Feature Shift Detection: Localizing Which Features Have Shifted via Conditional Distribution Tests

Sean M. Kulinski **Saurabh Bagchi** **David I. Inouye**

School of Electrical and Computer Engineering
Purdue University
{skulinsk,sbagchi,dinouye}@purdue.edu

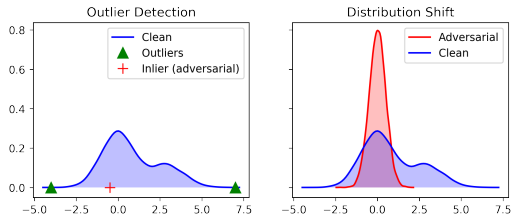
Abstract

While previous distribution shift detection approaches can identify if a shift has occurred, these approaches cannot localize which specific features have caused a distribution shift—a critical step in diagnosing or fixing any underlying issue. For example, in military sensor networks, users will want to detect when one or more of the sensors has been compromised, and critically, they will want to know which specific sensors might be compromised. Thus, we first define a formalization of this problem as multiple conditional distribution hypothesis tests and propose both non-parametric and parametric statistical tests. For both efficiency and flexibility, we then propose to use a test statistic based on the density model score function (*i.e.*, gradient with respect to the input)—which can easily compute test statistics for all dimensions in a single forward and backward pass. Any density model could be used for computing the necessary statistics including deep density models such as normalizing flows or autoregressive models. We additionally develop methods for identifying when and where a shift occurs in multivariate time-series data and show results for multiple scenarios using realistic attack models on both simulated and real world data.

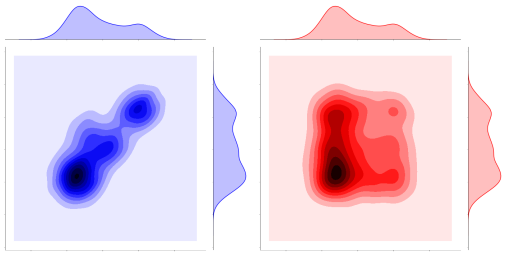
1 Introduction and Motivation

Adversarial attacks on sensor streams that feed into complex machine learning systems can create serious vulnerabilities. To support decision making in an adversarial setting, it is critical to identify *when* an adversarial attack has started and in a multi-sensor environment, *which* sensors have been compromised. The identification of which particular sensors have been compromised, a process referred to as *"localization"*, is critically important for correction either by physical intervention or hardware reset. While there has been much work on anomaly detection and some on distribution shift [2, 19, 27], the task of identifying or explaining the anomalous behavior or distribution shift remains a relatively open problem. We consider primarily a sensor network attack, where the adversary is able to observe and to control one or more nodes, which are then denoted as "compromised (or attacked) sensors". While most previous adversarial or anomaly detection methods focus on identifying an attack from a single sample [9, 22, 31], we focus on detecting sequences of samples that show anomalous behavior—which we argue is more natural for sensor streams and enables detection of powerful adversaries. For example, consider Figure 1a, which shows that taken in isolation a single sensor reading may not appear as anomalous, but a sequence of readings may appear to deviate from a notional correct distribution, thus identifying the attack. ***Thus, we frame our core problem as distribution shift detection via hypothesis testing instead of standard anomaly detection.*** We make the significant observation that dependencies between samples within a sensor (*i.e.*, captured by their marginal distribution) and dependencies across sensors (*i.e.*, captured by conditional distributions of one sensor, conditioned on other sensor streams) enable the detection of powerful adversarial attacks

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.



(a) In isolation, adversarial examples may appear as inliers rather than outliers (left), but when when modeling multiple adversarial examples, a shift from the original distribution can be detected because of the relationships between adversarial examples (right)—*i.e.*, adversarial examples are coming from a different distribution. While this illustration is only 1D, the same intuition can be applied to high-dimensional data.



(b) If an adversary compromises one sensor, they may try to match the marginal distribution of that sensor (*e.g.*, by looping the sensor data from a previous day). However, a shift in distribution can be detected by joint analysis across sensors because the adversary does not know the dependencies between all sensors.

Figure 1: (Left subfigure) Rationale for detecting a single compromised sensor using distribution shift detection rather than individual sensor readings. (Right subfigure) Rationale for using conditional distributions of sensor values for correlated sensors, rather than only marginal distributions, to detect attacks against single or multiple sensors.

that try to avoid detection. For example, consider Figure 1b, where the adversary can match the *marginal distribution* of sensor values for a single compromised sensor, but she may not be able to match the *conditional distribution* of the sensor values, conditioned on that of other sensors that have *not* been compromised. This raises the bar on the attack in that *all* sensors which are conditioned on one another will have to be compromised to defeat the detection.

One of the key challenges of our approach is computational efficiency (especially in an online learning setting) because we have to run a statistical test at every time point and for every feature in the multi-sensor setting (each sensor contributes one feature). To address this challenge, we develop a novel test statistic based on Fisher divergence that could be computed in an online manner for all features *simultaneously*. This test statistic is easy to calculate even for complex deep neural density models. Further, it has the desirable properties that it allows us to consider all features simultaneously and thus scales well with the number of sensors—roughly reducing the amount of computation by $O(d)$ where d is the number of sensors since we can compute the needed test statistic for all features simultaneously in one forward and backward pass of a density model. We extend our approach to handle time-series data where a straightforward application of our earlier approach enables us to identify when a sensor has been compromised and in a sensor network attack case, which set of sensors is compromised. We summarize our contributions as follows:

1. We motivate and define the problem of feature shift detection for localizing which specific sensor values (or more broadly which features) have been manipulated.
2. We define conditional distribution hypothesis tests and use this formalization as the key theoretical tool to approach this problem. We propose both model-free and model-based approaches for this conditional test and develop practical algorithms.
3. In particular, we propose a score-based test statistic (*i.e.*, gradient with respect to input) inspired by Fisher divergence but adapted for a novel context as a distribution divergence measure. This test statistic scales to higher dimensions and can be applied to modern deep density models such as normalizing flows and autoregressive models.
4. We propose simple extensions to time series data so that we can identify both when and where a security attack has occurred.

Notation. Let p denote a reference distribution and q be a query or test distribution, which may be the same or different than p. Let X denote samples from p and Y denote samples from q similarly. For model-based methods, we will denote the models as $\hat{p}$ and $\hat{q}$ for distributions p and q respectively. We will denote vectors by boldcase letters (e.g. $\boldsymbol{x}$) and single elements by non-bold face letters (*e.g.*, x_1, x_3, x_j). We will denote $\boldsymbol{x}_{-j}$ as the feature vector that includes all features except for the j-th feature (*e.g.*, $\boldsymbol{x}_{-2} = [x_1, x_3, x_4, \ldots, x_d]$, where d is the number of features. The set of all possible $\boldsymbol{x}$, x_j, and $\boldsymbol{x}_{-j}$ will be denoted by $\mathcal{X}$, $\mathcal{X}_j$ and $\mathcal{X}_{-j}$ respectively. We will denote a set of samples

(formed as a matrix) as a capital letter, *i.e.*, $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m]^T \in \mathbb{R}^{n \times d}$, where n is the number of samples and d is the number of dimensions (which is equal to the number of sensors in our setup). The set of compromised sensors will be denoted by $\mathcal{A} \subset \{1, 2, \cdots, d\}$, and its complement will be denoted by $\overline{\mathcal{A}}$.

1.1 Related Works

Distribution shift have been considered in many contexts (see [6, 19] for survey articles). Given a reference distribution, p, and a query distribution, q, distribution shift is the problem of detecting when q is different from p. While there exist standard methods for univariate detection, the multivariate case is still maturing [25]. Generally, there are different types of distribution shift including (1) covariate shift in which $q(\boldsymbol{x}) \neq p(\boldsymbol{x})$ but the conditional of the target given the covariates is the same, *i.e.*, $q(y|\boldsymbol{x}) = p(y|\boldsymbol{x})$, (2) label shift in which $q(y) \neq p(y)$ but $q(\boldsymbol{x}|y) = p(\boldsymbol{x}|y)$, and (3) concept drift in which either (a) $q(\boldsymbol{x}) = p(\boldsymbol{x})$ but $q(y|\boldsymbol{x}) \neq p(y|\boldsymbol{x})$ or (b) $q(y) = p(y)$ but $q(\boldsymbol{x}|y) \neq p(\boldsymbol{x}|y)$ [16, 18, 20]. For classification settings, a hierarchical hypothesis testing framework was developed for detecting concept drifts [29]. More recently, joint distribution shift detection via two-sample hypothesis testing for high-dimensional data has been explored in [25] motivated by detecting shifts for ML pipelines. It empirically compares many different methods including multivariate hypothesis tests (via Maximum-Mean Discrepancy [8]), a combination of multiple univariate hypothesis tests (via multiple marginal KS statistic tests combined via the Bonferonni correction [3]), and possible dimensionality reduction techniques prior to hypothesis testing. The key difference between our work and previous work (including [25]) is that we are concerned with *localizing* which feature or sensor (or set of sensors) is responsible for the distribution shift rather than merely determining if there is a shift. Thus, most previous methods are inapplicable to our problem setting because they either reduce the dimensionality (thereby removing information needed for localization) or perform a joint distribution test (via MMD or similar), which cannot localize which features are compromised. Additionally, most prior work considers the supervised setting whereas we are more interested in the unsupervised setting for detecting compromised sensors.

We will leverage insights from score-based methods (*i.e.*, the gradient of a density with respect to the input) [11] in our proposed test statistic for scalability. Score matching was originally proposed as a novel way to estimate non-normalized statistical models because the score is independent of the normalizing constant [11, 30]. This method of estimation can be seen as minimizing the Fisher Divergence [26], which is a divergence metric based on the score function [17]. Score matching has also been generalized to the discrete case [12, 21].

2 Feature Shift Detection

In our context, we assume that various sensors are *controllable* (*i.e.*, the values can be manipulated) by an adversary. Our key assumption is that an adversary can only control a subset of sensors in the network—otherwise, an adversary could manipulate the entire system and there would be no hope of detecting anything from sensor data alone. We also assume that there is no causal relationship between sensor values. Our goal is to identify which subset of sensors have been compromised or attacked. Formally, our problem statement can be defined as:

Definition 1. [Feature Shift Detection Problem Formulation] *We are given two sets of samples from p and q, denoted X and Y respectively. Now, identify which subset of features (if any) have been attacked, denoted by $\mathcal{A} \subset \{1, 2, \cdots, d\}$.*

The key tool we will use for this problem is two-sample hypothesis statistical tests and, in particular, a variant we define called the *conditional distribution hypothesis test*. We propose both model-free (*i.e.*, non-parametric) approach and model-based approaches to create this test. Importantly, we propose the novel use of a score-based test statistic for enabling more scalable hypothesis testing.

Unlike joint distribution shift detection, which cannot localize which features caused the shift, we define a new hypothesis test for each feature individually. Naïvely, the simplest test would be to check if the marginal distributions have changed for each feature (as explored by [25]); however, the marginal distribution would be easy for an adversary to simulate (*e.g.*, by looping the sensor values from a previous day). Thus, marginal tests are not sufficient for our purpose. Therefore, we propose to use conditional distribution tests. More formally, our null and alternative hypothesis for the j-th

feature is that its full conditional distribution (*i.e.*, its distribution given all other features) has not shifted for all values of the other features.

Definition 2. [Conditional Distribution Hypothesis Test] *The feature shift hypothesis test is defined by the following null and alternative hypotheses:*

$$H_0 : \forall \boldsymbol{x}_{-j} \in \mathcal{X}_{-j}, \quad q(x_j|\boldsymbol{x}_{-j}) = p(x_j|\boldsymbol{x}_{-j}); \quad H_A : \exists \boldsymbol{x}_{-j} \in \mathcal{X}_{-j}, \quad q(x_j|\boldsymbol{x}_{-j}) \neq p(x_j|\boldsymbol{x}_{-j}).$$

Essentially, this test checks for any discrepancy in prediction of one feature given all the other features. Thus, if there are probabilistic dependencies between sensors (*e.g.*, two vibration sensors near each other), then any independent manipulation of a subset of sensors will be noticeable from a joint distribution perspective. We note that a special case of this test is the marginal distribution test for each feature. We now define a simple class of statistics that could be used to perform these tests.

Definition 3. [Expected Conditional Distance] *The* expected conditional distance (ECD) *test statistic is defined as follows:* $\gamma = \mathbb{E}_{\boldsymbol{x}_{-j}}[\phi(p(x_j|\boldsymbol{x}_{-j}), q(x_j|\boldsymbol{x}_{-j}))]$ *where* ϕ *is a statistical divergence between two distributions (*e.g.*, KL divergence, KS statistic, Anderson-Darling statistic).*

To estimate ECD from samples, there are two design questions: (1) How should the conditional distributions $p(x_j|\boldsymbol{x}_{-j})$ and $q(x_j|\boldsymbol{x}_{-j})$ be estimated? and (2) Which statistical divergence ϕ should be used? First, for question (1), we develop *model-free* (via KNN) and *model-based* (via density model) approaches for estimating the conditional distributions. An illustration of the similarities and differences between model-free and model-based can be seen in Fig. 2. Second, for question (2), we compare the common non-parametric Kolmogorov-Smirnov (KS) divergence statistic with a Fisher divergence that only requires computation of the score function (*i.e.*, the gradient of the log density with respect to the input).

Model-free approach via k-nearest neighbors. To avoid any modeling assumptions, we could approximate the conditional distribution with samples. We will only consider the estimation of the distribution distance ϕ given an $\boldsymbol{x}_{-j}$: $\phi(p(x_j|\boldsymbol{x}_{-j}), q(x_j|\boldsymbol{x}_{-j})) \approx \phi(A_j(\boldsymbol{x}_{-j}; k), B_j(\boldsymbol{x}_{-j}; k))$ where $A_j(\boldsymbol{x}_{-j}; k) = \{z_j : \boldsymbol{z} \sim X, \boldsymbol{z}_{-j} \in \mathcal{N}_k(\boldsymbol{x}_{-j}; X)\}$, $B_j(\boldsymbol{x}_{-j}; k) = \{z_j : \boldsymbol{z} \sim Y, \boldsymbol{z}_{-j} \in \mathcal{N}_k(\boldsymbol{x}_{-j}; Y)\}$, X and Y denote the empirical distributions of p and q respectively, and $\mathcal{N}_k(\boldsymbol{x}_{-j}; X)$ denotes the set of k nearest samples in the dataset X when only considering $\boldsymbol{x}_{-j}$ (*i.e.*, ignoring x_j). If $\boldsymbol{x}$ is in a high-dimensional space, then k-nearest neighbors may be quite far from each other and thus may not approximate the conditional distribution well (see Fig. 2 for a 2D example). In our experiments, we found that the model-free approach to estimating conditional distributions is quite computationally expensive (approximately 4 times more expensive) and performs relatively poorly in comparison to the model-based approaches.

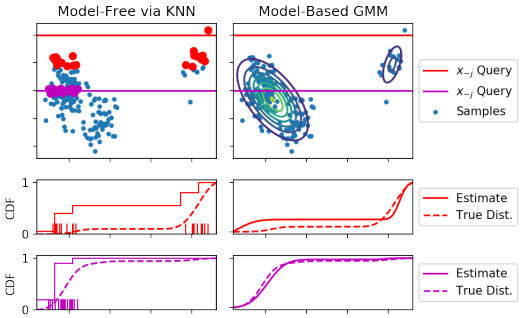


Figure 2: Conditional distributions (bottom) can be estimated using a model-free approach via KNN (left) or a model-based approach (right). Model-free could be better if true distribution is complex but may suffer from sparsity of data (especially in high dimensions) (notice red conditional distribution query point) whereas model-based will perform well if the data is sparse but may suffer if the modeling assumptions are violated.

Model-based approach via explicit density models. As an alternative to the computationally expensive model-free approach via KNN, we can consider a model-based approach that explicitly models the joint density function $\hat{p}(\boldsymbol{x})$ and computes the conditional distributions $\hat{p}(x_j \mid \boldsymbol{x}_{-j})$ based on the joint model. Ideally, we would be able compute the full conditionals, *i.e.*, $\hat{p}(x_j \mid \boldsymbol{x}_{-j})$ and be able to compute the ϕ efficiently (or even in closed-form). For Gaussian distributions, these conditional distributions can be computed in closed-form. For even more modeling power, we could estimate a normalizing flow such as MAF [23] or deep density destructors [13]. However, computing the conditional distributions $p(x_j|\boldsymbol{x}_{-j})$ of normalizing flows or recent deep density models is computationally challenging because, while they can explicitly compute the joint density function $p(\boldsymbol{x})$, they cannot provide us conditional densities easily. The simplest way to compute conditional densities from a deep density model is to compute a grid of joint density values along x_j while

keeping $\boldsymbol{x}_{-j}$ fixed and then renormalizing. To compute this for every dimension, this grid method would require $O(Md)$ evaluations of the deep density model where M is the number of grid points and d is the number of dimensions. Additionally, even in the case where the conditional distribution can be computed in closed-form (*e.g.*, multivariate Gaussian or Gaussian-copulas), we still have to compute the relevant test statistic for every feature—thus roughly scaling as $O(d)$ tests. In the next section, we show how to circumvent this key difficulty by leveraging a score-based test statistic.

Fisher divergence test statistic via score function. We now consider the second question about which statistical divergence to use given an estimate of the conditional distributions. For the model-free approach, we only have samples so we must use a model-free test statistic such as the KS or Anderson-Darling (AD) test statistic. For the model-based approach, we propose to use another divergence ϕ that only requires the score function, which is defined as the gradient of the log density [11]:[1] $\psi(\boldsymbol{x}; p) \triangleq \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$. Traditionally, this score function has been used primarily for estimating distributions with intractable normalization constants—a method called *score matching*—because the score function is independent of the normalization constant [11]. The Fisher divergence can now be defined as the expected difference of the score function [21, 26, 28]:

$$\phi_{\text{Fisher}}(p, q) \triangleq \mathbb{E}_{p(\boldsymbol{x})+q(\boldsymbol{x})}[\|\psi(\boldsymbol{x}; p) - \psi(\boldsymbol{x}; q)\|^2] = \mathbb{E}_{p(\boldsymbol{x})+q(\boldsymbol{x})}\left[\left\|\nabla_{\boldsymbol{x}} \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right\|^2\right] \tag{1}$$

This divergence gives a way to measure the dissimilarity of distributions similar to the KL divergence: $\text{KL}(p, q) = \mathbb{E}_{p(\boldsymbol{x})}[\log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}]$. If $\hat{p}$ and $\hat{q}$ are estimated models for p and q respectively, we can define the simple plug-in estimator: $\phi_{\text{Fisher}}(p, q) \approx \phi_{\text{Fisher}}(\hat{p}, \hat{q}) = \mathbb{E}_{\hat{p}(\boldsymbol{x})+\hat{q}(\boldsymbol{x})}[\|\psi(\boldsymbol{x}; \hat{p}) - \psi(\boldsymbol{x}; \hat{q})\|^2]$. We argue that this theoretically-motivated score-based statistic is desirable for the following reasons:

1. *Multiple feature test statistics simultaneously* - The joint score function can be used to trivially estimate the conditional score functions for all features at the same time because the conditional score function is merely the partial derivative (*i.e.*, a component of the gradient) of the log density function: $\psi(x_j; p_{x_j|\boldsymbol{x}_{-j}}) = \frac{\partial}{\partial x_j}[\log p(x_j|\boldsymbol{x}_{-j})] = \frac{\partial}{\partial x_j}\left[\log \frac{p(\boldsymbol{x})}{p(\boldsymbol{x}_{-j})}\right] = \frac{\partial}{\partial x_j} \log p(\boldsymbol{x}) = [\psi(\boldsymbol{x}; p)]_j$, where the term with $p(\boldsymbol{x}_{-j})$ is constant w.r.t. x_j and thus has a zero partial derivative—*i.e.*, the normalizing constant for conditional distributions can be ignored similar to the motivation for using scores for estimating unnormalized density models [11].
2. *Easy to compute for deep neural density and online learning models* - The score function is easy to compute for all features simultaneously using auto differentiation, even for complex deep density models (*e.g.*, [5, 13, 14, 24])—*i.e.*, only a single forward and backward pass is needed to compute all d conditional score functions corresponding to each feature. This makes this approach very attractive for online learning where the deep density can be updated and a test statistic can be computed at the same time.

Extension to time series. We can extend this method to online time-series data in several straightforward ways. We can merely perform these hypothesis tests at regular intervals to determine when an attack has occurred as we will initially explore in our experiments; this enables the identification of when an attack occurs. Again, this will require fast and efficient methods for computing test statistics in an efficient manner so we choose to use the score-based Fisher divergence. Second, in Section 3.2 we introduce a time-dependent variant of bootstrapping to find thresholds which account for natural shifts across time. Lastly, it is fairly straightforward to extend our framework to account for some dependencies across time by modeling the difference between consecutive time points, *i.e.*, model $\delta_{\boldsymbol{x}} = \boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t-1)}$, rather than the raw data points themselves. This could be further extended to second-order or higher-order differences and all these models could be cascaded for detecting shifts in the conditional distributions.

3 Experiments

Attack Model. Because we assume that the adversary can only observe and control a subset of sensors, an adversary cannot model or use the values of the other sensors to hide their attacks. Thus, if an attacker wants to avoid detection, the best they can do is to simulate the marginal

[1] More strictly, this is the traditional score function defined for Fisher information with respect to a hypothetical location parameter μ [11], *i.e.*, $\psi(\boldsymbol{x}; p) = \nabla_\mu p_\mu(\boldsymbol{x})|_{\mu=0} = \nabla_\mu p(\boldsymbol{x} + \mu)|_{\mu=0} = \nabla_{\boldsymbol{x}} p(\boldsymbol{x})$.

distribution of the subset of sensors they control independent of the other sensors. Thus, a strong adversarial attack distribution (to avoid detection) given our assumption is a marginal distribution attack defined as: MarginalAttack$(p, j) \triangleq p(\boldsymbol{x}_{-j})p(\boldsymbol{x}_j)$—notice that this forces $\boldsymbol{x}_j$ and $\boldsymbol{x}_{-j}$ to be independent under this attack distribution. Similarly, for a subset of sensors, we can define MarginalAttack$(p, \mathcal{A}) \triangleq p(\boldsymbol{x}_{\overline{\mathcal{A}}})p(\boldsymbol{x}_{\mathcal{A}})$. Practically, we implement this marginal attack by permuting the j-th column (or subset of columns) of Y (samples from the q distribution), which corresponds to sampling from the marginal distribution and breaks any dependencies between the other features. Also, we assume that once a sensor has been attacked, it will be attacked for all time points afterwards.

Attack Strength and Difficulty. We will define the *strength* of an attack in sensor networks as the number of observable and controllable sensors in the network. Compromising one sensor is the weakest attack model whereas compromising all sensors is the strongest attack model. Given our key observation that dependencies between features are critical for detecting sensor network attacks, we define the *difficulty* of the attack problem as the negative mutual information between compromised and uncompromised random variables of the true distribution. Formally, we define the attack strength and attack difficulty as: AttackStrength$(\mathcal{A}) \triangleq |\mathcal{A}|$, AttackDifficulty$(p_{\boldsymbol{x}}) \triangleq -\text{MI}(\boldsymbol{x}_{\mathcal{A}}, \boldsymbol{x}_{\overline{\mathcal{A}}})$, where $\mathcal{A} \subset \{1, 2, \cdots, d\}$ is the subset of sensors that are compromised. Intuitively, attack difficulty quantifies how well the uncompromised sensors should be able to predict the compromised sensors. On the one extreme, if all the sensors are completely independent of one another (*i.e.*, the mutual information is zero), then there is little hope of detecting an adversary who can mimic the marginal distributions via looping. On the other extreme, if all the sensors are equal to each other (*i.e.*, the mutual information is maximum), then even one observation could be used to determine which sensors are compromised (*i.e.*, any sensor that does not match the majority of other sensors).

Method Details. We compare three of the methods for estimating the expected conditional distance in Def. 3. First, we consider the model-free method based on KNN (§ 2) for estimating the conditional distributions paired with the non-parametric Kolmogorov-Smirnov (KS) statistic (denoted by "KNN-KS" in our results). To compare the model-free and model-based methods while keeping the statistic the same, we also consider a model-based method using a multivariate Gaussian to estimate both p and q and then computing the KS statistic by sampling $1,000$ samples from each model. Finally, we consider a model-based method using a multivariate Gaussian using the Fisher-divergence test statistic based on the score function (denoted "Score" or "MB-SM"). For the expectation over $\boldsymbol{x}_{-j}$ in Def. 3, we use 30 samples from both X_{-j} and Y_{-j} to empirically approximate this expectation. For all methods, we use bootstrap sampling to approximate the sampling distribution of the test statistic γ for each of the methods above. In particular, we bootstrap B two-sample datasets $\{(X^{(b)}_{\text{boot}}, Y^{(b)}_{\text{boot}})\}_{b=1}^{B}$ from the concatenation of the input samples X and Y, *i.e.*, simulating the null hypothesis. For each b, we fit our model on $\{(X^{(b)}_{\text{boot}}, Y^{(b)}_{\text{boot}})\}^{(b)}$ (for Gaussians the fitting is fairly simple, but for deep density models we could update our model (1-2 epochs) with each b rather than retraining), and estimate the expected conditional distance $\hat{\gamma}$ for all features (*i.e.*, $\hat{\gamma} = [\hat{\gamma}_1, \hat{\gamma}_2, \cdots, \hat{\gamma}_d]$). We set the target significance level to $\alpha = 0.05$ as in [25] (note: this is for the detection stage only; the significance level for the localization stage is not explicitly set). For multiple sensors, we use micro precision and recall which are computed by summing the feature-wise confusion matrices and then computing precision and recall, e.g., $\text{Prec}_{\text{micro}} = \frac{\sum_{i=1}^{d} \text{TP}_i}{\sum_{i=1}^{d} \text{TP}_i + \sum_{i=1}^{d} \text{FP}_i}$.

3.1 Simulated Experiments

Simulation via Gaussian copula graphical models. We generate data according to a Gaussian copula graphical model, denoted by $\mathcal{C}_{\mathcal{N}}(\boldsymbol{\mu}, \Sigma)$ where the non-zeros of Σ^{-1} are the edges in the graphical model, paired with Beta marginal distributions with shape parameters $a = b = 0.5$. Note that the conditional dependency structure, i.e., the graphical model, and mutual information are the same as the corresponding multivariate Gaussian. However, the copula with Beta marginals distribution is very non-Gaussian since the Beta distribution is multimodal because the shape parameters a and b are both less than one. Thus, our model-based methods are not given significant advantage since $\mathcal{C}_{\mathcal{N}}$ has bimodal marginals with high densities on the edges of the distribution; which does not match the our assumed Gaussian distribution. We simulate relationships between 25 sensors in a real-world network using four graphical models: a complete graph, a chain graph, a grid graph, and a random graph using the Erdos-Renyi graph generation model with the probability of an edge set to 0.1 (illustration of the graph types in the Appendix). We set all the edge weights to be equal to a constant. We adjust this

constant so that the attack difficulty for the middle sensor (*e.g.*, the mutual information between sensor 12 and all other sensors) is fixed at a particular value for each of the graphs to make them comparable. Note that this is possible to compute in closed-form for multivariate Gaussian distributions—and Gaussian-copula distributions have equivalent mutual information because mutual information is invariant under invertible transformations. We select mutual information of $\{0.2, 0.1, 0.05, 0.01\}$ based on preliminary experiments that explore the limits of each method (*i.e.*, where they work reasonably well and where they begin to breakdown).

Fixed single sensor. We first investigate the simplest case where a user wants to know if a specific sensor in a network is compromised. For simulation, the first set of samples, *i.e.*, X, is assumed to be samples from the clean reference distribution p. However, Y has two possible states: if q is not compromised, then Y also comes from the clean reference distribution, *i.e.*, $q = p$, but if q is compromised, we assume that the samples come from a marginal distribution attack on sensor j, i.e., $q = \text{MarginalAttack}(p, j)$. We ran this experiment for all four graphs and different attack difficulties with 600 replications (300 with j-th sensor compromised, 300 without compromise) using three random seeds (100 samples per case per seed). In the appendix, we give more results, and we can see that the score-based method outperforms the other two methods in terms of precision and recall and is also computationally most efficient. The best recall of 0.211 for 0.01 mutual information shows that we are testing the limits of our methods and detecting this attack is quite difficult.

Unknown single sensor. In this experiment, the user does not know *a priori* which single sensor is compromised, and thus the compromised sensor (if any) needs to be identified. We use the same simulation setup as in the fixed single sensor experiment. However, our detection method is split into two stages we call the *detection stage* and *localization stage* respectively. First, in the *detection stage*, we attempt to determine if any sensor is compromised. Second, for the *localization stage*, if the first stage detects an attack, we then attempt to localize which sensor was attacked. For the first stage, we report network has been compromised if *any* feature-level test is signficant at the level $\alpha_{\text{bon}} = \frac{\alpha}{d}$, where d is the number of sensors and α_{bon} is the conservative Bonferroni correction for multiple tests [3] which does not make any assumptions about the dependencies between tests. For the second stage (*i.e.*, if an attack is detected by stage one), we choose the sensor with the highest estimated expected conditional distance, *i.e.*, $\hat{j} = \arg\max_j \hat{\gamma}_j$. The positive class for computing precision and recall is defined as detecting an attack *and* identifying the correct sensor (*e.g.*, the method fails if it detects an attack but does not localize the correct sensor). In Table 1, we can see that the score method had the best recall in all cases with very similar precision values as other methods. The higher precision methods (sometimes MB-KS and sometimes KNN-KS) have much lower recall, suggesting that these methods would miss many attacks. The score-method performs almost equal in precision to the model-based KS method, while having both better recall and 9 times less the computational time. Given the significant computational expense of the model-free KNN method and its overall poor performance compared to the model-based methods, we do not include the model-free KNN method in the other experiments.

Table 1: Top: Precison and recall vs. mutual information with each divergence method for localizing the compromised single sensor, averaged over all four graphs. Marginal-KS refers to the state-of-the-art [25]. Bottom: wall-clock run time (seconds) per test.

	Unknown Single Sensor							
	MB-SM		**MB-KS**		**KNN-KS**		**Marginal-KS**	
MI	**Pre**	**Rec**	**Pre**	**Rec**	**Pre**	**Rec**	**Pre**	**Rec**
0.2	0.99	**0.86**	0.99	0.73	**1.00**	0.55	0.00	0.00
0.1	0.99	**0.79**	**0.99**	0.58	0.87	0.18	0.50	0.00
0.05	0.99	**0.43**	0.98	0.30	**1.00**	0.03	0.25	0.00
0.01	0.86	**0.03**	**0.89**	0.02	0.62	0.00	0.25	0.00
Time	**0.0068**		0.0608		0.2404		0.008	

Unknown multiple sensors (Attack strength). We consider the more complex case where multiple sensors could be compromised, *i.e.*, exploring the attack strength defined above. We assume we have some fixed budget of sensors that can be checked or verified, denoted by k, and we are seeking to identify which k sensors have been compromised. We use the same setup as in the single sensor attack but attack three sensors via the marginal attack method. Similar to the unknown single sensor case, we proceed in a two stage fashion. The first stage is the same as the single sensor case where we use the Bonferroni correction to detect whether an attack has occurred. For the second stage, if an attack is predicted, the set of compromised sensors is predicted to be the top k estimated statistics, i.e., $\widehat{\mathcal{A}} = \arg\max_{\{\mathcal{A}:\ |\mathcal{A}|=k\}} \sum_{j\in\mathcal{A}} \hat{\gamma}_j$—this is a simple generalization of the single sensor case. In the

appendix, we show the precision and recall for the first detection stage for $k = 3$. It can be seen that with three compromised sensors, both the MB-SM and MB-KS perform well in stage one precision until a mutual information of 0.01, with the MB-SM again performing better in recall. In Table 2, we show the micro precision and recall for the second localization stage for $k = 3$. As expected, the localization task (stage two) is significantly more difficult than merely identifying if an attack has occurred, with precision and recall usually less than 0.75. Yet again, we see that the score method outperforms MB-KS in general—suggesting that the score method is generally better for this task.

Table 2: Precision and Recall of the compromised sensor localization results using MB-SM or MB-KS with three attacked sensors.

Graph:	Complete				Cycle				Grid				Random			
Method:	MB-SM		MB-KS		MB-SM		MB-KS		MB-SM		MB-KS		MB-SM		MB-KS	
MI \Metric:	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
0.2	**0.65**	**0.72**	0.53	0.59	**0.75**	**0.80**	0.63	0.69	**0.72**	**0.78**	0.65	0.72	0.41	0.46	**0.49**	**0.48**
0.1	**0.63**	**0.66**	0.49	0.51	**0.64**	**0.68**	0.57	0.62	**0.68**	**0.71**	0.60	0.61	**0.50**	0.56	0.48	**0.57**
0.05	**0.50**	**0.47**	0.46	0.37	**0.55**	**0.51**	0.50	0.46	**0.54**	**0.45**	0.48	0.32	**0.53**	**0.50**	0.50	0.46
0.01	**0.25**	**0.06**	0.10	0.03	**0.20**	**0.04**	0.17	0.03	**0.17**	**0.04**	0.16	0.03	**0.37**	**0.07**	0.21	0.05

Detecting and localizing time-series attack. A strong use case for feature shift detection is detecting shifts in a time series. This involves not only finding which features have shifted, but also identifying *when* the shift happened. For this experiment, we sample according to our Gaussian copula model to produce a time series, which acts as a time series where the samples are temporally independent. As in the other experiments, we fix $X \sim p$ to be our clean reference distribution, but now $Y \sim \mathcal{W}_K$ where $\mathcal{W}$ is a sliding window with a window size of K. We perform a stage one detection test sequentially on the time series every 50 samples (*i.e.*, each step is 50 samples). We choose 50 samples for simulation convenience but there is an inherent tradeoff between fidelity and computational time that could be explored. If an attack is detected at a time point, then $\mathcal{A}$ is predicted to be the k greatest test statistics, $\hat{\gamma}_j$. This continues until the datastream has been exhausted of samples. For each experiment we compute a time-delay for the first detection, which we define to be $t_\Delta = t_{\text{det}} - t_{\text{comp}}$ where t_{comp} is the first time step where a compromised sample enters $\mathcal{W}_K$, and t_{det} is the step when an attack is detected. The first stage detection results can be seen in Table 4, where the shorter window-sizes (*e.g.*, $K = 200, 400$) have the highest precision, recall, and lowest delay for graphs other than random. However, in Table 3, it can be seen that the larger window sizes have better precision and recall in the second stage detection: finding which sensors are in $\mathcal{A}$. This highlights the trade-off between accuracy of first stage detection and that of second stage localization.

Table 3: Precision and Recall of localizing which sensors are compromised using the score method with different window sizes and three attacked sensors.

	Complete		Cycle		Grid		Random	
Window	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec
200	0.30	0.19	0.76	0.71	0.56	0.24	0.49	0.34
400	0.57	0.53	0.58	0.56	0.60	0.51	0.50	0.33
600	0.60	0.55	**0.87**	**0.82**	**0.78**	**0.71**	0.36	0.35
800	0.66	0.61	0.72	0.68	0.70	0.64	**0.66**	**0.64**
1000	**0.72**	**0.66**	0.85	0.81	0.68	0.62	0.44	0.44

Table 4: Precision, Recall, and Time-Delay of localizing an attack using MB+SM with different window sizes and $|\mathcal{A}| = 3$ (i.e. three sensors are compromised).

Graph:	Complete			Cycle			Grid			Random		
Window	Prec	Rec	Delay	Prec	Rec	Delay	Prec	Rec	Delay	Prec	Rec	Delay
200	**0.99**	0.64	3.33	**0.99**	0.94	2.67	**0.98**	0.42	5.00	0.96	0.69	9.33
400	0.96	**0.94**	**3.00**	0.93	**0.97**	**1.67**	0.95	0.85	**3.00**	0.94	0.65	16.67
600	0.93	0.92	3.67	0.84	0.95	2.67	0.90	0.90	4.67	0.97	0.96	2.33
800	0.95	0.92	4.00	0.92	0.95	2.33	0.95	0.91	4.67	**0.99**	0.97	1.67
1000	0.97	0.92	4.33	0.89	0.96	2.67	0.91	**0.91**	5.00	0.95	**0.98**	**1.00**

3.2 Experiments on Real-World Data

While our extensive simulation experiments were meant to demonstrate the feature-shift detection task, we now present some experiments on some real-world data. We present results on the UCI Appliance Energy Prediction dataset [4], UCI Gas Sensor Array Drift dataset [10], and the number of new deaths from COVID-19 for the 10 states with the highest total deaths as of September 2020, measured by the CDC [1] (more dataset and experiment details can be found in the Apendix). Given the complex time dependencies among the data, we introduce a time-dependent bootstrapping method, "Time-Boot". Time-Boot subsamples random contiguous chunks from clean held out data (in real-world situations, this could be data which has already passed shift testing) to be the bootstrap

Table 5: Detection (left) and localization (right) results for the UCI Appliance Energy Prediction, UCI Gas Sensor Array Drift, and CDC's COVID-19 datasets, (top, middle, and bottom, respectively).

Time Axis	MB-SM		MB-SM-Time-Boot		Deep-SM-Time-Boot		MB-SM		MS-SM-Time-Boot		Deep-SM-Time-Boot	
	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall
	Feature shift detection (1st stage)						*Feature shift localization (2nd stage)*					
Energy												
Unshuffled	50.00%	**100%**	16.67%	8.86%	**55.00%**	62.62%	1.92%	**100%**	2.38%	1.27%	**3.00%**	3.80%
Shuffled	74.57%	**97.74%**	75.25%	96.20%	**77.89%**	93.67%	2.87%	**97.74%**	**75.25%**	96.20%	64.21%	77.22%
Gas												
Unshuffled	**50.00%**	**100%**	11.11%	2.27%	22.22%	4.55%	8.85%	**100%**	**11.11%**	2.27%	**11.11%**	2.27%
Shuffled	97.30%	**100%**	75.86%	**100%**	**97.78%**	**100%**	51.43%	68.35%	56.90%	**75.00%**	**68.90%**	70.45%
COVID-19												
Unshuffled	**50.00%**	**100%**	0.00%	0.00%	0.0%	0.0%	**6.96%**	**13.92%**	0.00%	0.00%	0.00%	0.00%
Shuffled	66.67%	**88.61%**	75.00%	82.76%	**76.00%**	65.52%	**51.43%**	**68.35%**	40.62%	44.83%	36.00%	31.03%

distribution, as this accounts for time dependencies between data samples. In Table 5, we show our unknown sensor experiments with the original bootstrap ("MB-SM") and Time-Boot with the three datasets as well as results where the ordered samples are shuffled throughout the whole dataset ("Time Axis Shuffled"), which creates a semi-synthetic dataset that breaks time dependencies between points, while keeping the real-world feature dependencies intact. The need for the time dependent bootstrap can be seen in the results of "MB-SM" Unshuffled as the 100% recall and 50% precision shows that the the method always predicts a shift. This is expected because the time-dependencies mean the data distribution has a natural (*benign*) shift over time. This highlights the inherent difficulty of detecting *adversarial* shifts in the presence of natural shifts: it is difficult (if not impossible in certain cases) to determine if a shift is benign or adversarial without further assumptions. Because our solution, MB-SM Time-Boot, takes into account some of these time-dependent shifts, it causes our detection threshold to be much higher, and thus leads to few detections (very low recall) as these adversarial shifts are hidden among the natural shifts. One possible solution to this would be to estimate time-dependent density models (e.g., autoregressive time models); however, exploration of time-dependent density models is outside the scope of this paper.

In Table 5 it can be seen that using a deep density model can improve the performance of our Time-Boot method. For the deep density model, we fit a normalizing flow using iterative Gaussianization [13, 15] as it is fast and stable because it only requires traditional PCA and histogram algorithms rather than end-to-end backpropagation. While recall is slightly reduced, our deep density method ("Deep-SM") improves the precision of both detection and localization compared to the Gaussian-based method ("MB-SM") for the Energy and Gas datasets, even when the time axis is unshuffled. For the COVID-19 data, since the data has strong benign shifts, our time aware bootstrapping method seems to set the detection threshold too high to detect the adversarial ones, and thus leads to 0% detection and consequently localization in the unshuffled case. Clearly, other deep density models including more general normalizing flows or autoregressive models could be used but we leave extensive comparisons to future work.

4 Discussion and Conclusion

In this paper, we introduced the problem of localizing the cause of a distribution shift to a specific set of features, which corresponds to a set of compromised sensors. We formalized this problem as performing multiple conditional hypothesis tests of the distribution in question. We develop both model-free and model-based approaches, including one using a novel score-function statistic based on Fisher divergence. Our hypothesis test using this statistic performs both shift detection and localization using the model's gradient with respect to the input (which is usually already available for machine learning models). Using a realistic attack model, we then performed extensive experiments with simulated data and show the state-of-the-art performance of the score function even with low levels of mutual information between features. Moreover, our results on three real-world datasets illuminate the difficulties of detecting adversarial shift in the presence of natural shifts in a time-series dataset, even when using a deep density model. Further experiments are needed to see how how to properly model time-dependent behaviors so that anomalous changes can be differentiated from natural time-based changes.

We hope our work suggests multiple natural questions and extensions such as using an autoregressive model as seen in [7] to take better account for time-series data, or using density models that perform well in small data regimes thus allowing for better performance with smaller window sizes leading to faster detection. Another natural question is: if we have detected and even localized a shift, what do we do with this information? For example, automatic mitigation or validation checks for the localized feature shifts. More broadly, the question this poses is can we characterize shifts beyond localization to give more information to system owners?

Broader Impact

Positive implications of our technology. Our solution allows the deployment of sensor networks, including large-scale ones, with trust placed on the values obtained from the network. This can happen despite the presence of sophisticated adversaries, such as, an adversary that can observe and faithfully reproduce the values of some sensors. Such trust now opens up the possibility of deploying sensor networks in critical operations, such as, monitoring dangerous terrain (e.g., for IEDs), monitoring large industrial plants (e.g., for noxious gases), and monitoring industrial control systems through digital PLC controllers (e.g., for vibrations of motors). The huge volume of sensor network research has had challenges in translating to practical impact and one of the well-identified concerns has been that the networks are easy to compromise, thus eroding trust in their operation. Our work can start to address this concern.

Due to the fact that the compromised sensors can be localized, our work sheds some light on the reason behind its detection, allowing mitigation actions. This gets at the more explainable nature of ML models. Due to the scalable nature of our solution (e.g., the score metric computation is lightweight), it is applicable to large-scale sensor networks.

Negative implications of our technology. An adoption of our solution without careful understanding of the adversary capabilities may lead to a false sense of confidence. For example, a sophisticated adversary may be able to compromise multiple correlated sensors and change their values in a correlated manner so as to defeat our defense. Generally, reasoning about such correlated attacks needs a higher degree of sophistication and is called upon prior to the use of our technology.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work was provided by: Northrop Grumman Corporation through the Northrop Grumman Cybersecurity Research Consortium (NGCRC), Army Research Lab through Contract number W911NF-2020-221, and National Science Foundation through their SaTC program, grant number CNS-1718637. There are no competing interests associated with this work.

References

[1] United states covid-19 cases and deaths by state over time. URL `https://data.cdc.gov/Case-Surveillance/United-States-COVID-19-Cases-and-Deaths-by-State-o/9mfq-cb36`.

[2] C. Alippi and M. Roveri. Just-in-time adaptive classifiers—part i: Detecting nonstationary changes. *IEEE Transactions on Neural Networks*, 19(7):1145–1153, 2008.

[3] J. M. Bland and D. G. Altman. Multiple significance tests: the bonferroni method. *Bmj*, 310(6973):170, 1995.

[4] L. M. Candanedo, V. Feldheim, and D. Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, 140:81–97, 2017.

[5] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[6] J. Gama, I. vZliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[7] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. Deep autoregressive networks. volume 32 of *Proceedings of Machine Learning Research*, pages 1242–1250, Bejing, China, 22–24 Jun 2014. PMLR. URL `http://proceedings.mlr.press/v32/gregor14.html`.

[8] A. Gretton. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

[9] D. Hendrycks, M. Mazeika, and T. G. Dietterich. Deep anomaly detection with outlier exposure. *International Conference on Learning Representation (ICLR)*, abs/1812.04606, 2019. URL `http://arxiv.org/abs/1812.04606`.

[10] R. Huerta, T. Mosqueiro, J. Fonollosa, N. F. Rulkov, and I. Rodriguez-Lujan. Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring. *Chemometrics and Intelligent Laboratory Systems*, 157:169–176, 2016.

[11] A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

[12] A. Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5): 2499–2512, 2007.

[13] D. I. Inouye and P. Ravikumar. Deep density destructors. In *International Conference on Machine Learning (ICML)*, pages 2172–2180, jul 2018.

[14] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[15] V. Laparra, G. Camps-Valls, and J. Malo. Iterative Gaussianization: From ICA to random rotations. *IEEE Transactions on Neural Networks*, 22(4):537–549, 2011.

[16] A. Liu, J. Lu, and G. Zhang. Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[17] S. Liu, T. Kanamori, W. Jitkrittum, and Y. Chen. Fisher efficient inference of intractable models. In *Advances in Neural Information Processing Systems*, pages 8790–8800, 2019.

[18] V. Losing, B. Hammer, and H. Wersing. Knn classifier with self adjusting memory for heterogeneous concept drift. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 291–300. IEEE, 2016.

[19] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[20] N. Lu, J. Lu, G. Zhang, and R. L. De Mantaras. A concept drift-tolerant case-base editing technique. *Artificial Intelligence*, 230:108–133, 2016.

[21] S. Lyu. Interpretation and generalization of score matching. *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 359–366, 2009.

[22] J. Ma and S. Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618, 2003.

[23] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *NIPS*, 2017.

[24] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[25] S. Rabanser, S. Günnemann, and Z. Lipton. Failing loudly: an empirical study of methods for detecting dataset shift. In *Advances in Neural Information Processing Systems*, pages 1394–1406, 2019.

[26] P. Sánchez-Moreno, A. Zarzo, and J. S. Dehesa. Jensen divergence based on fisher's information. *Journal of Physics A: Mathematical and Theoretical*, 45(12):125305, 2012.

[27] M. Yamada, A. Kimura, F. Naya, and H. Sawada. Change-point detection with feature selection in high-dimensional time-series data. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[28] Y. Yang, R. Martin, and H. Bondell. Variational approximations using fisher divergence. *arXiv preprint arXiv:1905.05284*, 2019.

[29] S. Yu, X. Wang, and J. C. Principe. Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3033–3039, 2018.

[30] S. Yu, M. Drton, and A. Shojaie. Generalized score matching for non-negative data. *Journal of Machine Learning Research*, 20(76):1–70, 2019.

[31] H. Zenati, M. Romain, C. Foo, B. Lecouat, and V. Chandrasekhar. Adversarially learned anomaly detection. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 727–736, Nov 2018. doi: 10.1109/ICDM.2018.00088.

A Experiment Details

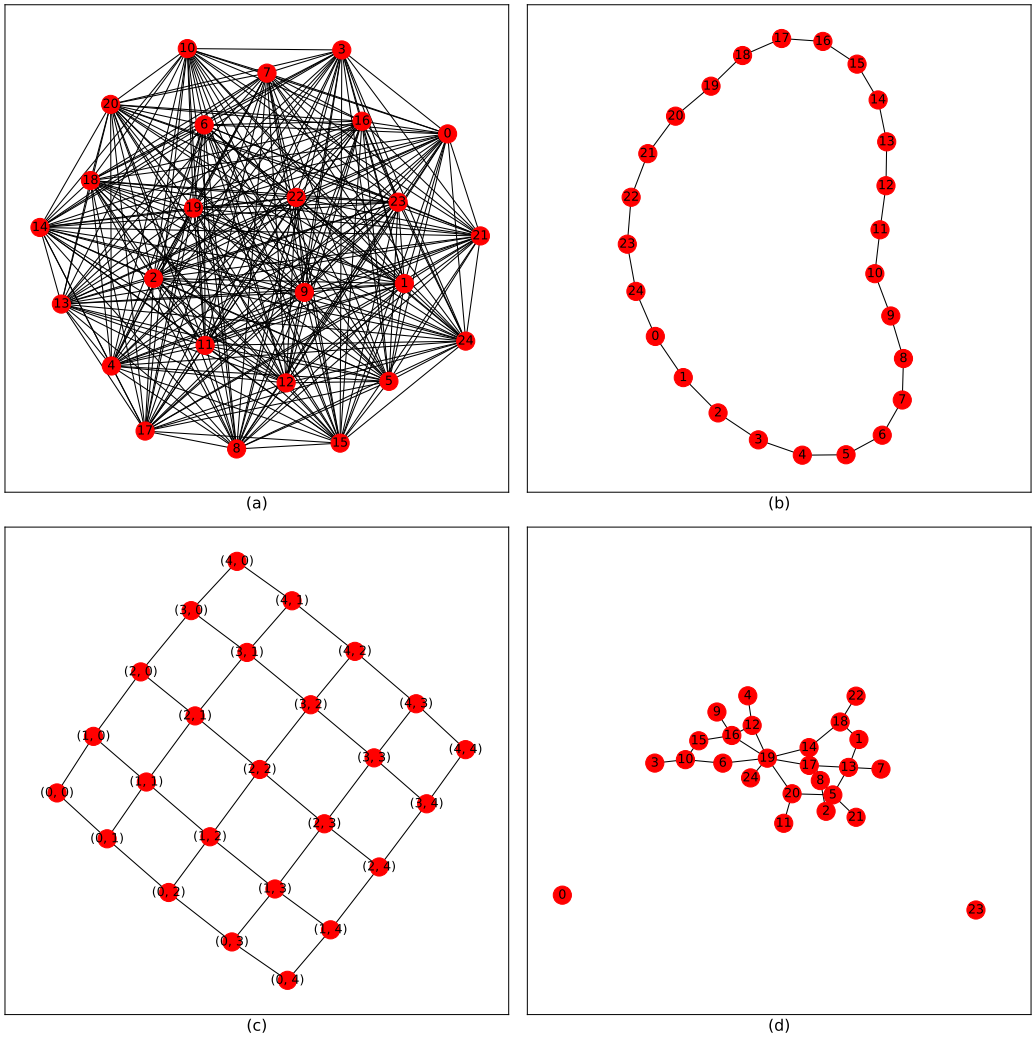


Figure 3: Graphical models used for generating simulated data. (a) complete: all nodes directly dependent, (b) cycle: only adjacent nodes are directly dependent, (c) grid: other than edge nodes, all nodes have direct dependency with four other nodes, and (d) random (Erdos-Renyi): dependencies between nodes randomly generated (note: the random graph changes with every seed).

A.1 Fixed Single Sensor

This experiment simulates where a users wants to know if a specific sensor in a network is compromised.Each experiment was run with MB-SM, MB-KS, KNN-KS, and Marginal-KS methods, for each of the four graph types, and each with decreasing mutual information, $MI \in \{0.2, 0.1, 0.05, 0.01\}$. These MI values were chosen via the breakdown points from previous empirical results (from where the models start performing poorly (0.2) to where the models perform very poorly (0.01)). Each experiment was ran with 600 replications (300 with j-th sensor compromised, 300 without compromise) using three random seeds (0, 1, 2). In each experiment, the j-th sensor which is attacked or not attacked is chosen randomly per replication, and detection success is measured solely on the chosen sensor in each replication. The results can be seen in Table 6 where it can be seen that the MB-SM method has the greatest recall for each MI. The Marginal-KS method has the greatest precision, but with a near zero recall. Excluding the Marginal-KS method, the MB-SM and MB-KS methods have similar levels of precision for each MI, with the MB-SM being the method with the lowest

computational time of them all. In each experiment the computational time is approximated to be $\frac{t_\gamma}{d}$ where t_γ is the time to calculate the threshold, γ, for all sensors, and d is the number of sensors. This assumption is reasonable since in practice only γ_j needs to be calculated, thus dividing γ's computational time by d.

A.2 Unknown Single Sensor

This experiment explores the task of detecting if a sensor is compromised and if so, localizing which sensor has been taken over. The setup of the experiment is the same as the fixed sensor case, but with a different detection method, and the KNN-KS and Marginal-KS are not included due to expensive computational costs combined with low performance. The detection method consists of first seeing if any γ_j is significant at $\alpha_{bon} = \frac{\alpha}{d}$ (i.e. if $\exists \gamma_j > \gamma_{0_j}$ where γ_{0_j} is the $1 - \alpha_{bon}$ percentile of the j-th feature's conditional distance found from the bootstrapping step and α_{bon} is the Bonferroni correction for multiple tests). If any of the feature-level tests satisfy this condition, then a 'global attack' is detected. The attack is localized to a specific sensor by finding the sensor with the highest estimated conditional distance (i.e. $\hat{j} = \arg\max_j \hat{\gamma}_j$). The results for the localization of the unknown sensor can be seen in Table 1.

A.3 Unknown Multiple Sensors

Here we consider the more complex case where multiple sensors could be compromised. We assume we are seeking to identify which k sensors have been compromised. The setup for this experiment is the same as the unknown single sensor case, except we explore $k = \{2, 3, 4, 5\}$, and like-wise if an attack is detected, localize the compromised sensors with $|\mathcal{A}| = k$ (i.e. the top k sensors with the highest conditional distance are predicted to be compromised). When a set of sensors are attacked, their compromised outputs sample from the joint marginal rather than from each individual sensor's marginal distribution. This is a much smarter attack, and can be more difficult to detect. The results are measured in two ways, the first is the detection results (i.e. detecting if a 'global attack' has occurred), and the second is localization results (i.e. if a 'global attack' was detected, how many of the predicted compromised sensors in $\hat{\mathcal{A}}$ actually compromised). This is done by computing a confusion matrix for each feature, and then summing along the feature axis (i.e. micro precision and recall). The detection results for $k = \{2, 3, 4, 5\}$ can be seen in Table 7 and Table 8 where it can be seen that the MB-SM outperforms the MB-KS in precision, recall, and as seen in the previous results, computational complexity. It can also be seen that there are slight improvements in detection as the number of compromised sensors increases, which makes sense as this means an attack should be easier to detect. The localization results can be seen in Table 9 and Table 10. Again the MB-SM outperforms the MS-KS in every category, except for the random graphs in which the MB-KS has higher precision and recall. The results show that as k grows, the ability to localize which sensors are in $\mathcal{A}$ diminishes for both MB-KS and MB-SM. This makes sense as more and more conditional information is being destroyed (e.g. in the case of $d = 2$ if one sensor is compromised, it would be impossible to detect which is acting anomalously without any prior knowledge.)

A.4 Detecting and localizing time-series attack

This experiment is to simulate detecting and localizing feature shift in time-series data. To do so, we sample 10,000 samples according to our Gaussian copula model to produce a time series where the samples are temporally independent. We set $X \sim p$ as our reference distribution. $Y \sim \mathcal{W}_\mathcal{K}$ is our query distribution where $\mathcal{W}_\mathcal{K}$ is a window of size $K = |X| = |Y|$ (unrelated to k compromised sensors). $\mathcal{W}_\mathcal{K}$ slides over the time-series data with a step size of 50 (i.e. each step updates 50 new samples into and the 50 oldest samples out of the window), which was chosen as it is large enough to speed up calculations of the whole time series, but not too large that detection is strongly affected. The experiment was run for all graphs with MI=0.5 (in time-series experiments this is a less direct metric of problem hardness), with $k = \{1, 2, 3\}$, and the attack happening at 80% of the time-series (samples 8,000+). It should also be noted that the random selection of $\mathcal{A}$ (which sensors are compromised) only happens once, and $\mathcal{A}$ stays constant for the entire time of the attack. Within each window, the detection process is the same for the unknown multiple sensors experiment, with an additional metric introduced: time-delay. This metric measures how many steps of $\mathcal{W}_\mathcal{K}$ are taken between the time when an compromised sample enters the window and a attack is detected. More formally, $t_\Delta = t_{\text{det}} - t_{\text{comp}}$ where t_{comp} is the first time step when a compromised sample enters $\mathcal{W}_K$,

and t_{det} is the step when an attack is detected. The detection results can be seen in Table 15, where it can be seen that larger window sizes have typically have better precision and recall when compared to smaller window sizes, however they also have longer time-delays (excluding $K = 200$ which can have very large time-delays due to poor detection overall). On average, the $K = 500$ has the best intersection of precision, recall, and time-delay. The localization results can be seen in Table 16 where again a clear trend of larger window-sizes ($K \geq 500$) have on average greater precision and recall than smaller window-sizes ($K < 500$). Choice of window-size will depend on the preferences of the user, but as a starting point, we suggest setting setting $K = 500$ for a balance of detection speed and localization accuracy. One thing we wish to explore further is how changing the step size affects these metrics as well.

A.5 Experiments on Real-World Data

While our extensive simulation experiments were meant to demonstrate the feature-shift detection task, we also performed more experiments on real-world data. Each experiment consisted of detecting whether a *non-benign* distribution shift has happened, and if so, localizing it to a specific feature. This was performed on three datasets, the UCI Appliances Energy Prediction dataset [4], the UCI Gas sensors for home activity monitoring Data Set [10], and the CDC's United States COVID-19 Cases and Deaths by State over Time [1] as of late Sepetember, 2020. For the COVID-19 data, we extracted the number of new deaths per day for the 10 states with the greatest total number of deaths ('MI', 'PA', 'IL', 'NY', 'MA', 'FL', 'TX', 'CA', 'NJ', 'NYC') (note: the dataset lists New York state, 'NY' and New York City, 'NYC' as separate entities), and used a 2nd order polynomial to interpolate between each day with a resolution of $\frac{1}{2}$ hour. This resulted in an upsampling to $\sim$ 10,000 samples.

We used two models for the experiments. The first was simply fitting X and Y to multivariate Gaussians using $\hat{p} = \mathcal{N}(\text{mean}(X), \text{var}(X))$ and likewise for $\hat{q}$. The second used a deep density model which fit a normalizing flow using iterative Gaussianization with the number of layers set to 2 for each X and Y. The score function was used with both models, but two variants of bootstrapping were used. The first was the classical form of bootstrapping as explained in section 3, we will denote this as 'Classic Boot' for the remainder of this section. The second was 'Time Boot' which randomly subsampled contiguous time-series from held out clean data and set (X, Y) to be the first and second half, respectively, of the subsampled data. More formally, $\left\{X^{(b)}_{boot}, Y^{(b)}_{boot}\right\}_{b=1}^{B} = \left\{T_{clean}[t^{(b)} - n : t^{(b)}], T_{clean}[t^{(b)} : t^{(b)} + n]\right\}_{b=1}^{B}$ where B = number of bootstrap runs, n is the number of samples in X and Y, T is the held out clean data, and $t^{(b)} \sim U[n, N_{clean} - n]$ is the $b^{(th)}$ split index uniformly sampled across the held out data (from n to $N_{clean} - n$ to avoid boundary conditions and assure $X^{(b)}$ and $Y^{(b)}$ are the same size). As a preprocessing step for both bootstrapping methods, a first order approximation of the gradient of the samples w.r.t. time (i.e. $X_{t_i} = X_{t_{i+1}} - X_{t_i}$) was taken on the concatenation of X and Y in order to lessen some of the time-dependencies. Next, a power-transform (using Sklearn's power-transform with the Yeo-Johnson and standardizing set to true) was performed on the concatenation in order to make the data more Gaussian-like. Finally, the respective model was fit on the transformed X and Y.

For experiments with 'Classic Boot' the number of trials was determined by $\lfloor \frac{N-2n}{s} \rfloor$ where N is the number of samples in the dataset, n is the number of samples in X and Y, and s is the step size for stepping through the dataset. For experiments with 'Time Boot', the number of trials was similarly determined after half of the data was removed for our global time-based bootstrapping. X and Y were set via a sliding window through the dataset with size $2n$ and with a step size s. For half of the trials the Marginal Attack was performed on a random feature on a local copy of Y, and for the other half of the trials no attack was performed. For the experiments with 'Classic Boot', $\hat{p}$ and $\hat{q}$ were re-fit with each new X, Y (i.e. each trial), and the hypothesis testing was performed with the threshold, γ specific to that trial. In these experiments, we set $B_{classic} = 50$ and $\alpha = 0.05$. Thus, due to time constraints the 'Classic-Boot' experiments were only performed using the score function with the multivariate Gaussian ('MB-SM'). For experiments with 'Time-Boot', a global threshold, γ_{global} was used for the hypothesis testing. The threshold was determined using the held out data with $B_{time} = 500$ and $\alpha = 0.05$. The results for these experiments can be seen in Table 5.

Table 6: Top: precision and recall vs. mutual information with each divergence method for fixed single sensor, averaged over all four graphs. Bottom: wall-clock run time (seconds) per test (* fixed single index time is approximate).

Fixed Single Sensor								
	MB-SM		**MB-KS**		**KNN-KS**		**Marginal-KS**	
MI	**Pre**	**Rec**	**Pre**	**Rec**	**Pre**	**Rec**	**Pre**	**Rec**
0.2	0.94	**0.98**	0.93	0.92	0.79	0.68	**1.00**	0.01
0.1	0.95	**0.95**	0.95	0.89	0.86	0.47	**1.00**	0.02
0.05	0.95	**0.80**	0.94	0.70	0.84	0.22	**1.00**	0.02
0.01	0.84	**0.21**	0.83	0.16	0.68	0.06	**1.00**	0.02
Time	**0.17**		1.52		6.01		0.20	

Table 7: Precision and Recall of the compromised sensor detection results using MB-SM with $k = \{2, 3, 4, 5\}$ compromised sensors (out of 25 sensors total).

MB-SM								
	Complete							
	2		3		4		5	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.909	1.000	0.909	1.000	0.909	1.000	0.909	1.000
0.1	0.945	0.967	0.946	0.997	0.946	1.000	0.946	1.000
0.05	0.925	0.740	0.937	0.893	0.941	0.957	0.942	0.980
0.01	0.716	0.160	0.743	0.183	0.740	0.180	0.787	0.233
	Cycle							
	2.000		3.000		4.000		5.000	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.935	1.000	0.935	1.000	0.935	1.000	0.935	1.000
0.1	0.946	0.983	0.946	0.997	0.946	1.000	0.946	1.000
0.05	0.935	0.770	0.942	0.873	0.947	0.947	0.947	0.947
0.01	0.655	0.120	0.689	0.140	0.732	0.173	0.750	0.190
	Grid							
	2.000		3.000		4.000		5.000	
MI	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0.2	0.917	0.997	0.917	1.000	0.917	1.000	0.917	1.000
0.1	0.938	0.963	0.939	0.980	0.940	0.987	0.940	1.000
0.05	0.907	0.647	0.921	0.777	0.928	0.853	0.928	0.863
0.01	0.651	0.137	0.707	0.177	0.703	0.173	0.722	0.190
	Random							
	2.000		3.000		4.000		5.000	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.900	0.990	0.901	1.000	0.901	1.000	0.901	1.000
0.1	0.883	0.983	0.884	0.993	0.885	0.997	0.885	1.000
0.05	0.952	0.790	0.957	0.893	0.959	0.937	0.960	0.960
0.01	0.860	0.123	0.903	0.187	0.922	0.237	0.929	0.263

Table 8: Precision and Recall of the compromised sensor detection results using MB-KS with $k = \{2, 3, 4, 5\}$ compromised sensors (out of 25 sensors total).

MB-KS								
	Complete							
	2.000		3.000		4.000		5.000	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.895	0.997	0.896	1.000	0.896	1.000	0.896	1.000
0.1	0.929	0.910	0.932	0.953	0.934	0.983	0.934	0.983
0.05	0.906	0.640	0.917	0.733	0.923	0.797	0.928	0.853
0.01	0.623	0.127	0.623	0.127	0.589	0.110	0.646	0.140
	Cycle							
	2.000		3.000		4.000		5.000	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.912	1.000	0.912	1.000	0.912	1.000	0.912	1.000
0.1	0.913	0.980	0.914	0.993	0.915	1.000	0.915	1.000
0.05	0.911	0.753	0.920	0.840	0.925	0.903	0.926	0.913
0.01	0.592	0.097	0.661	0.130	0.688	0.147	0.726	0.177
	Grid							
	2.000		3.000		4.000		5.000	
MI	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0.2	0.908	0.987	0.909	0.997	0.909	1.000	0.909	1.000
0.1	0.910	0.847	0.918	0.933	0.921	0.967	0.921	0.977
0.05	0.862	0.437	0.893	0.587	0.903	0.653	0.909	0.703
0.01	0.710	0.147	0.719	0.153	0.727	0.160	0.735	0.167
	Random							
	2.000		3.000		4.000		5.000	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.948	0.847	0.952	0.927	0.954	0.960	0.955	0.980
0.1	0.823	0.943	0.829	0.987	0.831	0.997	0.831	0.997
0.05	0.924	0.730	0.934	0.847	0.937	0.893	0.939	0.930
0.01	0.632	0.143	0.658	0.160	0.706	0.200	0.722	0.217

Table 9: Precision and Recall of the compromised sensor localization results using MB-SM with $k = \{2, 3, 4, 5\}$ compromised sensors (out of 25 sensors total).

MB-SM								
	Complete							
	2		3		4		5	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.756	0.832	0.657	0.722	0.586	0.644	0.542	0.596
0.1	0.702	0.718	0.632	0.666	0.595	0.628	0.556	0.588
0.05	0.569	0.455	0.502	0.479	0.498	0.507	0.470	0.489
0.01	0.216	0.048	0.252	0.062	0.243	0.059	0.301	0.089
	Cycle							
	2		3		4		5	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.832	0.890	0.754	0.807	0.669	0.716	0.630	0.674
0.1	0.707	0.735	0.647	0.681	0.613	0.648	0.576	0.609
0.05	0.587	0.483	0.556	0.516	0.540	0.540	0.529	0.529
0.01	0.209	0.038	0.202	0.041	0.239	0.057	0.263	0.067
	Grid							
	2		3		4		5	
MI	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0.2	0.810	0.880	0.724	0.789	0.665	0.725	0.619	0.675
0.1	0.731	0.750	0.681	0.710	0.646	0.678	0.620	0.659
0.05	0.596	0.425	0.544	0.459	0.529	0.487	0.524	0.487
0.01	0.103	0.022	0.173	0.043	0.196	0.048	0.213	0.056
	Random							
	2		3		4		5	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.473	0.520	0.418	0.464	0.405	0.449	0.404	0.449
0.1	0.545	0.607	0.502	0.564	0.482	0.543	0.465	0.526
0.05	0.592	0.492	0.535	0.499	0.525	0.513	0.513	0.513
0.01	0.360	0.052	0.376	0.078	0.351	0.090	0.358	0.101

Table 10: Precision and Recall of the compromised sensor localization results using MB-KS with $k = \{2, 3, 4, 5\}$ compromised sensors (out of 25 sensors total).

MB-KS								
	Complete							
	2		3		4		5	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.575	0.640	0.530	0.592	0.475	0.530	0.463	0.517
0.1	0.551	0.540	0.499	0.511	0.489	0.515	0.468	0.493
0.05	0.514	0.363	0.468	0.374	0.443	0.383	0.425	0.391
0.01	0.107	0.022	0.153	0.031	0.161	0.030	0.191	0.041
	Cycle							
	2		3		4		5	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.684	0.750	0.635	0.697	0.606	0.665	0.576	0.631
0.1	0.606	0.650	0.574	0.623	0.558	0.610	0.532	0.582
0.05	0.524	0.433	0.504	0.460	0.501	0.489	0.496	0.489
0.01	0.143	0.023	0.175	0.034	0.207	0.044	0.241	0.059
	Grid							
	2		3		4		5	
MI	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0.2	0.716	0.778	0.657	0.720	0.634	0.698	0.585	0.643
0.1	0.659	0.613	0.608	0.618	0.587	0.617	0.564	0.598
0.05	0.493	0.250	0.489	0.321	0.474	0.343	0.471	0.364
0.01	0.129	0.027	0.161	0.034	0.178	0.039	0.226	0.051
	Random							
	2		3		4		5	
MI	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0.2	0.547	0.488	0.497	0.483	0.490	0.493	0.471	0.484
0.1	0.512	0.587	0.486	0.578	0.463	0.556	0.449	0.539
0.05	0.530	0.418	0.509	0.461	0.476	0.453	0.457	0.452
0.01	0.206	0.047	0.210	0.051	0.226	0.064	0.249	0.075

Table 11: Full Precision and Recall Tabel for complete and cycle graphs with the detecting an attack using Score Method (SM) or Model-Based+Kolmogrov-Shmirnov (MB-KS) with three attacked sensors.

Graph:	Complete				Cycle			
Method:	SM		MB-KS		SM		MB-KS	
MI Metric:	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
0.2	**0.9091**	1.0000	0.8955	1.0000	**0.9346**	1.0000	0.9119	1.0000
0.1	**0.9462**	**0.9967**	0.9316	0.9533	**0.9462**	**0.9967**	0.9141	0.9933
0.05	**0.9371**	**0.8933**	0.9167	0.7333	**0.9424**	**0.8733**	0.9197	0.8400
0.01	**0.7432**	**0.1833**	0.6230	0.1267	**0.6885**	**0.1400**	0.6610	0.1300

Table 12: Full Precision and Recall Tabel for grid and random graphs with the detecting an attack using Score Method (SM) or Model-Based+Kolmogrov-Shmirnov (MB-KS) with three attacked sensors.

Graph:	Grid				Random			
Method:	SM		MB-KS		SM		MB-KS	
MI Metric:	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
0.2	**0.9174**	**1.0000**	0.9088	0.9967	0.9009	**1.0000**	**0.9521**	0.9267
0.1	**0.9393**	**0.9800**	0.9180	0.9333	**0.8843**	**0.9933**	0.8291	0.9867
0.05	**0.9209**	**0.7767**	0.8934	0.5867	**0.9571**	**0.8933**	0.9338	0.8467
0.01	0.7067	**0.1767**	**0.7188**	0.1533	**0.9032**	**0.1867**	0.6575	0.1600

Table 13: Full Precision and Recall of the combined univariate detection results for complete and cycle graphs using Score Method (SM) or Model-Based+Kolmogrov-Shmirnov (MB-KS) with three attacked sensors.

Graph:	Complete				Cycle			
Method:	SM		MB-KS		SM		MB-KS	
MI Metric:	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
0.2	**0.6566**	**0.7222**	0.5303	0.5922	**0.7539**	**0.8067**	0.6353	0.6967
0.1	**0.6319**	**0.6656**	0.4995	0.5111	**0.6466**	**0.6811**	0.5736	0.6233
0.05	**0.5023**	**0.4789**	0.4681	0.3744	**0.5564**	**0.5156**	0.5036	0.46
0.01	**0.2523**	**0.0622**	0.153	0.0311	**0.2022**	**0.0411**	0.1751	0.0344

Table 14: Full Precision and Recall of the combined univariate detection for grid and random graphs results using Score Method (SM) or Model-Based+Kolmogrov-Shmirnov (MB-KS) with three attacked sensors.

Graph:	Grid				Random			
Method:	SM		MB-KS		SM		MB-KS	
MI Metric:	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
0.2	**0.7238**	**0.7889**	0.6565	0.72	0.4184	0.4644	**0.4966**	**0.4833**
0.1	**0.6805**	**0.71**	0.6077	0.6178	**0.5025**	0.5644	0.4855	**0.5778**
0.05	**0.5441**	**0.4589**	0.489	0.3211	**0.5345**	**0.4989**	0.5086	0.4611
0.01	**0.1733**	**0.0433**	0.1615	0.0344	**0.3763**	**0.0778**	0.21	0.0511

Table 15: Precision and Recall of detecting which sensors are compromised using the score method with different window sizes (WS) and $k = \{1, 2, 3\}$.

MB-SM Detection									
	Complete								
| A |	**1**			**2**			**3**		
WS	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**
200	0.86	0.05	24.33	0.98	0.36	7.33	0.99	0.64	3.33
300	0.89	0.40	9.00	0.94	0.84	5.67	0.95	0.90	4.33
400	0.95	0.69	6.67	0.96	0.92	3.33	0.96	0.94	3.00
500	0.96	0.87	4.67	0.97	0.93	3.00	0.97	0.93	3.33
600	0.92	0.86	7.33	0.93	0.88	6.00	0.93	0.92	3.67
700	0.89	0.88	6.00	0.89	0.91	4.33	0.89	0.93	4.00
800	0.95	0.84	8.67	0.95	0.90	4.67	0.95	0.92	4.00
900	0.92	0.84	9.00	0.93	0.90	6.00	0.93	0.93	4.00
1000	0.96	0.86	8.67	0.96	0.88	7.00	0.97	0.92	4.33
	Cycle								
| A |	**1.00**			**2.00**			**3.00**		
WS	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**
200	0.98	0.38	17.33	0.99	0.86	2.67	0.99	0.94	2.67
300	0.84	0.49	5.33	0.91	0.93	3.00	0.91	0.95	2.33
400	0.93	0.86	2.67	0.93	0.96	1.67	0.93	0.97	1.67
500	0.85	0.90	4.00	0.85	0.94	3.00	0.86	0.95	2.33
600	0.83	0.92	4.00	0.83	0.94	3.33	0.84	0.95	2.67
700	0.80	0.91	4.00	0.80	0.96	2.33	0.80	0.96	2.33
800	0.92	0.93	3.00	0.92	0.95	2.67	0.92	0.95	2.33
900	0.88	0.91	5.00	0.88	0.94	3.33	0.88	0.94	3.33
1000	0.89	0.92	4.67	0.89	0.94	3.33	0.89	0.96	2.67
	Grid								
| A |	**1.00**			**2.00**			**3.00**		
WS	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**
200	0.92	0.09	32.00	0.97	0.24	8.67	0.98	0.42	5.00
300	0.57	0.03	22.67	0.90	0.20	15.33	0.97	0.60	4.67
400	0.91	0.50	3.33	0.95	0.88	3.33	0.95	0.85	3.00
500	0.97	0.89	5.00	0.97	0.89	5.00	0.97	0.93	3.33
600	0.82	0.46	21.67	0.89	0.85	7.00	0.90	0.90	4.67
700	0.76	0.51	9.00	0.84	0.83	8.00	0.85	0.92	3.67
800	0.93	0.60	17.67	0.95	0.82	9.33	0.95	0.91	4.67
900	0.92	0.82	8.67	0.93	0.91	5.00	0.93	0.91	4.67
1000	0.89	0.74	9.33	0.90	0.88	6.00	0.91	0.91	5.00
	Random								
| A |	**1.00**			**2.00**			**3.00**		
WS	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**	**Precision**	**Recall**	**time-delay**
200	0.96	0.65	15.33	0.96	0.67	10.33	0.96	0.69	9.33
300	0.96	0.65	16.00	0.96	0.65	16.00	0.97	0.87	2.00
400	0.89	0.33	32.33	0.94	0.65	16.67	0.94	0.65	16.67
500	0.70	0.33	33.67	0.88	0.98	1.00	0.88	0.98	1.00
600	0.96	0.61	11.67	0.97	0.94	3.00	0.97	0.96	2.33
700	0.96	0.65	18.67	0.97	0.94	2.67	0.98	0.98	1.00
800	0.98	0.60	21.33	0.99	0.95	2.33	0.99	0.97	1.67
900	0.96	0.98	1.00	0.96	0.98	1.00	0.96	0.98	1.00
1000	0.95	0.97	1.67	0.95	0.97	1.67	0.95	0.98	1.00

Table 16: Precision and Recall of localizing which sensors are compromised using the score method with different window sizes (WS) and $k = \{1, 2, 3\}$.

	Complete					
| A |	**1**		**2**		**3**	
WS	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
200	0.333	0.015	0.330	0.117	0.298	0.189
300	0.527	0.210	0.366	0.308	0.363	0.326
400	0.646	0.444	0.586	0.542	0.565	0.530
500	0.900	0.780	0.622	0.577	0.617	0.576
600	0.828	0.712	0.710	0.628	0.600	0.553
700	0.937	0.821	0.757	0.691	0.687	0.636
800	0.936	0.786	0.766	0.693	0.658	0.607
900	0.979	0.822	0.872	0.782	0.741	0.686
1000	0.955	0.817	0.827	0.731	0.717	0.661
	Cycle					
| A |	**1.000**		**2.000**		**3.000**	
WS	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
200	1.000	0.379	0.873	0.754	0.758	0.712
300	0.985	0.486	0.824	0.764	0.728	0.691
400	0.976	0.840	0.924	0.885	0.576	0.556
500	1.000	0.900	0.965	0.907	0.830	0.791
600	1.000	0.917	0.822	0.769	0.869	0.825
700	0.993	0.907	0.955	0.914	0.813	0.778
800	0.994	0.929	0.897	0.854	0.719	0.681
900	1.000	0.908	0.973	0.917	0.673	0.634
1000	0.994	0.911	0.809	0.764	0.849	0.811
	Grid					
| A |	**1.000**		**2.000**		**3.000**	
WS	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
200	0.917	0.083	0.719	0.174	0.560	0.237
300	0.000	0.000	0.500	0.098	0.538	0.324
400	0.861	0.431	0.783	0.691	0.596	0.509
500	1.000	0.887	0.677	0.600	0.676	0.631
600	1.000	0.462	0.799	0.676	0.780	0.705
700	0.988	0.506	0.956	0.796	0.738	0.679
800	1.000	0.601	0.938	0.771	0.702	0.639
900	1.000	0.816	0.788	0.716	0.665	0.607
1000	0.963	0.717	0.639	0.561	0.677	0.617
	Random					
| A |	**1.000**		**2.00**		**3.00**	
WS	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
200	0.500	0.326	0.489	0.330	0.487	0.336
300	1.000	0.652	0.750	0.489	0.583	0.507
400	1.000	0.326	0.750	0.490	0.500	0.326
500	1.000	0.327	0.500	0.490	0.447	0.438
600	0.926	0.564	0.500	0.471	0.362	0.346
700	0.500	0.327	0.673	0.636	0.369	0.362
800	0.980	0.589	0.541	0.515	0.665	0.645
900	0.667	0.655	0.500	0.491	0.333	0.328
1000	0.646	0.628	0.497	0.483	0.444	0.437