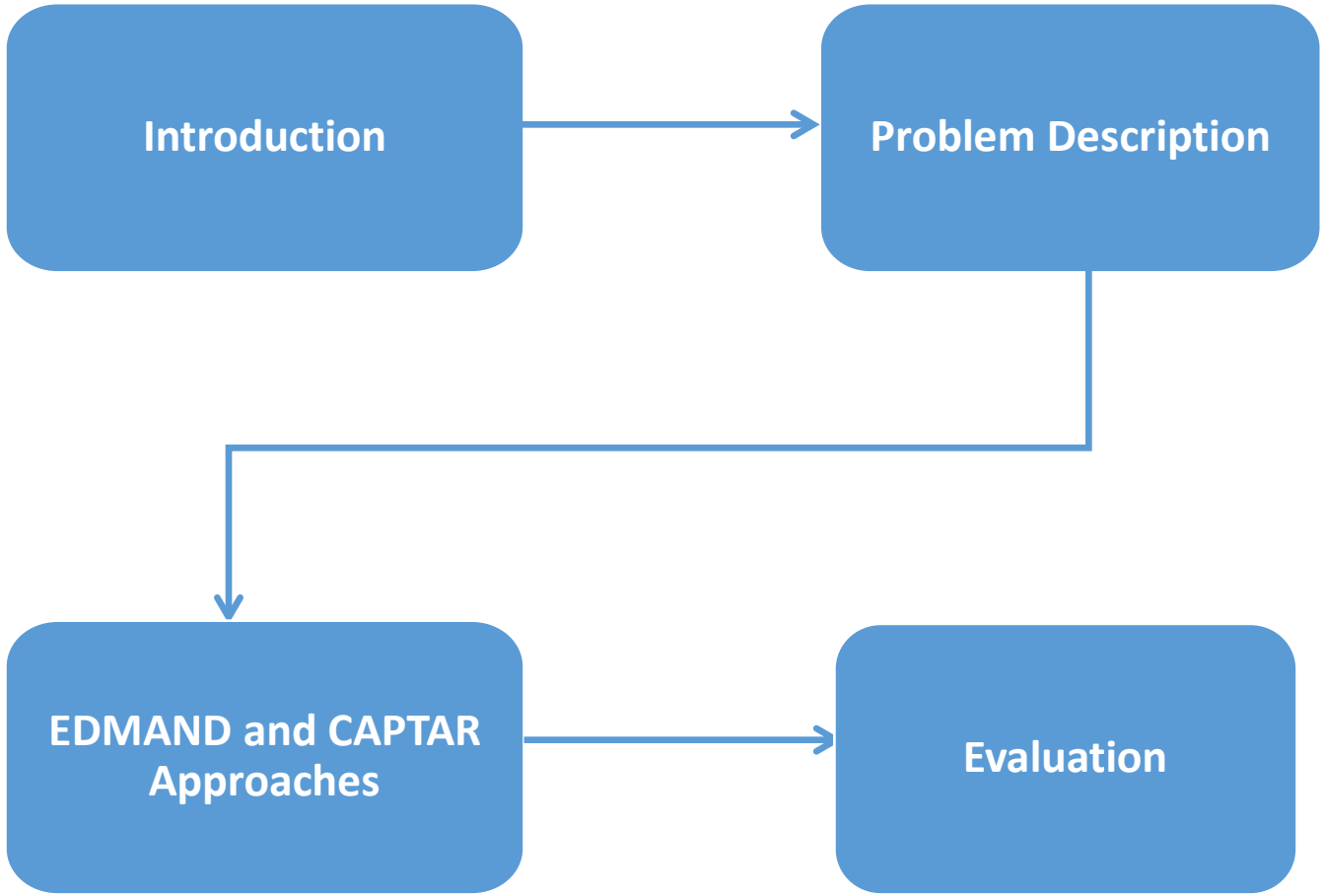
Anomaly Detection and Causal Reasoning about Attacks for SCADA Networks

Klara Nahrstedt

klara@illinois.edu

Joint Work with Wenuy Ren, Tuo Yu, Atul Bohara, Ghada Elbez, Al Valdes, Tim Yardley, William Sanders

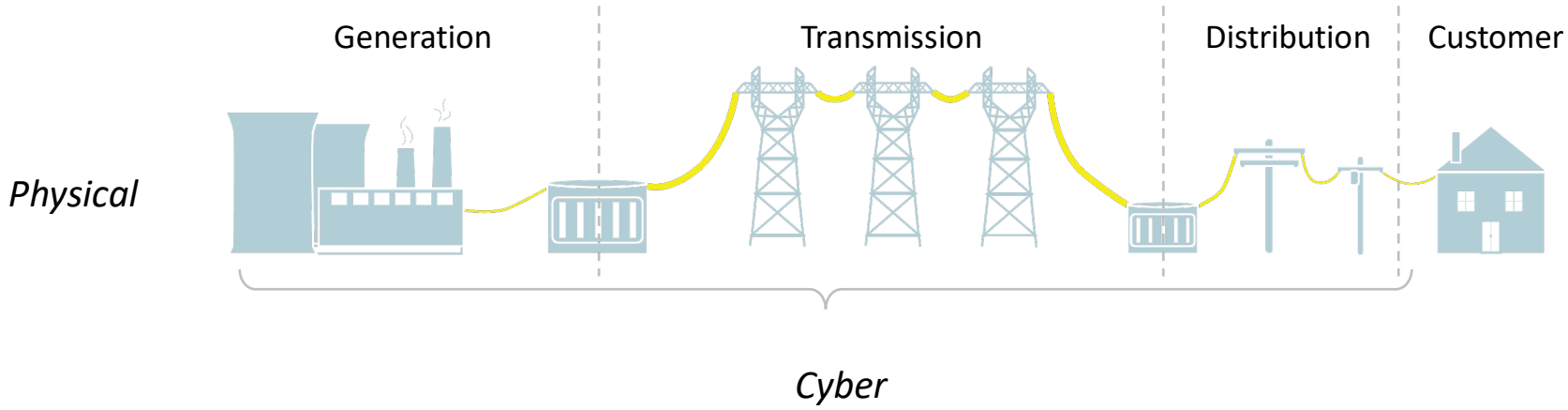
Outline


Introduction
Problem Description
EDMAND and CAPTAR Approaches
Evaluation

Introduction

Background

- Smart Grid: intelligent electric power



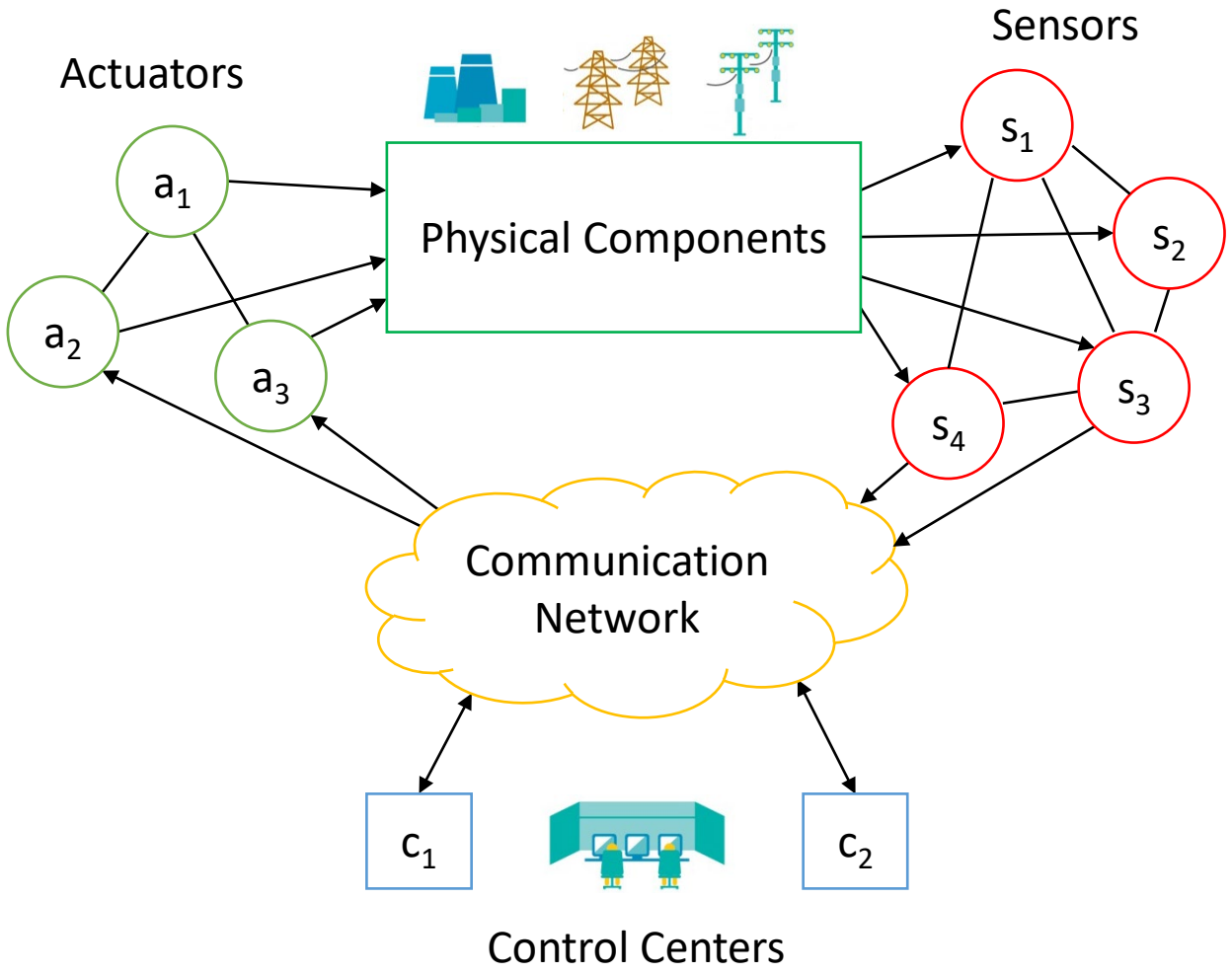
- Supervisory Control and Data Acquisition (SCADA)

  SCADA systems are industrial control systems (ICSs) used for real-time monitoring, data collection, and control for large-scale distributed critical infrastructure systems.

Introduction

Background

- Smart Grid
- Cyber-Manufacturing
- Other Mission-critical CPS



Introduction

Challenge and Motivation

- The key to monitor and control Smart Grid and other Mission-critical Cyber-Physical Systems (CPS) is to provide *situational awareness* of the system.
  - "the goals of situational awareness are to understand and ultimately optimize the management of power-network components, behavior, and performance, as well as to anticipate, prevent, or respond to problems before disruptions arise" [1]

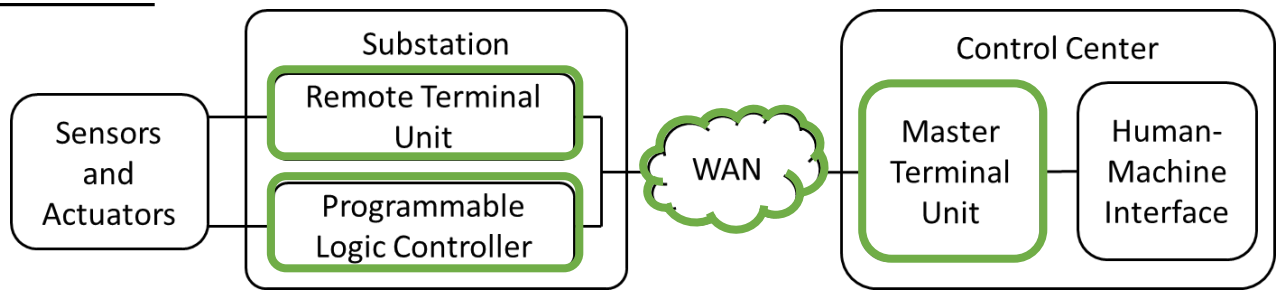
- Providing situational awareness of Smart Grid is challenging.
  - There are different challenges in providing situational awareness in SCADA networks.

[1] C. Greer, D. A. Wollman, D. E. Prochaska, P. A. Boynton, J. A. Mazer, C. T. Nguyen, G. J. FitzPatrick, T. L. Nelson, G. H. Koepke, A. R. Hefner Jr et al., "Nist framework and roadmap for smart grid interoperability standards, release 3.0," Tech. Rep., 2014.

Introduction

Challenge and Motivation

- SCADA architecture



- The main challenge to guarantee situational awareness in SCADA is the *lack of security protection*.
  - Connecting a growing number of heterogeneous programmable devices together introduces new security risks.
  - Many devices and protocols in SCADA are not designed with security in mind and lack the vital security protection capabilities.

Introduction

Challenges and Motivation

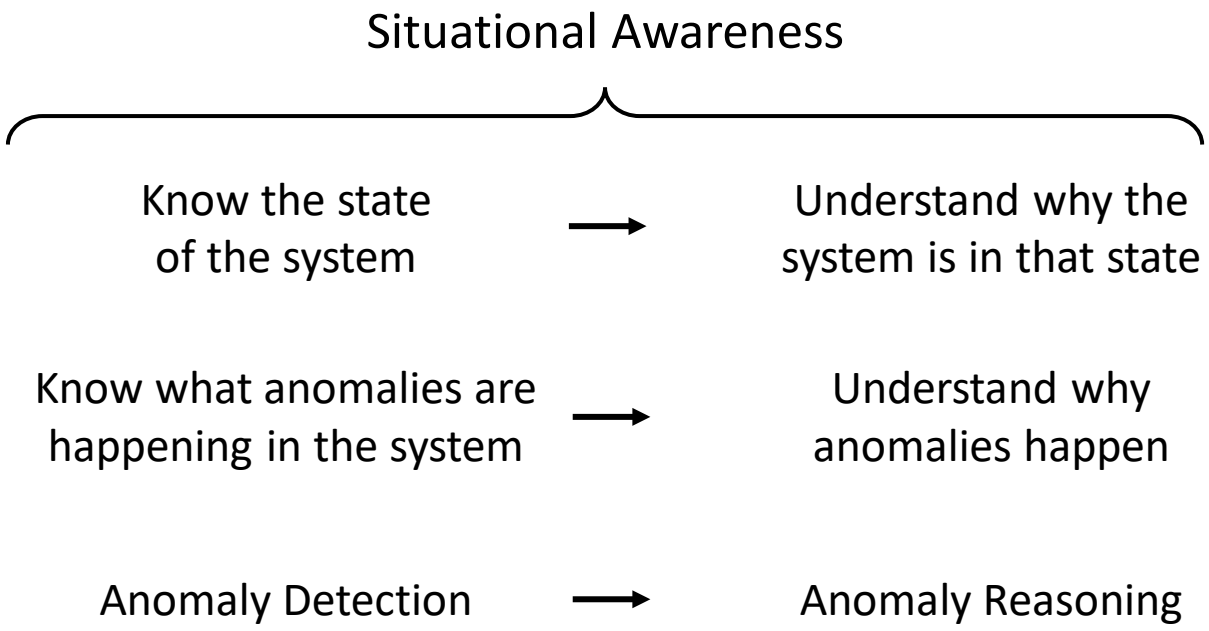
- SCADA systems are subject to a wide range of serious threats due to the following vulnerabilities:
  1. The adoption of cutting-edge communication technologies contributes to the increasing complexity and interconnection of SCADA systems.
  2. Devices in SCADA systems are usually not built with cybersecurity in consideration and lack authentication or encryption mechanisms.
  3. Most ICS protocols lack authentication features and provide no protection for the network traffic.

- Developing techniques to target those vulnerabilities and provide security to SCADA systems is important.

Introduction

Causal Security Analysis

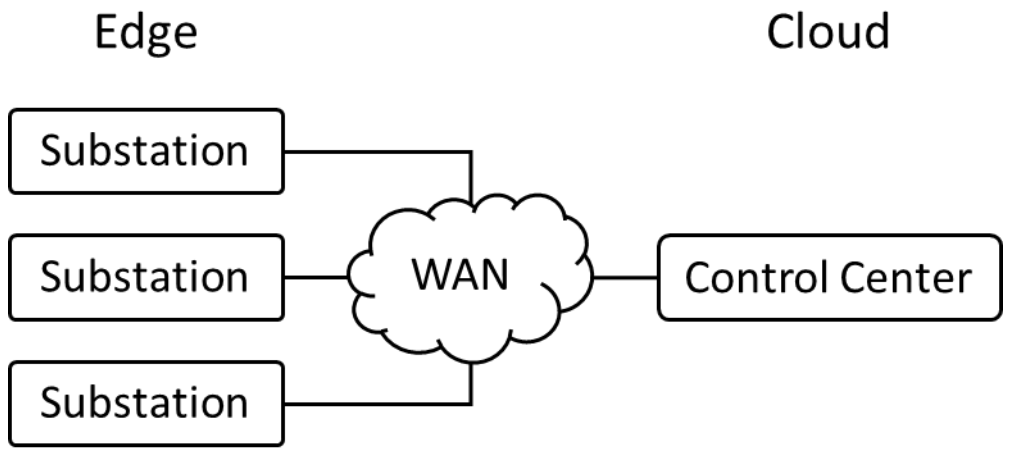
- It is necessary to provide causal security analysis of the system which includes anomaly detection AND reasoning of anomalies.



Introduction

Edge-cloud Design

- An edge-cloud design is essential in providing situational awareness to Smart Grid.
  - Faster actuation and response of the Smart Grid system to events
  - Better utilization of the communication bandwidth
  - Increase of reliability and scalability



EDMAND – Edge-Based Multi-Level Anomaly Detection for SCADA

Problem Description

- Objective

  Design a framework to provide real-time security in SCADA.

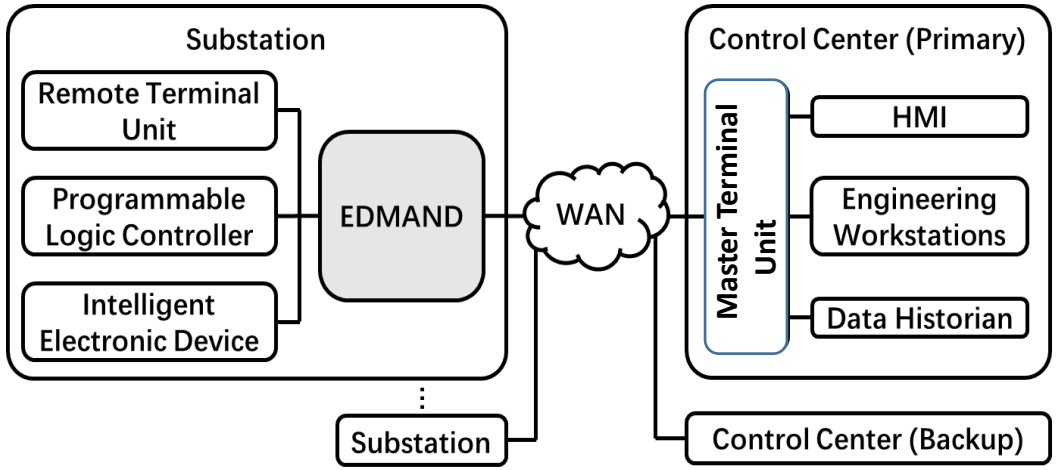
- We focus on network-based analysis due to its less intrusive nature.
- We divide data in SCADA network traffic into three levels
  - Transport level: Transport level data refers to statistics in IP headers and transport protocol headers.
  - Operation level: Operation level data refers to operation statistics in ICS protocols.
  - Content level: Content level data refers to measurement statistics from field devices.

EDMAND Approach

- EDge-based Multi-level ANomaly Detection framework for SCADA networks named EDMAND.
  - EDMAND is located inside the remote substations, which are the edges of the SCADA network.
  - EDMAND contains a multi-level anomaly detector to monitor all three levels of network traffic data passing by.
  - Appropriate anomaly detection methods are applied based on the distinct characteristics of data in various levels.
  - Generated alerts are aggregated, prioritized, and sent back to control centers when anomalies are detected.

Placement of EDMAND

- Network Architecture
  - Major components in SCADA network: MTUs, remote devices, and the communication network.



  - EDMAND is deployed in each substation between the remote devices and the wide area network.

EDMAND Design

- Design decision
  - Divide traffic data into multiple levels and apply appropriate anomaly detection mechanisms to data in each level based on their characteristics.
  - Introduce the concept of confidence into the anomaly detection process and assign confidence scores to generated alerts.
- Confidence Score

Confidence that the corresponding alert is an anomaly.

$$\boldsymbol{Confidence\ Score(CS) = Model\ Accuracy(MA)} \times \boldsymbol{Anomaly\ Score(AS)}$$

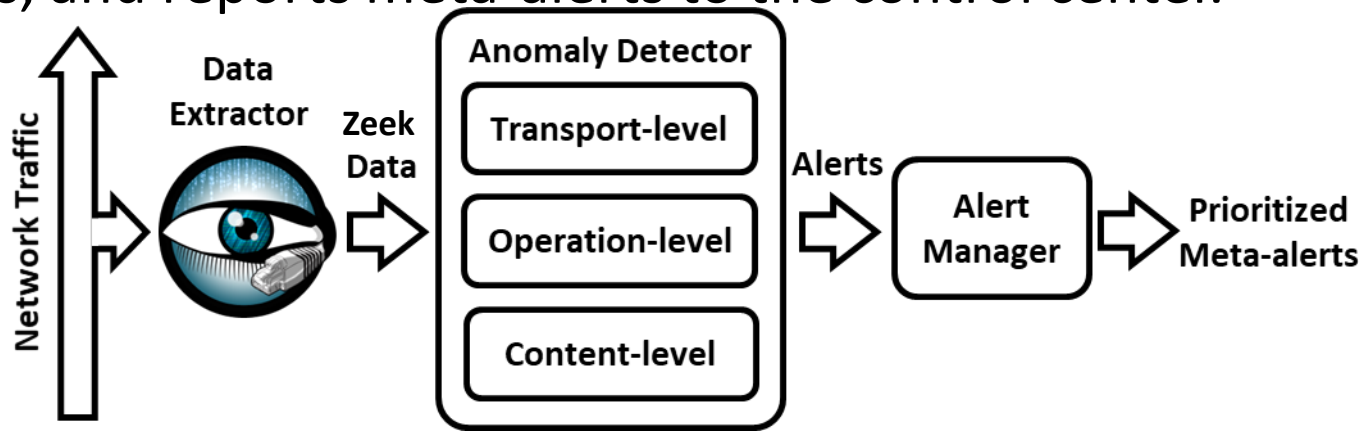
$\in [0, 1]$

How accurate is our model in describing normal behavior

How far does the current value deviate from the normal value

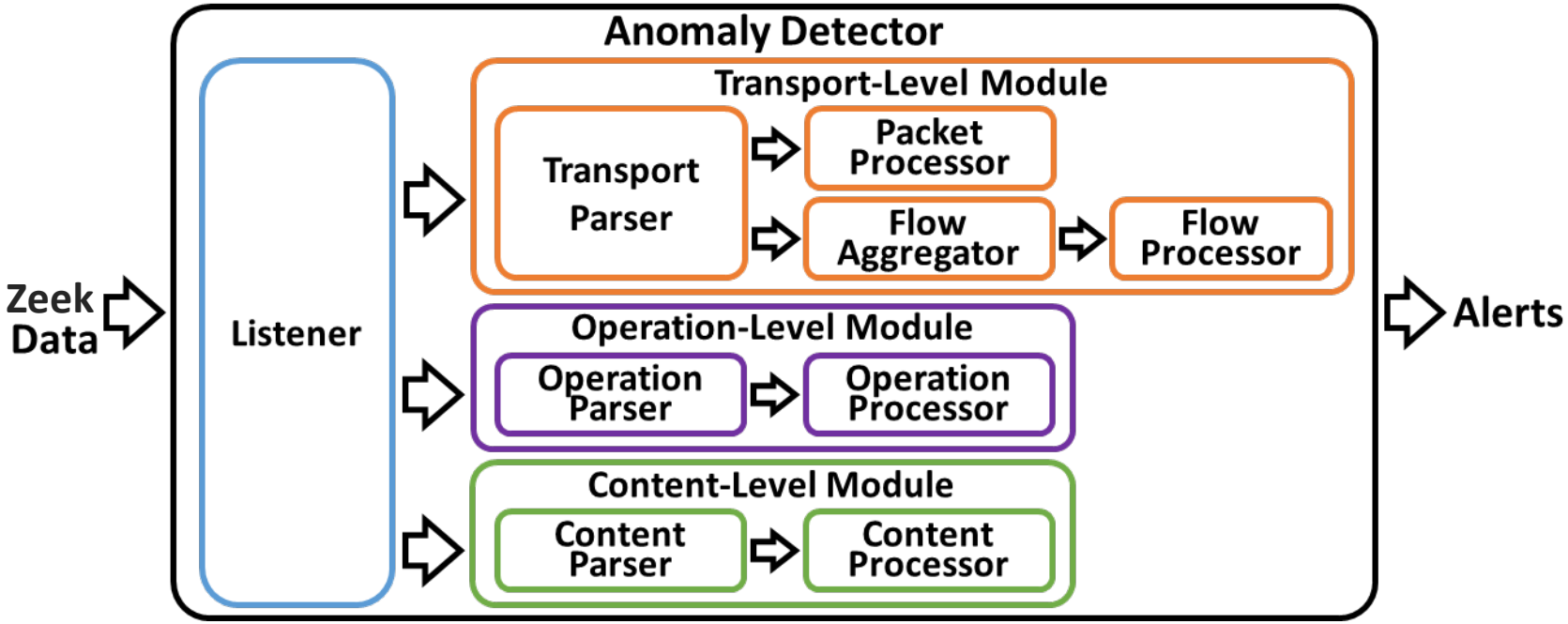
Framework Design

- Main components of EDMAND
  - Data Extractor : monitors the network traffic passing by and forwards all three levels of network traffic data to the anomaly detector.
  - Anomaly Detector: contains three levels and each level uses appropriate method to detect anomalies and generates alerts.
  - Alert Manager : aggregates similar alerts into meta-alerts, calculates priorities of meta-alerts, and reports meta-alerts to the control center.



Anomaly Detector

- Multi-level anomaly detector



Anomaly Detector: Transport Level

- Two kinds of analysis at different time scales
  - Packet processor: runs every packet for short-term analysis.
  - Flow processor: runs every period T_{flow} for long-term analysis.

Packet count (*PC*)
Average Packet Size (*APS*) } Unimodal distribution ⟶ μ, σ

Interarrival time (*IAT*)
Packet size (*PS*) } Multimodal distribution ⟶ Clustering

INPUT FIELDS AND ANOMALY DETECTION MECHANISM OF PACKET PROCESSOR AND FLOW PROCESSOR

	Packet Processor	Flow Processor
Index Field	(originator, responder, transport protocol, port number)	
Data Field	interarrival time (IAT) packet size (PS)	packet count (PC) average packet size (APS)
Mechanism	1D-DenStream	Mean-STD

Anomaly Detector: Operation Level

- Operation processor
  - Objective: detect anomalies in operations of industrial control protocols (Modbus, DNP3)
  - Index fields: originator, responder, ICS protocol, unit id, function code
  - Data field: interarrival time (IAT)

ANOMALY AND DETECTION MECHANISM IN OPERATION LEVEL

Anomaly	Mechanism
invalid function code	CS=1
wrong direction of operation	
new operation	AS=1
early operation	Mean-STD
late operation	
missing operation	

CS – Confidence Score
AS – Anomaly Score

Anomaly Detector: Content Level

- Content processor
  - Objective: detect anomalies in measurement values which are included in responses to read requests
  - Index fields: measurement source, ICS protocol, unit id, measurement type, measurement index
  - Data field: measurement value
  - Method: different methods for different measurement types

- DNP3 measurement type
  - Binary
  - Analog
  - Counter

} most common

Anomaly Detector: Content Level

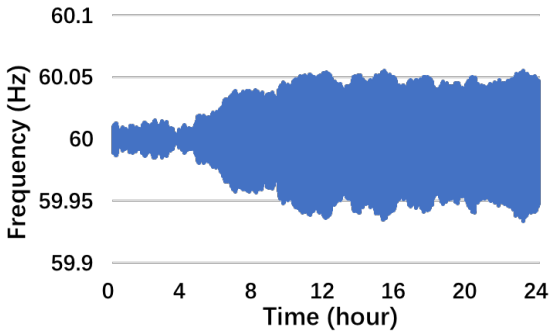
- Binary
  - Intuition: binary measurement usually has a normal value and an abnormal value
  - Method: count 0s and 1s and try to identify the normal value
  - Anomaly Score (AS): 1 – Entropy(observed samples)

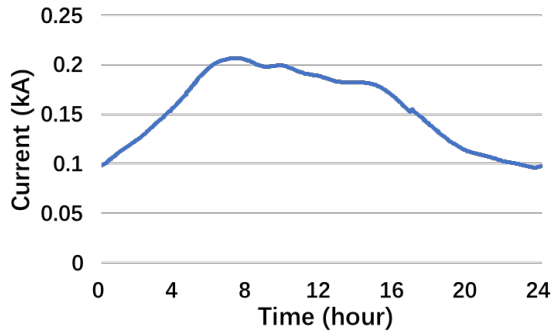
$$AS(\gamma) = \begin{cases} 1 & \gamma = 0 \text{ or } 1 \\ 1 + \gamma \log_2 \gamma + (1-\gamma)\log_2(1-\gamma) & 0 < \gamma < 1 \end{cases}$$

where $\gamma = \dfrac{number\ of\ 0s\ observed}{number\ of\ samples\ observed}$

Anomaly Detector: Content Level

- Analog
  - Most common analog measurements include frequency, voltage, current, power
  - They have quite different characteristics

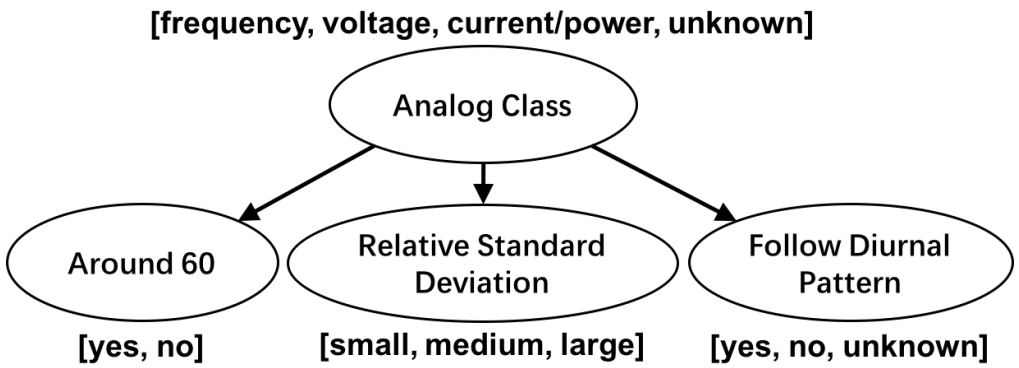




- 2-step anomaly detection
  1. Categorizes analog measurements into different analog classes
  2. Uses appropriate method for each class

Anomaly Detector: Content Level

- Step 1: Bayesian analog type inference model



- We denote y^k as the observation at k^{th} leaf node and x_i as the i^{th} analog type at the root node
- Let $P(x_i)$ be the prior probability for the hypotheses of the root

$$P(x_i|y^1, y^2, y^3) = \alpha P(x_i) \prod_{k=1}^{3} P(y^k|x_i)$$

where $\alpha = \frac{1}{P(y^1, y^2, y^3)}$ and can be calculated using $\sum_i P(x_i|y^1, y^2, y^3) = 1$

Anomaly Detector: Content Level

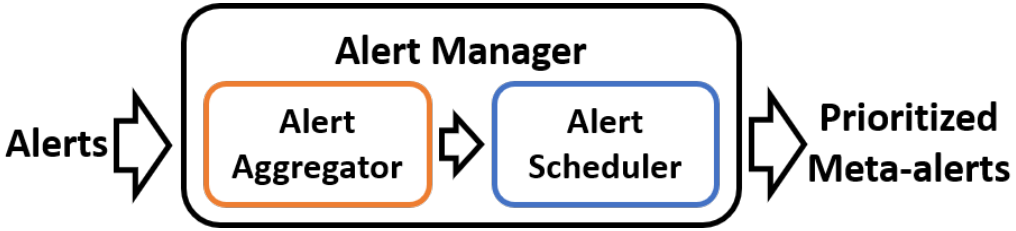
- Step 2: Apply different anomaly detection method for each analog class

ANALOG MEASUREMENT CLASS AND DETECTION MECHANISM

Analog Class	Mechanism
frequency	Mean-STD
voltage	
current/power	slotted Mean-STD
unknown	1D-DenStream

Alert Manager

- Alert field
  - Index fields (same as index fields of the corresponding processor)
  - Alert type
  - Timestamp
  - Confidence score
  - Statistical fields (current value, mean, standard deviation, etc.)
  - Abnormal data (original parsed data of the corresponding level)
- Alert manager structure



Alert Aggregator

- Objective

  Aggregate alerts that have same type as well as index fields and have little difference in timestamp

META-ALERT FIELDS AND AGGREGATION RULES

Alert Field	Aggregation Rule
index field	shared by all of the aggregated alerts
alert type	
timestamp	keep minimum and maximum
confidence score	keep maximum
statistical fields	inherit from the last alert aggregated
anomaly data	
count	number of aggregated alerts

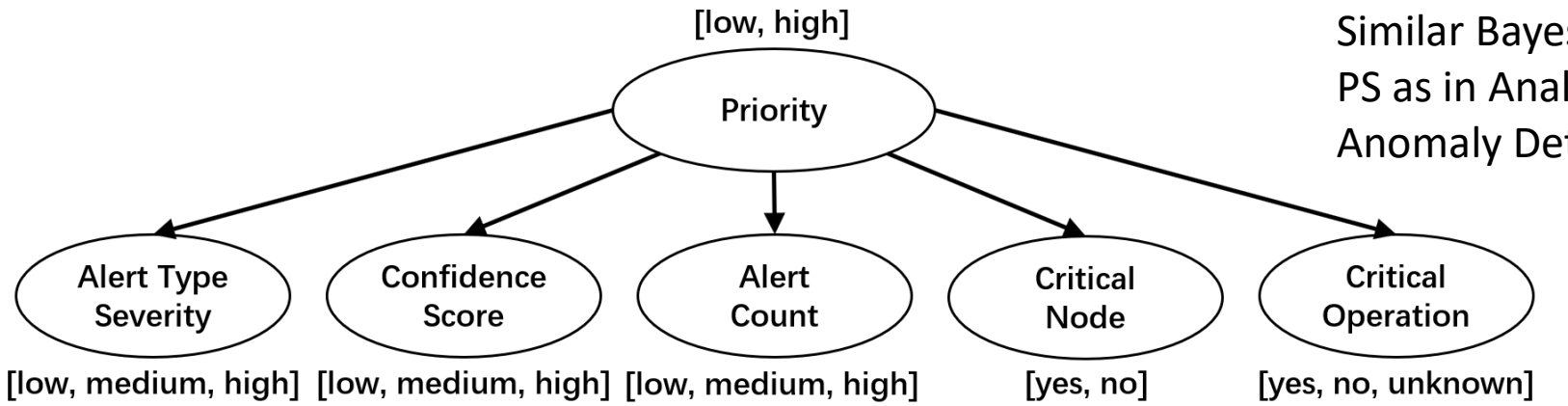
Alert Scheduler

- Objective

  Calculate priority score for each meta-alert and decide when to report it to the control center

- Priority score

  - We denote y^k as the observation at k^{th} leaf node
  - Define $PS = P(Priority = high | y^1, y^2, y^3, y^4, y^5)$



Similar Bayesian calculation of PS as in Analog Content Level Anomaly Detection

Alert Scheduler

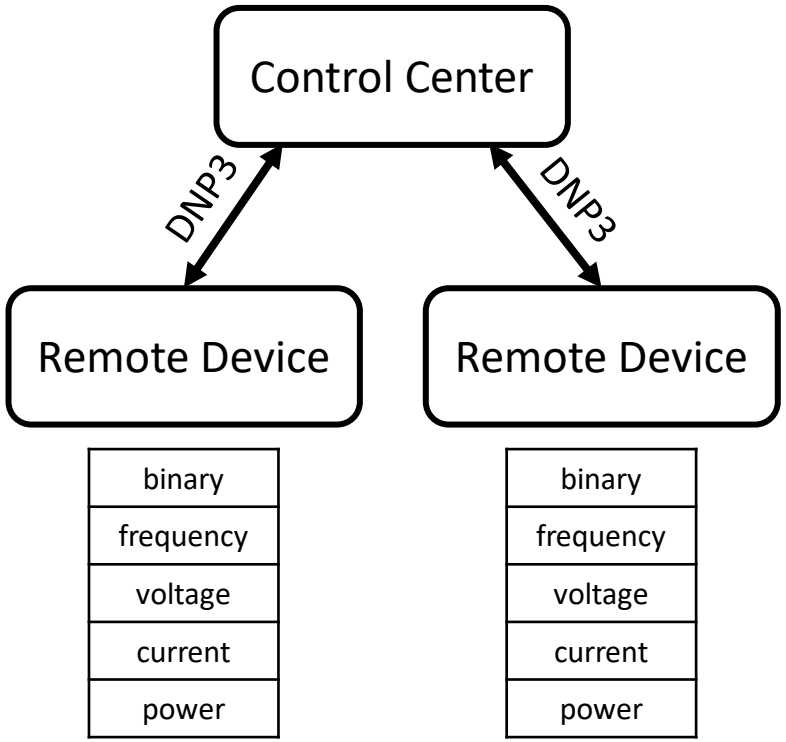
- Meta-alert report frequency

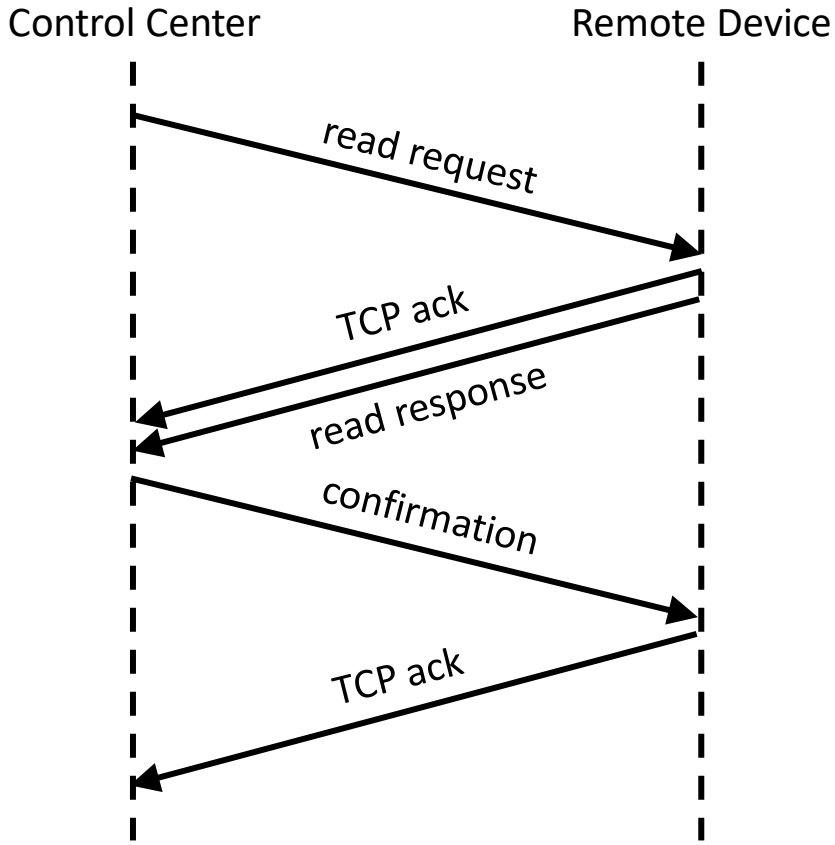
META-ALERT REPORT MECHANISM

	High-Priority	Low-Priority
Definition	$PS \geq \theta_p$	$PS < \theta_p$
Report when first created	yes	no
Report period	T_h	$T_l (> T_h)$

Evaluation

- Baseline traffic





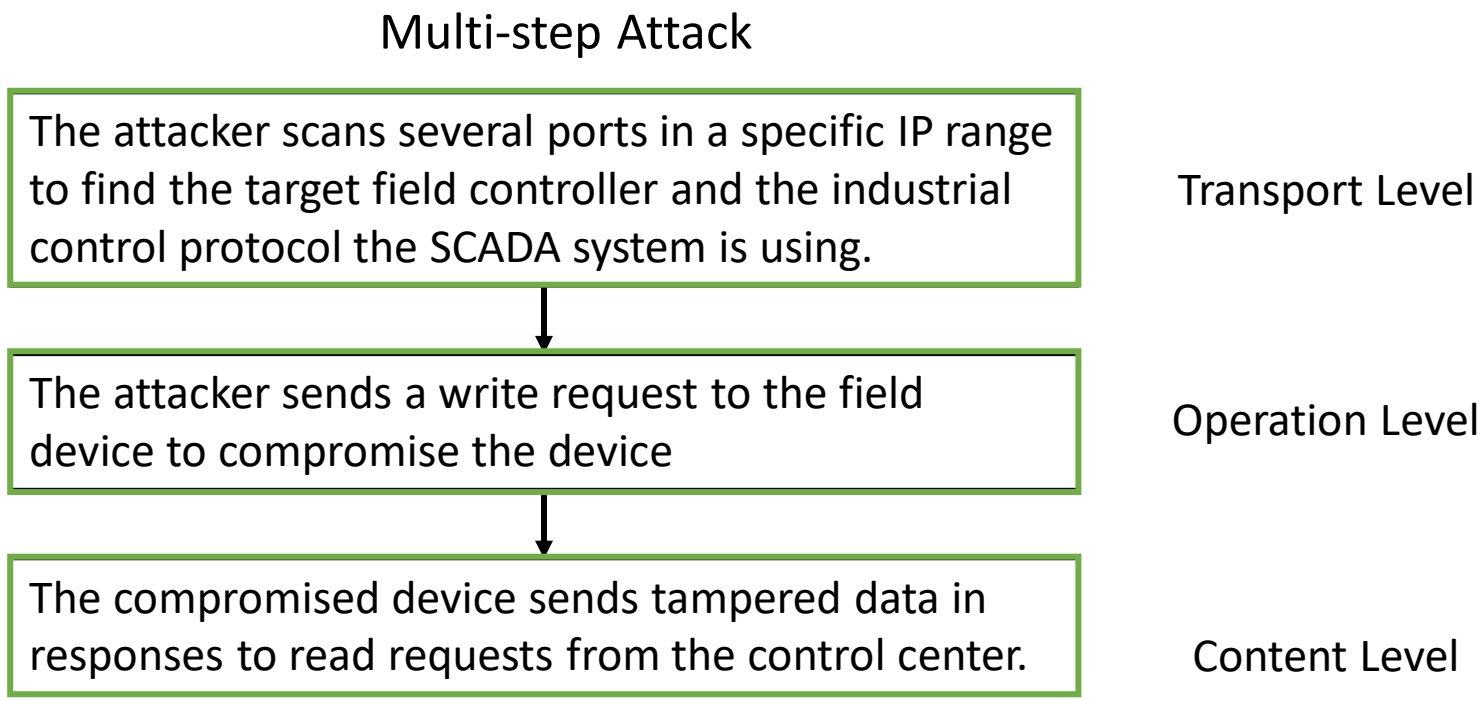
Evaluation

- Detection ability
  - The analog class inference model correctly identifies all analog classes.
  - EDMAND can detect all the anomalies injected.
  - 12135 alerts are aggregated to 22 meta-alerts.

Level	Anomaly
Transport	add a new node to send several packets to one field controller
	pad one response from a field controller with more payload
	delay one TCP acknowledgement from a field controller intentionally
	send lots of ICMP packets in a short period to one field controller
Operation	send one operation with invalid function code to one field controller
	let one field controller send a control command to the control center
	delay one response from a field controller intentionally
Content	tamper the binary value from one field controller for a short period
	introduce over voltage and under voltage tripping to voltage measurements
	introduce over current tripping to current measurements
	tamper the frequency value from one field controller for a short period
	tamper the power value from one field controller for a short period

Evaluation

- Detection ability



Evaluation

- Time overhead
  - Ubuntu 16.04 desktop with 12 Intel Xeon 3.60GHz CPUs and 16GB memory
  - Total analysis time (data extraction + anomaly detection) per packet:
    - Transport level: 3.87ms
    - Operation level: 6.66ms
    - Content level: 1.94ms
  - The average time overhead of the anomaly manager for each alert is 423ms.

CAPTAR – Causal Reasoning about Anomalies in SCADA

Problem Description

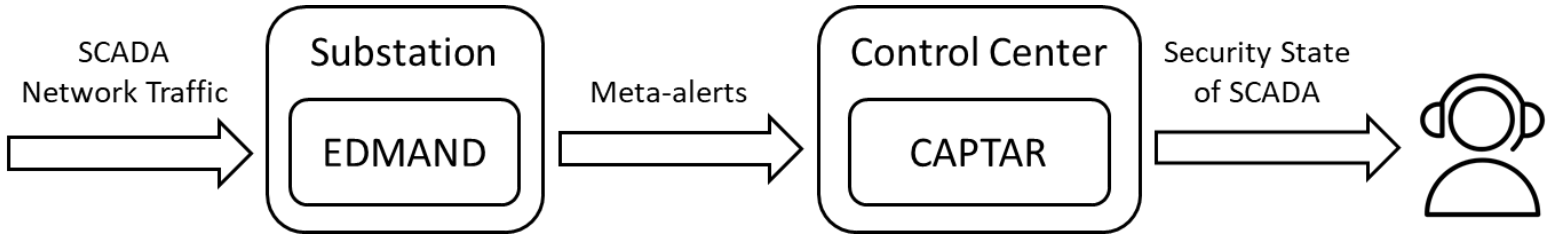
- Problem
  - Traditional intrusion detection systems for SCADA networks continuously generate tremendous number of alerts without further comprehending them.
  - SCADA operators are almost blind to see any useful information in the ocean of unstructured alerts mixed with false positives.

- Objective

  Design an efficient system for SCADA to correlate alerts from intrusion detection systems in an intelligent manner and discover attack strategies based on domain knowledge as well as causal reasoning.

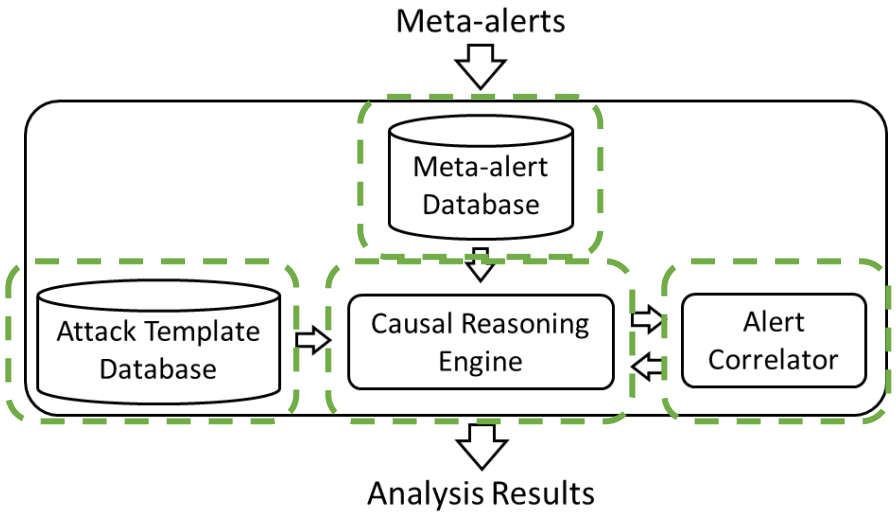
CAPTAR Approach

- CAPTAR: CAusal-PolyTree-based Anomaly Reasoning framework for SCADA networks.
  - CAPTAR resides in the control center of the SCADA network and takes the meta-alerts from EDMAND as input.
  - CAPTAR correlates the alerts using a naive Bayes classifier and matches them to predefined causal polytrees which represent attacks.
  - Utilizing Bayesian inference on the causal polytrees, CAPTAR is able to reveal the attack scenarios from the alerts and produces a high-level view of the security state of the protected SCADA network.

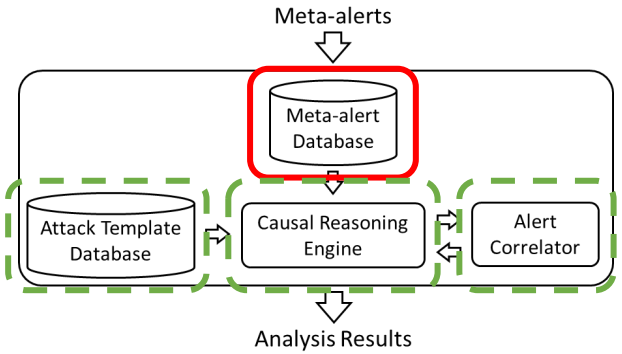


Framework Design

- Components of CAPTAR
  - Meta-alert Database: stores the meta-alerts from EDMAND.
  - Attack Template Database: stores potential attack templates which are causal polytrees created by domain experts.
  - Alert Correlator: decides if two meta-alerts are correlated or not
  - Causal Reasoning Engine: matches meta-alerts to attack templates and performs belief propagation.



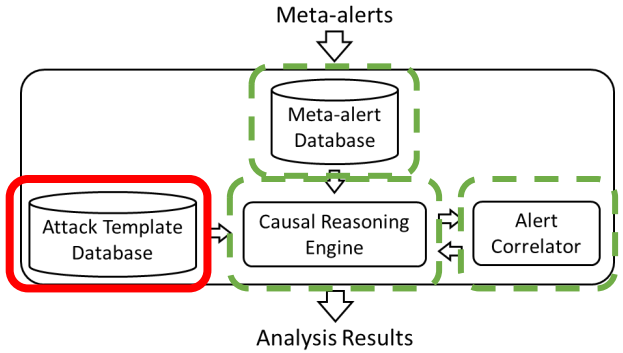
Meta-alert Database



- Meta-alert fields
  - Alert ID: a unique id for retrieving the meta-alert from the database
  - Alert Type: a name that describes the meta-alert
  - Index Field : a set of additional information that helps to describe the meta-alert
  - Timestamp : (start time, end time)
  - Confidence Score : the confidence that the meta-alert is triggered by an anomaly

Index	Alert Type
0	PACKET_IAT
1	PACKET_BYTES
2	NEW_ORIG
3	NEW_RESP
4	NEW_PROTOCOL
5	NEW_SERVICE
6	PACKET_AB_TOO_MANY
7	PACKET_AB_TOO_FEW
8	PACKET_BA_TOO_MANY
9	PACKET_BA_TOO_FEW
10	MEAN_BYTES_AB_TOO_LARGE
11	MEAN_BYTES_AB_TOO_SMALL
12	MEAN_BYTES_BA_TOO_LARGE
13	MEAN_BYTES_BA_TOO_SMALL
14	OPERATION_TOO_LATE
15	OPERATION_TOO_EARLY
16	OPERATION_MISSING
17	INVALID_FUNCTION_CODE
18	RESPONSE_FROM_ORIG
19	REQUEST_FROM_RESP
20	NEW_OPERATION
21	BINARY_FAULT
22	ANALOG_TOO_LARGE
23	ANALOG_TOO_SMALL

Attack Template Database

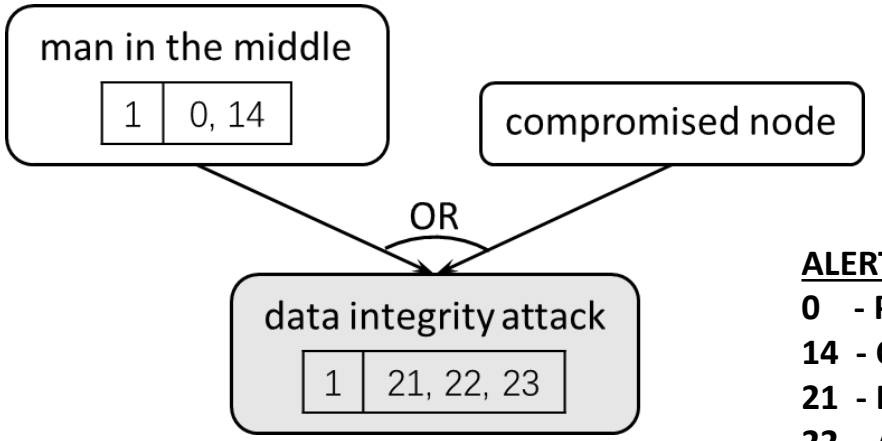


- Nodes in attack template (AT)
  - Each node in an attack template is an attack step.
  - Each parent represents a prior cause attack step.
  - Each child represents a posterior consequence attack step.
  - Sink nodes $\boldsymbol{S}_{AT}$ represent the final targets of the entire attack and we call them consequence nodes.

    – The maximum probability of existence of all consequence nodes in AT, $BEL_{max}\ (AT)$, represents the inferred success rate of the attack.

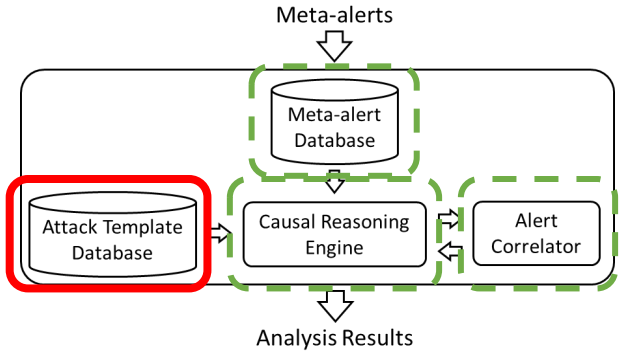
$$BEL_{max}\ (AT) = \max_{X \in \boldsymbol{S}_{AT}} BEL_x(1)$$

Attack Template AT (Causal Polytree Example)



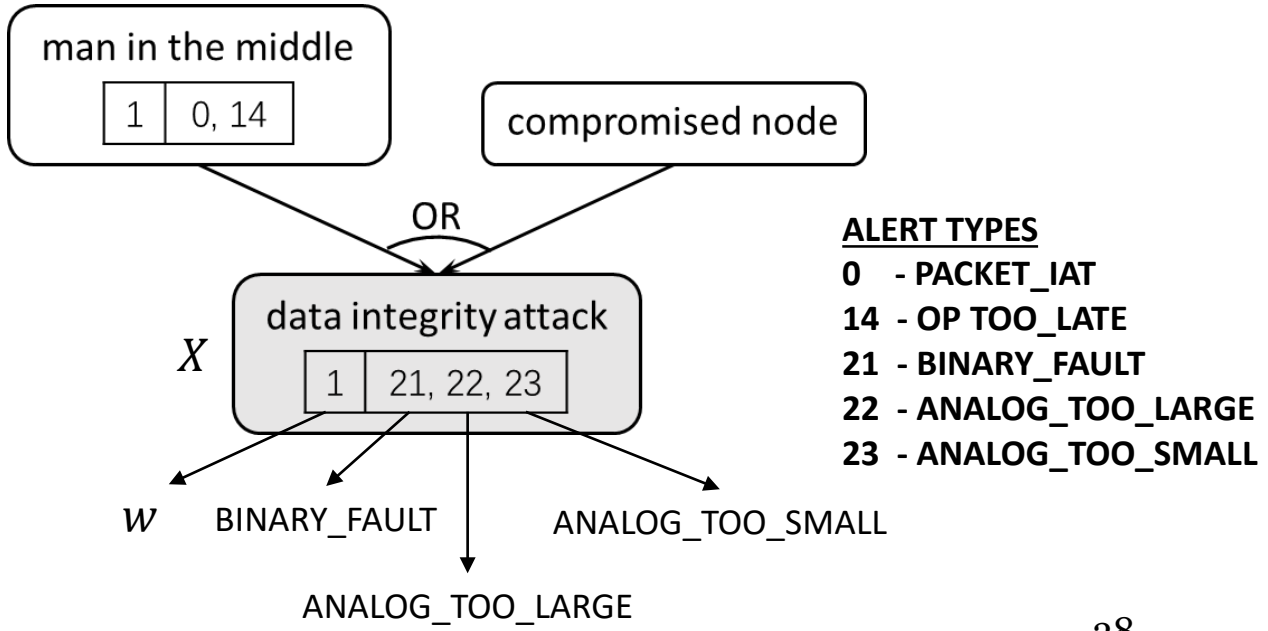
ALERT TYPES
0 - PACKET_IAT
14 - OP TOO_LATE
21 - BINARY_FAULT
22 - ANALOG_TOO_LARGE
23 - ANALOG_TOO_SMALL

Attack Template

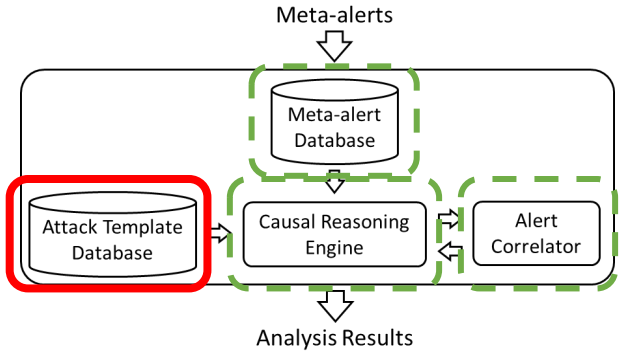


- Alert unit table
  - Alert Unit $AU_i = (w_i, A_{i1}, A_{i1}, \ldots, A_{in_i})$
  - $\sum_i w_i = 1$
  - Alert types in the same alert unit represent similar kinds of anomalies caused by the attack step

Alert Unit	Weight	Alert Types
AU_1	w_1	$A_{11}, A_{12}, \ldots, A_{1n_1}$
AU_2	w_2	$A_{21}, A_{22}, \ldots, A_{2n_2}$
$\vdots$	$\vdots$	$\vdots$
AU_k	w_k	$A_{k1}, A_{k2}, \ldots, A_{kn_k}$



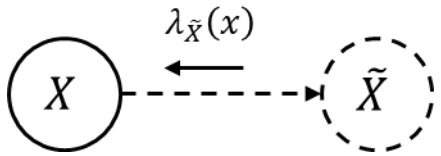
Attack Template



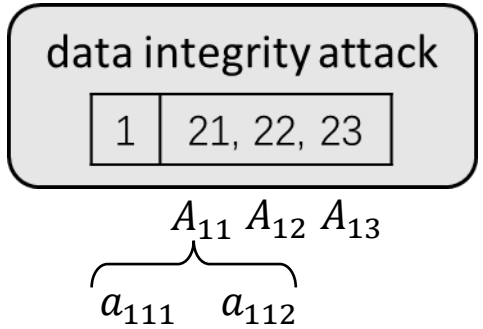
- The confidence scores of the matched meta-alerts are used to calculate the diagnostic support message $\lambda_{\tilde{X}}(x)$ (indirect evidence).
  - Assume there are m_{ij} meta-alerts $a_{ij1}, a_{ij2}, \ldots, a_{ijm_{ij}}$ *matched* to alert type A_{ij}.
  - **Confidence score of the alert type A_{ij} :**

$$CS(A_{ij}) = \begin{cases} \dfrac{\prod_{l=1}^{m_{ij}} CS(a_{ijl})}{\prod_{l=1}^{m_{ij}} CS(a_{ijl}) + \prod_{l=1}^{m_{ij}} (1 - CS(a_{ijl}))} & if\ m_{ij} > 0 \\ P_{miss} & if\ m_{ij} = 0 \end{cases}$$

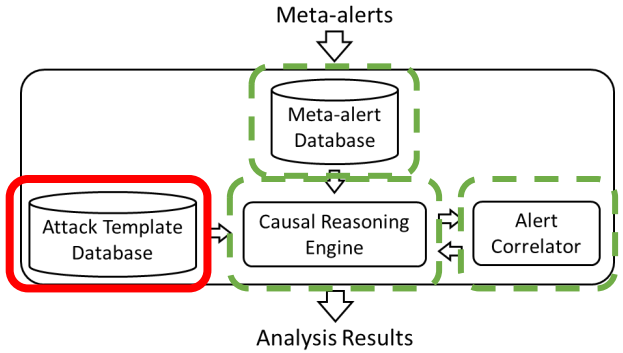
Note: Matching Done by Causal Reasoning Engine



Alert Unit	Weight	Alert Types
AU_1	w_1	$A_{11}, A_{12}, \ldots, A_{1n_1}$
AU_2	w_2	$A_{21}, A_{22}, \ldots, A_{2n_2}$
⋮	⋮	A_{ij} ⋮ $a_{ij1}, a_{ij2}, \ldots, a_{ijm_{ij}}$
AU_k	w_k	$A_{k1}, A_{k2}, \ldots, A_{kn_k}$

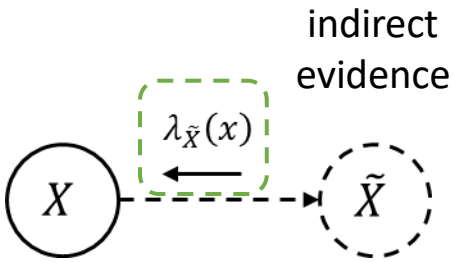


Attack Template



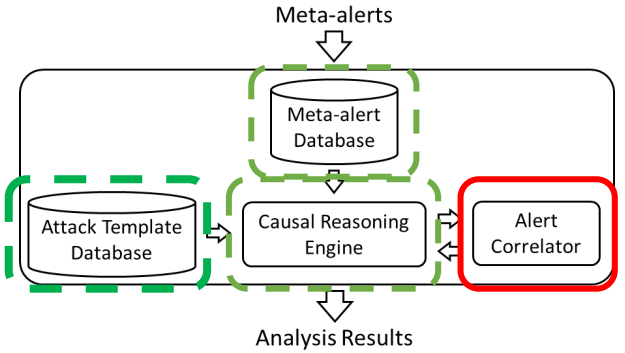
- The confidence scores of the matched meta-alerts are used to calculate the diagnostic support message $\lambda_{\tilde{X}}(x)$.
  - **Confidence score of the alert unit AU_i** : $CS(AU_i) = \max_{j=1}^{n_i} CS(A_{ij})$
  - **Total confidence score of the attack step**: $CS_{total} = \sum_{i=1}^{k} w_i CS(AU_i)$
  - **Diagnostic support** provided by all the matched alerts to node X:

$$\lambda_{\tilde{X}}(x) = \begin{cases} 1 - CS_{total} & if\ x = 0 \\ CS_{total} & if\ x = 1 \end{cases}$$

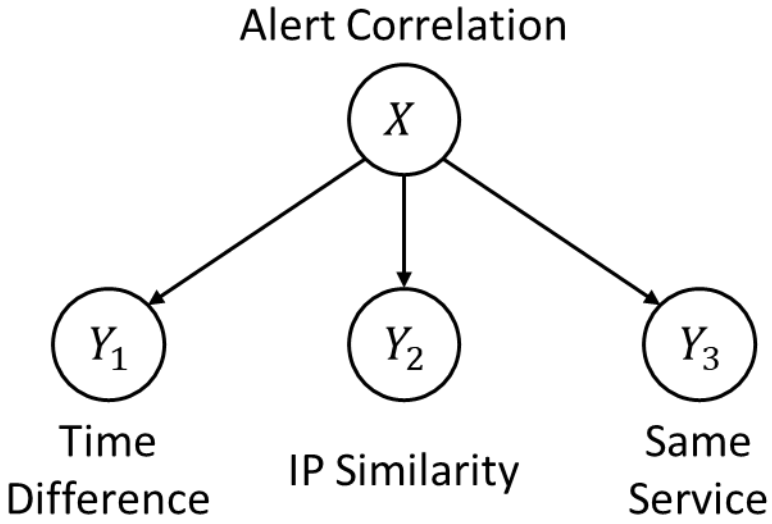


	Alert Unit	Weight	Alert Types
	AU_1	w_1	$A_{11}, A_{12}, \ldots, A_{1n_1}$
	AU_2	w_2	$A_{21}, A_{22}, \ldots, A_{2n_2}$
AU_i	$\vdots$	$\vdots$	$\vdots$
	AU_k	w_k	$A_{k1}, A_{k2}, \ldots, A_{kn_k}$

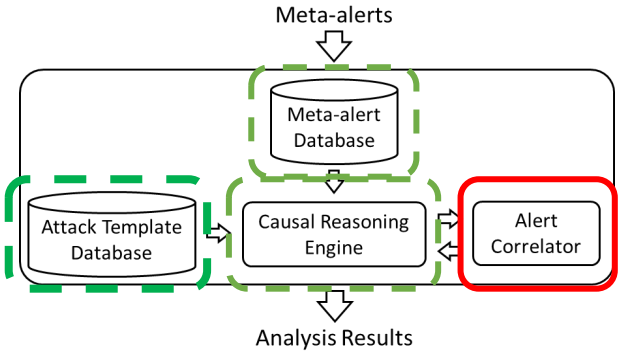
Alert Correlator



- Objective: To decide if two meta-alerts are correlated or not.
- Graphical representation: a Bayesian network with one root node X and three leaf nodes Y_1, Y_2, and Y_3.



Alert Correlator

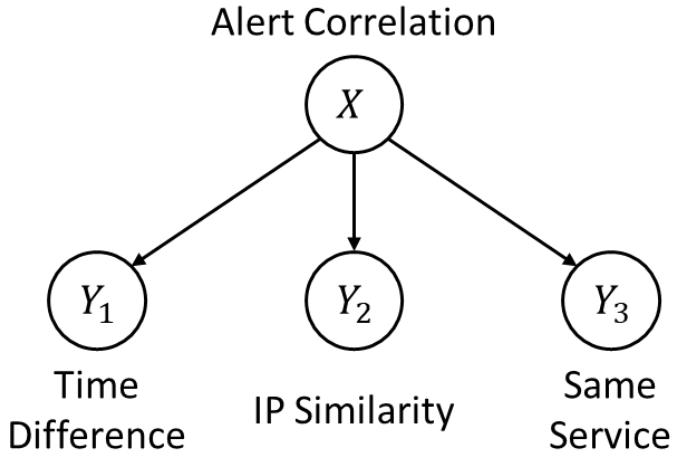


- Evidence nodes
  - Time Difference: the closeness in the time axis of the two meta-alerts.
  - IP Similarity: the similarity of IP addresses related to the two meta-alerts.
  - Same Service: whether the two meta-alerts share the same service (i.e., the same industrial control protocol).
  - Belief at root X:

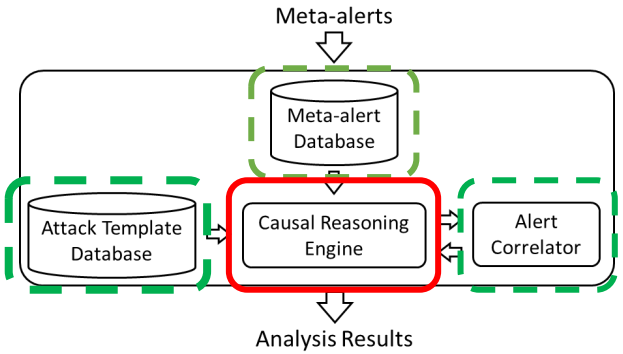
$$BEL_X(x) = \alpha P(x) \prod_{j=1}^{3} P(y_j|x)$$

  - The CORRELATE procedure for two input meta-alerts a and b:

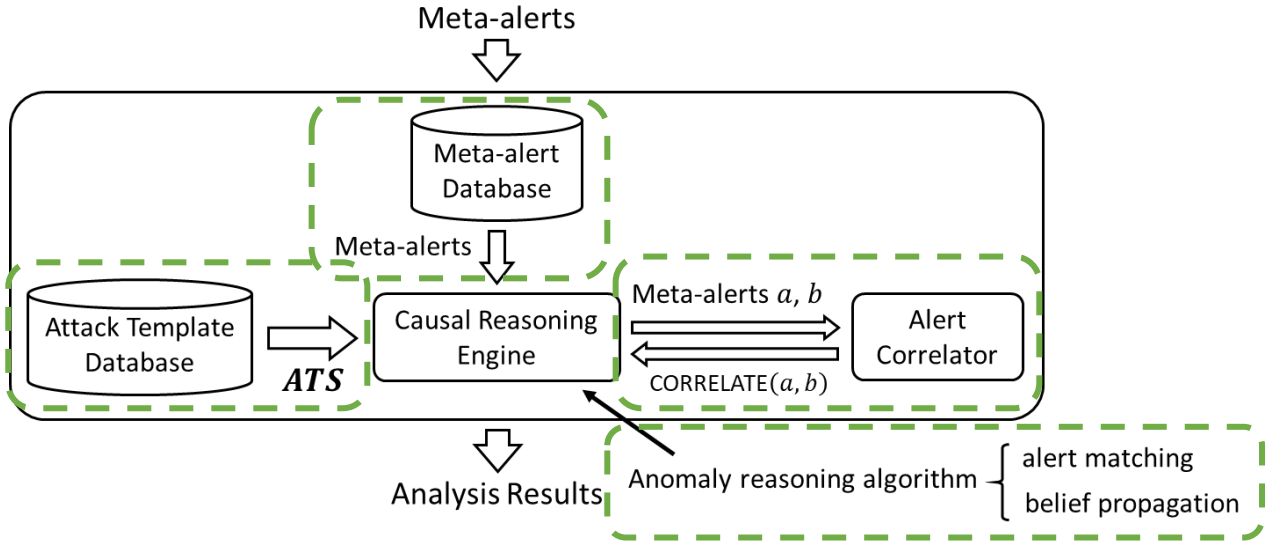
$$\text{CORRELATE}(a,b) = \begin{cases} BEL_X(1) & if\, BEL_X(1) > 0.5 \\ -1 & otherwise \end{cases}$$



Causal Reasoning Engine



- The causal reasoning engine is the core component.
  - It fetches copies of attack templates ATs from the attack template database and creates an attack template set $\boldsymbol{ATS}$.
  - It runs an anomaly reasoning algorithm to perform alert matching and belief propagation on the attack templates in the attack template set.
  - It retrieves meta-alerts from the meta-alert database.
  - It uses the alert correlator to correlate meta-alerts during the matching process.

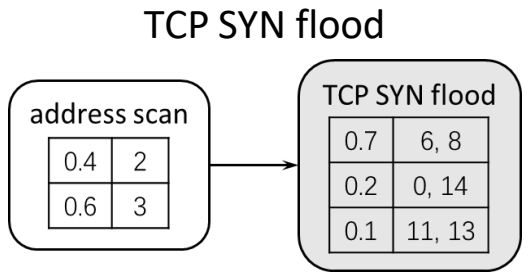


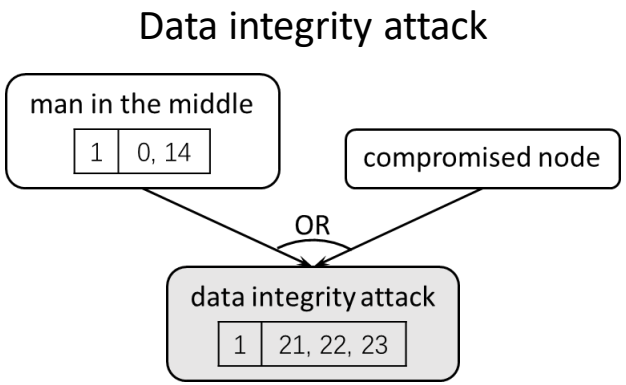
Evaluation

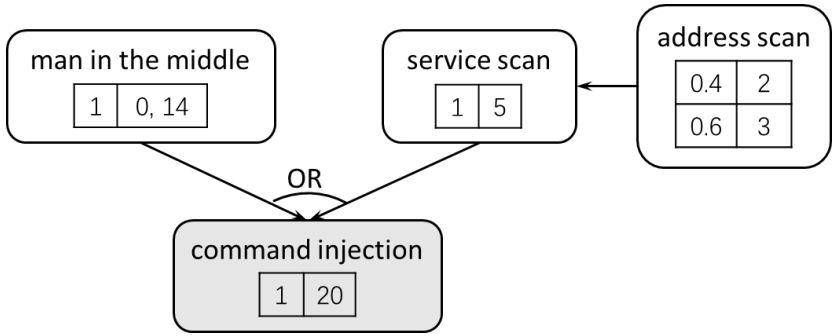
- Baseline traffic

  14 days of simulated DNP3 traffic of one control center communicating with 10 remote terminal units (RTUs).

- Attacks
  - TCP SYN flood
  - Data integrity attack
  - Command injection







Evaluation

- Detection ability
  - CAPTAR together with EDMAND are able to identify and differentiate all three attacks.
- The anomaly reasoning algorithm has an estimated time complexity of $O(KLMN)$ in the worst case.
  - M: number of meta-alerts in the database
  - K: maximum limit for the number of attack templates to keep for each kind of attack
  - N: maximum number of nodes in any attack template
  - L: number of attack templates in the database
- Average time overhead

Attack	FindCorrelation	Belief Propagation	Anomaly Reasoning
TCP SYN flood	7.39ms	0.18ms	37.66ms
Data integrity attack	0.62ms	0.16ms	18.08ms
Command injection	2.77ms	0.16ms	12.50ms

Summary

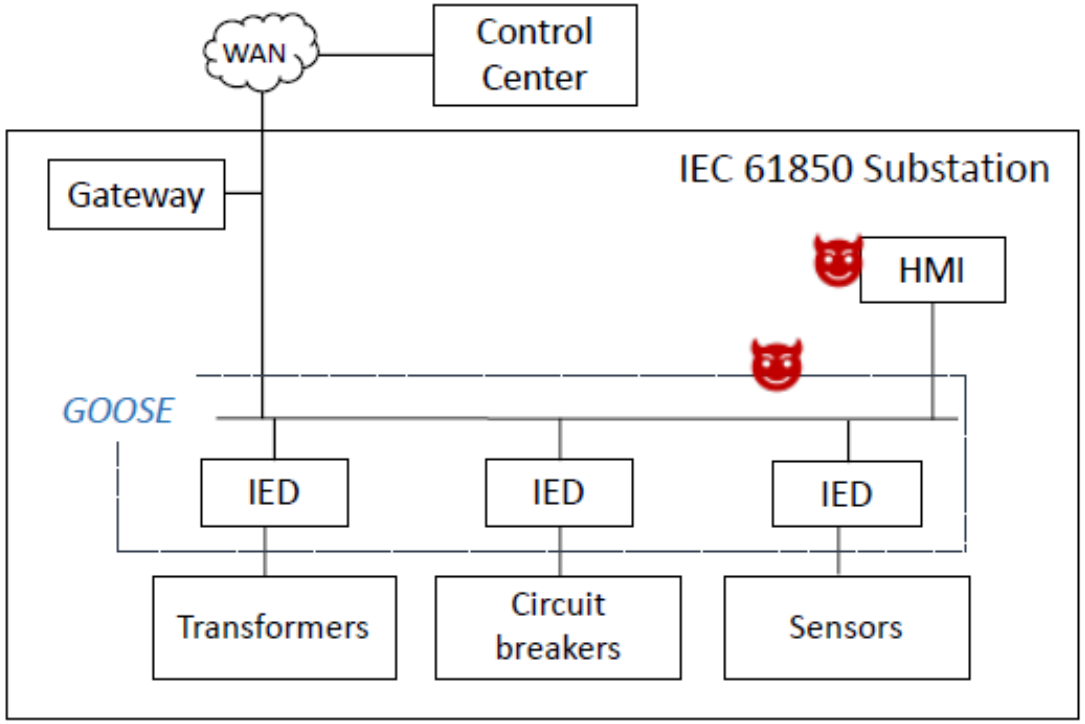
- EDMAND: an edge-based multilevel anomaly detection framework for SCADA systems
  - EDMAND divides traffic data into multiple levels and applies appropriate anomaly detection mechanism to data in each level based on their characteristics.
  - EDMAND introduces the concept of confidence into the anomaly detection process and assign confidence scores to generated alerts.

- CAPTAR: Causal-Polytree-based Anomaly Reasoning for SCADA Networks
  - CAPTAR goes on step further than anomaly detection and uses alert correlation and causal reasoning to understand the causes of the anomalies.
  - CAPTAR provides situational awareness that is explainable.

ED4GAP – Efficient Detection for GOOSE-based Poisoning Attacks

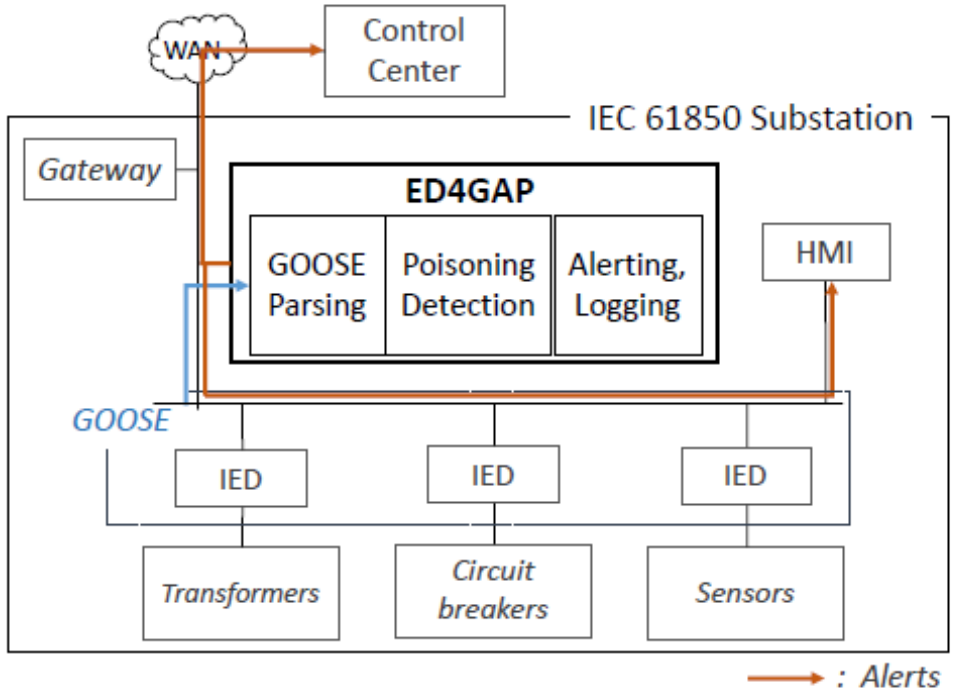
Problem and Motivation

- GOOSE Poisoning
  - Form of false data injection attack
  - Cause of outages and equipmen damage
- Why is this relevant to GOOSE?
  - Lack of encryption and authentication in GOOSE protocol
- Key Challenges
  - Strict timing requirements (4ms)
  - Constrained end devices

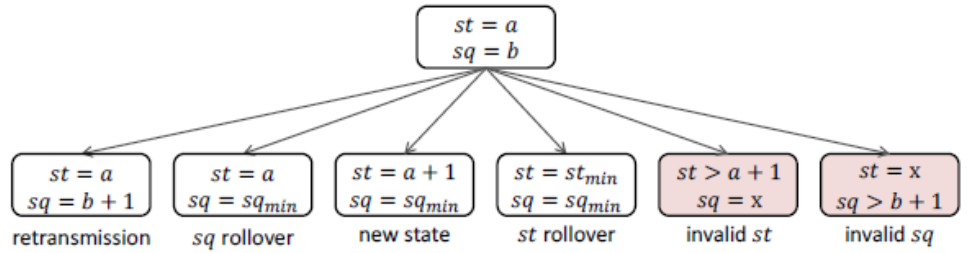


Threat Model:
Compromised Network; trustworthy IEDs

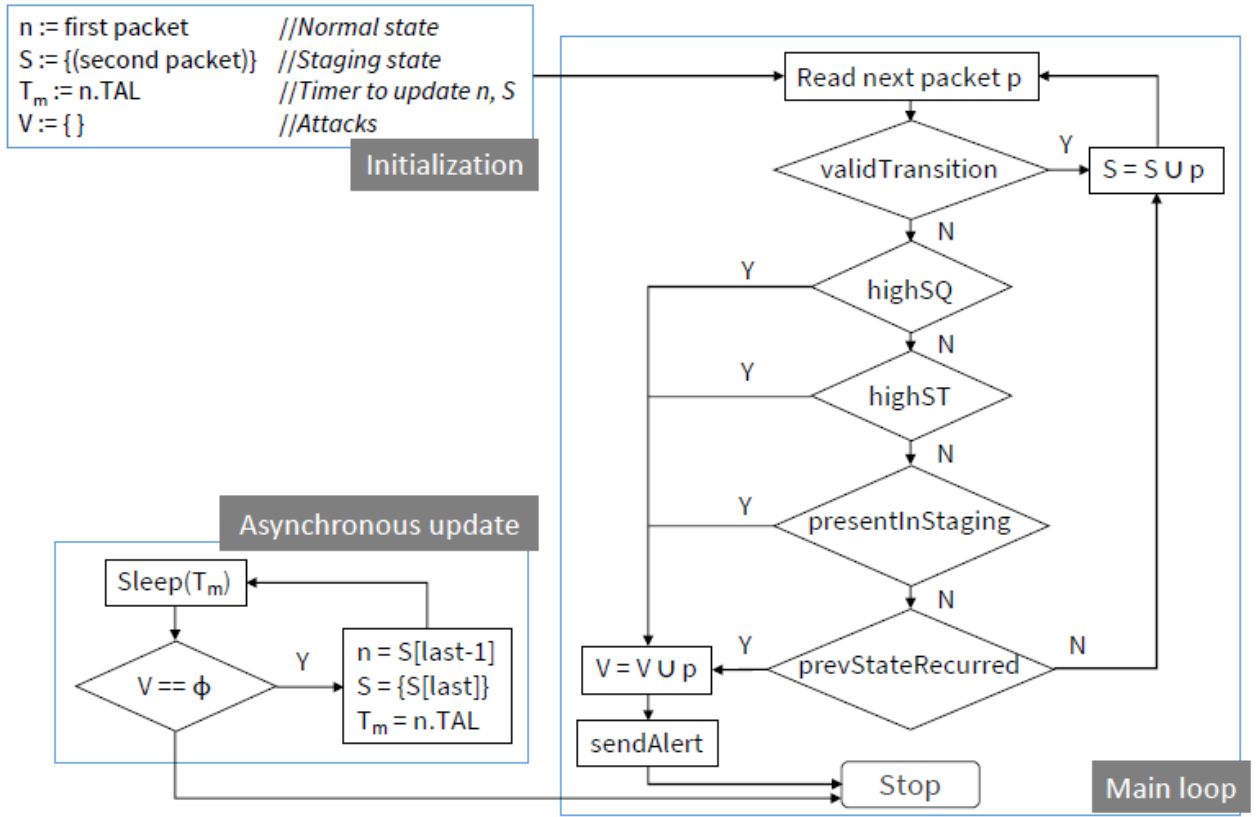
ED4GAP System



Network-Edge Placement of ED4GAP

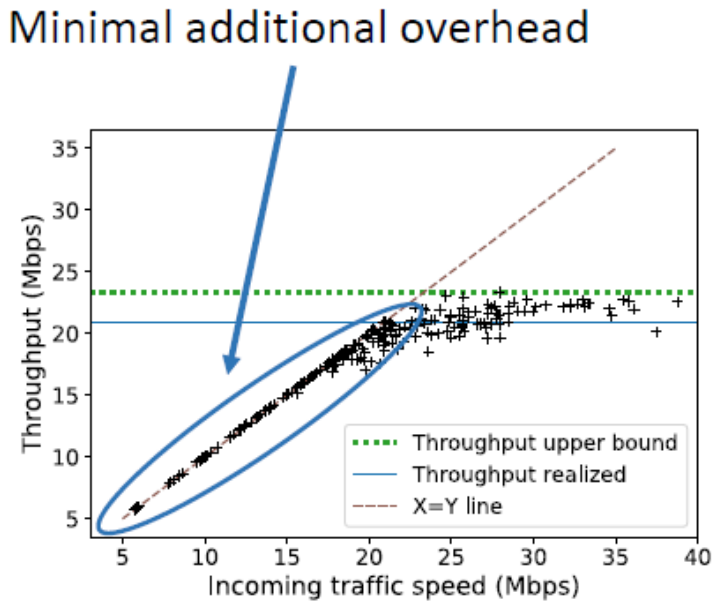


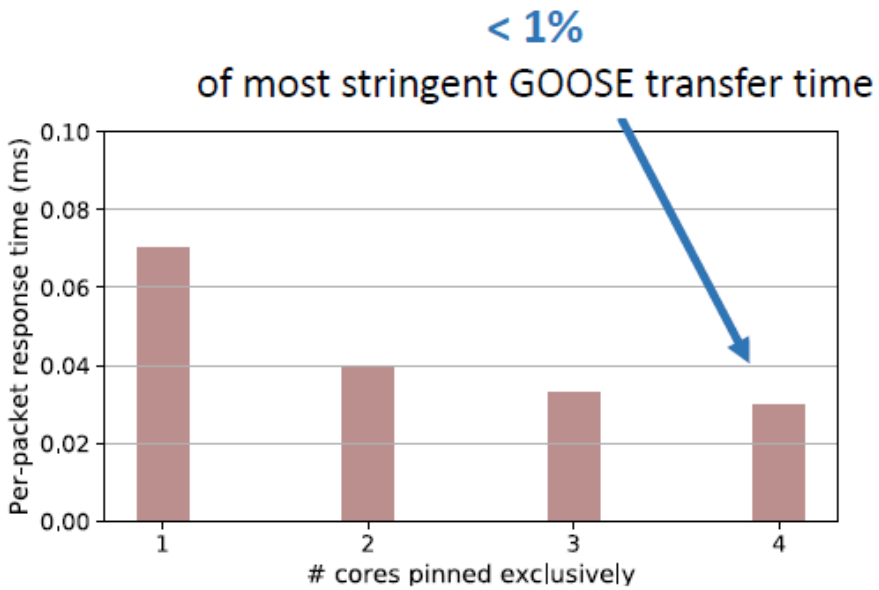
GOOSE Transmission State Machine for StNum and SqNum



Results

- ED4GAP used Zeek network security monitor
- GOOSE traffic was synthesized and replayed [Biswas 2019]
- We have detected all forms of GOOSE poisoning attacks
- Real-time performance
  - Minimal overhead on throughput
  - Response time suitable for GOOSE
  - Systematic approach to analyze bottleneck and improve response time



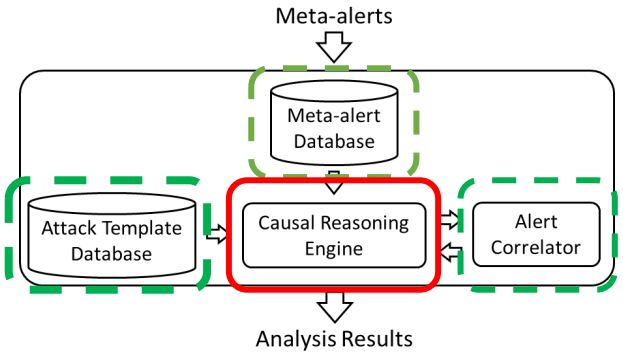


Publication

- Atul Bohara, Jordi Ros-Giralt, Ghada Elbez, Alfonso Valdes, Klara Nahrstedt, William Sanders, “ED4GAP: Efficient detection for GOOSE-based poisoning attacks on IEC 61850 substations”, IEEE SmartGridComm 2020
- Wenyu Ren, Tim Yardley, Klara Nahrstedt, “CAPTAR: Causal-Polytree-Based Anomaly Reasoning for SCADA Networks”, *IEEE Smart Grid Communications (SmartGridComm)*, October 2019
- Wenyu Ren, Tim Yardley, Klara Nahrstedt, “EDMAND: Edge-Based Multi-Level Anomaly Detection for SCADA Networks”, *IEEE Smart Grid Communications (SmartGridComm)*, October 2018.

Additional slides

Causal Reasoning Engine



- Anomaly reasoning algorithm
  - The algorithm takes the meta-alert a and the current attack template set $\boldsymbol{ATS}$ as inputs and outputs a new attack template set $\boldsymbol{ATS}_{new}$.
  - For update to an existing meta-alert, the algorithm updates the nodes containing the meta-alert and initiates belief propagations from those nodes
  - For a new meta-alert, the algorithm tries to match it to potential nodes and performs a belief propagation for every successful match.

Algorithm 2 Anomaly Reasoning Algorithm

Input:
a - meta-alert to be analyzed
$\boldsymbol{ATS}$ - attack template set
Output:
$\boldsymbol{ATS}_{new}$ - new attack template set

procedure AnalyzeAlert($a, \boldsymbol{ATS}$)
  $\boldsymbol{ATS}_{new} \leftarrow \emptyset$
  if a is an update of an existing meta-alert **then**
    for each AT in $\boldsymbol{ATS}$ that has a as a matched alert **do**
      recalculate CS_{total} and $\lambda_{\vec{x}}(x)$ of the matched node X
      start a new belief propagation in AT from node X
    end for
    $\boldsymbol{ATS}_{new} \leftarrow \boldsymbol{ATS}$
  else
    for each AT in $\boldsymbol{ATS}$ **do**
      $\boldsymbol{ATS}_{match} \leftarrow$ MatchAlert(a, AT)
      add $\boldsymbol{ATS}_{match}$ to $\boldsymbol{ATS}_{new}$
    end for
  end if
  return $\boldsymbol{ATS}_{new}$
end procedure

Details at IEEE SmartGridComm 2019