasets is $\{X^{(k)}_{unknown}, k = 0, 1, 2, ...\}$
initial training dataset is $X^{(0)}_{train}$
while $i < \infty$ **do**
  $\mu_i \leftarrow$ *train decoder with* $X^{(i)}_{train}$
  $S^{(i)}_{train} \leftarrow \mu_i(X^{(i)}_{train})$
  $\nu i \leftarrow$ *train osPCA with* $S^{(i)}_{train}$
  $\tau_i \leftarrow$ *update* τ_i *with* $S^{(i)}_{train}$
  $S^{(i)}_{unknown} \leftarrow \mu_i(X^{(i)}_{unknown})$
  while $s_{test} \in S^{(i)}_{unknown}$ **do**
    if $\nu_i(s_{test}) \leq \tau_i$ **then**
      $X^{(i)}_{train} \leftarrow X^{(i)}_{train} \bigcup s_{test}$
    else
      if *Determine* s_{test} *is anomaly manually* **then**
        $X^{(i)}_{train} \leftarrow X^{(i)}_{train} \bigcup s_{test}$
      else
        Discard s_{test}
      end if
    end if
  end while
  $X^{(i+1)}_{train} \leftarrow X^{(i)}_{train}$
  $i \leftarrow i + 1$
end while

The proposed semi-supervised incremental updating algorithm determines whether incoming data is anomaly and maintains the training dataset and the parameter τ incrementally. The process of algorithm can be divided into two phases:

- *Unsupervised training phase*: In the training phase, the algorithm performs two steps: (a) training the decoder μ with current training dataset $X^{(i)}_{train}$ and unseen dataset $X_{unknown}$,; (b) updating the classifier ν and the parameter τ with decoded features outputted by the well-trained decoder μ.
- *Supervised sorting phase*: In the sorting phase, the algorithm determines whether the unseen network data in $X_{unknown}$ joins the next training dataset $X^{(i+1)}_{train}$ or is discarded.

Details of the general framework for the semi-supervised incremental updating algorithm are shown in Table 1.

6. Experiment and Performance Evaluation

6.1 Description of Dataset

A set of experiments was performed on the Modbus PLC Simulator (MOD_RSSIM) for evaluating performance of the algorithm in Modbus environments. We generated 250000

Table 2 Features used in experiments.

Feature Name	Description	Value
Transaction Identifier	Identification of a MODBUS Request/Response transaction	0 - 65535
Protocol Identifier	0 = MODBUS protocol	0 - 65535
Length	Number of following bytes	0 - 65535
Unit Identifier	Identification of a remote slave connected on a serial line or on other buses.	0 - 255
function code	indicate to the server which kind of action to perform	0 - 255
data	depended on Length field	real number

packets, based on a simulated Modbus server and a simulated Modbus client on MOD_RSSIM, which consist of 1495 normal instances (shown in yellow dots in Fig. 7) and 5 anomalous instances (shown in red dots in Fig. 7). In order to construct the needed feature space, we have to determine a basic set of features describing the behavior pattern in a specific time interval (the observation epoch). The extracted features, derived from 12 key Modbus protocol fields, are listed in Table 2. Most features take a value in a numerical range, and are indicated as "continuous". Other features are nominal, i.e., they assume one value from a discrete set of possible values. These features are tagged as "discrete". Note that some features are derived, i.e., calculated starting from the values of other features.

Some of features in normal instances, for example, "byte count", are generated from a random non-linear combining function (shown in Eq. (24), (25)) that consists of a Poisson factor (sampled from a Poisson distribution), a number of Sine factors and a noise factor (sampled from a normal distribution), which ensure not only the time continuity of data but also the complexity to approximate to actual network data. And the same features in anomalous instances are randomly sampled only from the range [95, 105]. Obviously, it causes a covert deviated segment in a consecutive network data records, although the mean value of anomalous instances is still in the normal range. To simplify the process of data preprocessing, we organized these generated raw features into a fixed size vector directly, and forwardly arranged all of the vectors in chronological order into a matrix (or sequence) form that our system can deal with. By testing these counterfeit data in the forementioned mature NIDS, which is NGIDES, it turns out that there are only 3 anomalous instances can be detected. It has proved that some of the hidden anomaly features (like anomaly distribution) in this dataset can not be easily found out by conventional methods. The Fig. 6 shows an example of a counterfeit Modbus packet.

$$v_k(x_t) = aX + \sum_i^b c_i \sin(d_i * (t + e_i)) + fY \tag{24}$$

$$a + \sum_i^b c_i = 100, \quad f = 1 \tag{25}$$

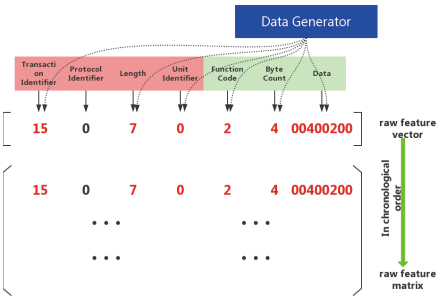


Fig. 6 An example of a counterfeit Modbus packet

$$X \sim \text{Poisson(a)}, \quad Y \sim N(100, 1), \quad e_i \sim U(-\pi, \pi), \quad d_i \sim U(0, 50) \tag{26}$$

Moreover, over 13000 real Modbus packets with 9 anomalous instances have also been used in our experiments. However, these anomalous instances only includes two types of common anomalous behaviors since the hidden anomaly cases are not able to be collected.

6.2 Evaluation Criteria

If the task of the algorithm is to classify cases into one of the several categories, examining a confusion matrix can be highly informative (shown in Fig. 5). A confusion matrix contains as many rows and columns as there are classes. For every case, its class is chosen to be that corresponding to the output neuron that has the maximum activation. The content of the entry at the i-th row and the j-th column of the confusion matrix is the number of test cases that truly belong to class j but which were classified into class i. Ideally, one would obtain a strictly diagonal matrix. Quantities in off-diagonal positions represent misclassifications. The principal strength of the confusion matrix is that it clearly identifies the nature of errors, as well as their quantity. The experimenter is then free to evaluate the performances of a NIDS in terms of relative severity of misclassifications. The evaluation parameters that have been selected are thus:

- false positive rate: as known as false alarm ratio, refers to the probability of falsely rejecting the null hypothesis for a particular test:

$$\textit{false positive rate} = \frac{FP}{N'}$$

- true negative rate: as known as specificity, measures the proportion of negatives that are correctly identified:

$$\textit{true positive rate} = \frac{TP}{P}$$

Ideally, it is the most worthwhile goals for the NIDS that the false positive rate is supposed to be zero and the true positive rate is supposed to be one.

6.3 Result Using Decoders

We first apply our GBRBM-decoder on the entire exper-

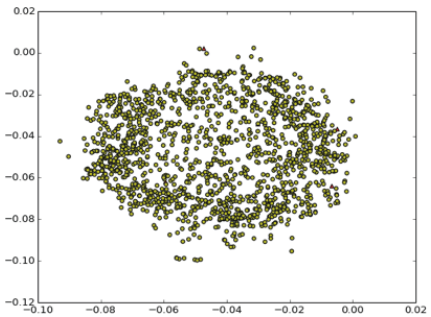
imental dataset. To see whether the learned GBRBM-decoded feature vectors preserve class information, we used t-distributed stochastic neighbor embedding (t-SNE) [27] to visualize the decoded and extracted feature vectors of all experimental data from both normal and anomalous classes. Figure 7 (a) shows that for the experimental dataset the 12-dimensional extracted features are visualized in a two-dimensional space.

Figure 7 (a) (b) show a little difference between the distributions of original extracted features vectors and our 200-bits decoded feature vectors produced by the GBRBM. The GBRBM seems to provide a more uniform points distribution and better dispersion due to the decoding process. But there is still no clear boundary between normal data and anomalous data in Fig. 7 (b). The red points (anomaly) are distributed around the edge of the cluster and mixed with some of the yellow points (normal). And as shown in Fig. 7 (c), the normal data are highly concentrated in the upper right corner of the 2-D feature space. Rather, four fifth of outlier are placed at the bottom left of the space dispersedly. It indicates that our decoded feature vectors are far better at preserving the information of original data than the extracted feature vectors when the decoded feature vectors ascended up the dimension from 200 to 2000. Higher dimensionality means more bits of information or more "experts" (each expert model can constrain a different subset of the dimensions in a high-dimensional space, and their product will then constrain all of the dimensions [13]) on data. The results of Fig. 7 (a) (b) (c) reveal that the GBRBM significantly magnifies the fine distinction between normal data and anomalous data and optimizes the representations of network data, which are better for classification.

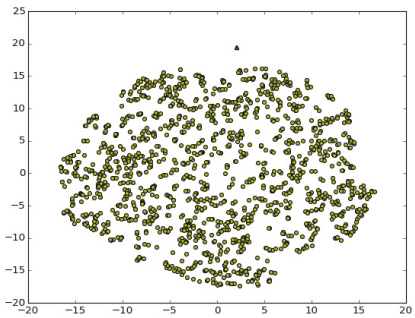
Moreover, using RNN-GBRBM decoder to process the same dataset, as shown in Fig. 7 (d), got a really astonishing result that all of outliers have been separated from the normal data points since RNN structure takes the context of network data into account.

6.4 Performance Evaluation

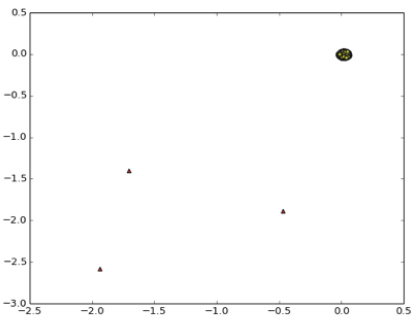
For the purpose of comparison, the results of the proposed hybrid method are compared with that of a single osPCA and an one class support vector machine (OCSVM) based algorithm [28]. Note that The OCSVM algorithm maps input data into a high dimensional feature space (via a kernel) and iteratively finds the maximal margin hyper plane which best separates the training data from the origin. The OCSVM may be viewed as a regular two-class SVM where all the training data lies in the first class, and the origin is taken as the only member of the second class. The outcome of the experiments is summarized in Table 3. For each experiment, the same dataset has been tested, including 250000 conterfeit network packets (with 1495 normal instances and 5 anomalous instances) and 13000 real Modbus packets (with 874 normal instances and 9 anomalous instances). Also the performance of each method is indicated for counterfeit data and real data respectively in



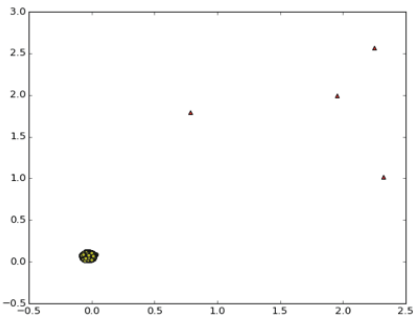
(a)



(b)



(c)



(d)

Fig. 7 A two-dimensional embedding based on t-SNE. (a) 12-dimensional extracted feature vectors. (b) 200-bit GBRBM-decoded feature vectors. (c) 2000-bit GBRBM-decoded feature vectors. (d) 2000-bit RNN-GBRBM-decoded feature vectors

Table 3 Experimental results

Model	Precision	Recall
RNN-GBRBM & osPCA	0.9553, 0.9231	0.9985, 0.7915
osPCA	0.9252, 0.9252	0.9525, 0.7232
OCSVM	0.9739, 0.9210	0.7552, 0.5341
NGIDES	0.9987,0.9413	0.4550, 0.3142

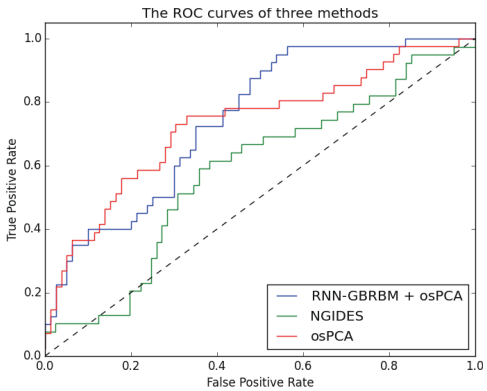


Fig. 8 The ROC curves of three methods

Table 3. On the first row of the table, the precision (1 − false negative rate) and recall (1 − false positive rate) for RNN-GBRBM decoder & osPCA are listed. And on the following two rows of the table, the results using single osPCA and OCSVM with radial basis function (RBF) kernel are shown analogously. The left number shows the result for counterfeit data and right for real data.

In order to show the effectiveness of the decode more directly, in Fig. 8, we show the receiver operating characteristic (ROC) curve of each method, which is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The x-axis indicate the false positive rate from 0 to 1, and the y-axis indicate the true positive rate from 0 to 1. The curve in Fig. 8 is created by plotting the true positive rate against the false positive rate at various threshold settings. It can be seen that there is a substantial improvement going from the proposed method to the other two methods since the area under the curve (AUC) of the proposed method (RNN-GBRBM decoder & osPCA) is much larger than that of other methods.

7. Conclusions

In this paper, we proposed a novel RNN-GBRBM based feature decoder, which aimed to make extracted features a better representation for classification. We have shown the performances of the proposed system from two aspects. On one hand, we show the effectiveness of the RNN-GBRBM decoder, on the other hand we illustrate the sustainability of this architecture, which means it has a better performance as time goes on and environment varies, since the combination of osPCA and semi-supervised incremental updating algorithm. The method delivers a substantial recall increases while maintaining a general precision.

Since the complexity of deep architecture is relatively high, the GPU based computation device is needed to accelerate the parallel computation speed on the increase of network traffic or model complexity. The simulation model in this paper only applies the relatively simple architecture for quick verification of the algorithm feasibility. And the proposed architecture is merely an attempt that applies the deep learning model on the anomaly detection in industrial control network. However, in real network environment, the actual scalability cost or other factors will be considered in the future.

References

[1] B. Galloway and G.P. Hancke, "Introduction to Industrial Control Networks," IEEE Commun. Surveys Tuts., vol.15, no.2, pp.860–880, 2013.
[2] C. Tsang and S. Kwong, "Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction," Ind. Technol. 2005. ICIT 2005. . . . , pp.51–56, 2005.
[3] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," Comput. Networks, vol.51, no.12, pp.3448–3470, 2007.
[4] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," Comput. Secur., vol.28, no.1–2, pp.18–28, 2009.
[5] D. Anderson, T. Frivold, and A. Valdes, "Next-generation Intrusion Detection Expert System (NIDES): A summary," SRI Int., p.47, May 1995.
[6] P. a Porras and P.G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," Proc. 20th NIST-{NCSC} Natl. Inf. Syst. Secur. Conf., pp.353–365, 1997.
[7] V.M. Igure, S.A. Laughter, and R.D. Williams, "Security issues in SCADA networks," Comput. Secur., vol.25, no.7, pp.498–506, 2006.
[8] M.B. Line, A. Zand, G. Stringhini, and R. Kemmerer, "Targeted Attacks against Industrial Control Systems: Is the Power Industry Prepared?," Proc. ACM Work. Smart Energy Grid Secur., pp.13–22, 2014.
[9] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on SCADA systems," Proc. - 2011 IEEE Int. Conf. Internet Things Cyber, Phys. Soc. Comput. iThings/CPSCom 2011, pp.380–388, 2011.
[10] A. Fischer and C. Igel, "An Introduction to Restricted Boltzmann Machines," Lect. Notes Comput. Sci. Prog. Pattern Recognition, Image Anal. Comput. Vision, Appl., vol.7441, pp.14–36, 2012.
[11] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," Neurocomputing, vol.122, pp.13–23, 2013.
[12] G. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines A Practical Guide to Training Restricted Boltzmann Machines," Computer (Long. Beach. Calif)., vol.9, no.3, p.1, 2010.
[13] G.E. Hinton, "Training products of experts by minimizing contrastive divergence.," Neural Comput., vol.14, no.8, pp.1771–1800, 2002.
[14] T. Yamashita, M. Tanaka, E. Yoshida, Y. Yamauchi, and H. Fujiyoshii, "To be bernoulli or to be gaussian, for a restricted boltzmann machine," Proc. - Int. Conf. Pattern Recognit., pp.1520–1525, 2014.
[15] I. Sutskever, G. Hinton, and G. Taylor, "The Recurrent Temporal Restricted Boltzmann Machine," Neural Inf. Process. Syst., vol.21, no.1, pp.1601–1608, 2008.
[16] N. Boulanger-Lewandowski, P. Vincent, and Y. Bengio, "Modeling

Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription," Proc. 29th Int. Conf. Mach. Learn., pp.1159–1166, 2012.
[17] J. Martens, "Learning Recurrent Neural Networks with Hessian-Free Optimization," Proc. 28th Int. Conf. Mach. Learn., pp.1033–104, 2011.
[18] R. Salakhutdinov and G.E. Hinton, "Deep Boltzmann Machines," Proc. 12th Int. Conf. Artif. Intell. Statics, no.3, pp.448–455, 2009.
[19] M. Mantere, M. Sailio, and S. Noponen, "Network Traffic Features for Anomaly Detection in Specific Industrial Control System Network," Futur. Internet, vol.5, no.4, pp.460–473, 2013.
[20] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," 2010 IEEE Symp. Secur. Priv., vol.0, no.May, pp.305–316, 2010.
[21] M. Mahoney and P.K. Chan, "PHAD: Packet header anomaly detection for identifying hostile network traffic," Florida Inst. Technol. Tech. Rep. CS-2001-04, no.1998, 2001.
[22] M. Welling, M. Rosen-Zvi, and G. Hinton, "Exponential family harmoniums with an application to information retrieval," Adv. Neural Inf. Process. Syst., vol.17, pp.1481–1488, 2005.
[23] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence, pp.399–421, MIT Press, 2013.
[24] Y.-J. Lee, Y.-R. Yeh, and Y.-C.F. Wang, "Anomaly detection via online oversampling principal component analysis," IEEE Trans. Knowl. Data Eng., vol.25, no.7, pp.1460–1470, 2013.
[25] T. Ahmed, B. Oreshkin, and M. Coates, "Machine learning approaches to network anomaly detection," Proc. 2nd USENIX Work. Tackling Comput. Syst. Probl. with Mach. Learn. Tech., pp.7:1–7:6, 2007.
[26] D. Pokrajac, A. Lazarevic, and L.J. Latecki, "Incremental Local Outlier Detection for Data Streams," 2007 IEEE Symp. Comput. Intell. Data Min., no.Cidm, pp.504–515, 2007.
[27] L. Van Der Maaten and G. Hinton, "Visualizing Data using t-SNE," J. Mach. Learn. Res., vol.9, pp.2579–2605, 2008.
[28] F. Schuster and A. Paul, "Potentials of Using One-class SVM for Detecting Protocol-specific Anomalies in Industrial Networks," 2015 IEEE Symp. Series on Comput. Intell., pp.83–90, 2015.

Hua Zhang received the B.S. degree in telecommunications engineering from the Xidian University in 1998, the M.S. degree in cryptology from Xidian University, Xi'an, China in 2005, and the Ph.D degree in cryptology from Beijing University of Posts and Telecommunications in 2008. Now she is a lecturer of Beijing University of Posts and Telecommunications. Her research interests include cryptography, information security and network security.

Shixiang Zhu received the B.S. and M.S. degrees in Computer Science from Beijing University of Posts and Telecommunications in 2014 and 2017, respectively. During 2014 to 2017, he stayed in State Key Lab of Networking and Switching, Institute of Network Technology, Beijing University of Posts and Telecommunications. His research interests include deep learning, machine learning and network security. E-mail: meowoodie@outlook.com

Xiao Ma received the B.S. degree in industrial automation from the Northeastern University in 1999. Now he is Assistant Chief Engineer of Beijing Kedong electric power control system Co. Ltd. His research interests include dispatching automation, information security and network security. E-mail: maxiao3@sgepri.sgcc.com.cn

Jun Zhao received the B.E. degree in Electric Power System and Automation from Wuhan Hydraulic and Electrical Engineering University in 1997. Now he is Senior Engineer of StateGrid Liao Ning Electric Power Supply Co. Ltd Dispatching Automation Department. His research interests include dispatching automation, information security and network security. E-mail: zhaojun@ln.sgcc.com.cn

Zeng Shou received the B.E. degree in Electric Power System and Automation from Science Technology Beijing University in 1998. Now he is Senior Engineer of StateGrid Liao Ning Electric Power Supply Co. Ltd Dispatching Automation Department. His research interests include dispatching automation, information security and network security. E-mail: gaoxiaosoft@163.com