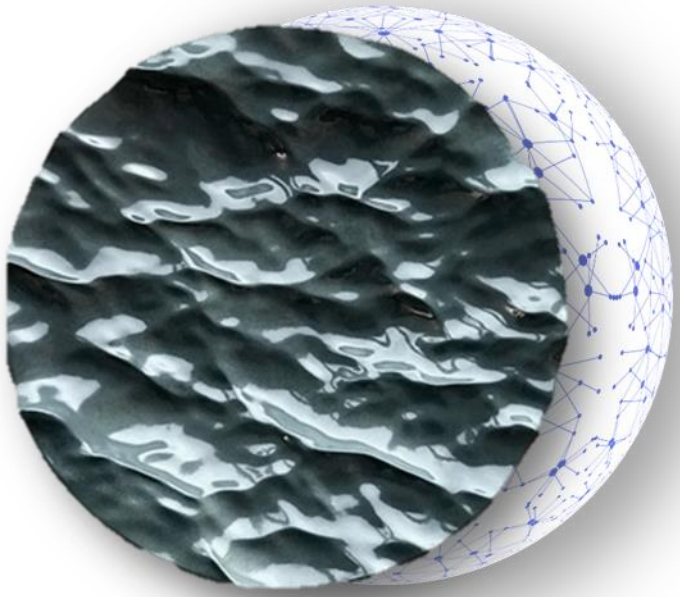
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
Τομέας Υδατικών Πόρων και Περιβάλλοντος

Διπλωματική Εργασία

ΕΦΑΡΜΟΓΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΗ ΔΙΑΓΝΩΣΗ ΚΥΒΕΡΝΟΕΠΙΘΕΣΕΩΝ ΣΕ ΕΤΑΙΡΕΙΕΣ ΝΕΡΟΥ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ

DIPLOMA THESIS

Machine Learning applications for real-time detection of cyber-physical attacks on Water Distribution Systems



Τσιάμη Λυδία-Μαρία

Επιβλέπων: **Μακρόπουλος Χρήστος, Αναπληρωτής Καθηγητής ΕΜΠ**

Οκτώβριος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
Τομέας Υδατικών Πόρων και Περιβάλλοντος

Διπλωματική Εργασία

ΕΦΑΡΜΟΓΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΗ ΔΙΑΓΝΩΣΗ ΚΥΒΕΡΝΟΕΠΙΘΕΣΕΩΝ ΣΕ ΕΤΑΙΡΕΙΕΣ ΝΕΡΟΥ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ

DIPLOMA THESIS

Machine Learning applications for real-time detection of cyber-physical attacks on Water Distribution Systems

Τσιάμη Λυδία-Μαρία

Επιβλέπων: **Μακρόπουλος Χρήστος, Αναπληρωτής Καθηγητής, ΕΜΠ**

Οκτώβριος 2020

ΕΥΧΑΡΙΣΤΙΕΣ / ACKNOWLEDGEMENTS

Η παρούσα διπλωματική εργασία αποτελεί το τελευταίο και πολύ σημαντικό για εμένα κεφάλαιο των προπτυχιακών σπουδών μου στη Σχολή Πολιτικών Μηχανικών του Εθνικού Μετσόβιου Πολυτεχνείου. Είναι πραγματικά αξιοσημείωτη η συνειδητοποίηση ότι η Σχολή διαμόρφωσε τόσο καθοριστικά τον τρόπο σκέψης μου και με εφοδίασε με τόσες πολύτιμες γνώσεις. Σε αυτό το σημείο θα ήθελα να ευχαριστήσω από καρδιάς όλα τα πρόσωπα που συνέβαλλαν, το καθένα με το δικό του τρόπο, στην εκπόνηση αυτής της εργασίας.

Πρωταρχικά, θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου και επιβλέποντα κ. Χρήστο Μακρόπουλο για την εμπιστοσύνη που μου έδειξε με την ανάθεση ενός θέματος που μου προσέφερε την ευκαιρία να ασχοληθώ με ένα ζήτημα που όχι μόνο μου είναι εξαιρετικά ενδιαφέρον, αλλά και μου άνοιξε νέους επιστημονικούς ορίζοντες. Η επιστημονική του κατάρτιση είναι για εμένα αξιοθαύμαστη και η συμβολή του στάθηκε καθοριστική, καθώς με καθοδήγησε και με ενέπνευσε να ασχοληθώ με ζητήματα που κατά το ξεκίνημα της εργασίας, δεν φανταζόμουν καν ότι θα καταφέρω να προσεγγίσω. Με βοήθησε να δοκιμάσω τις δυνάμεις μου και μου παρουσίασε τα πιο πρόσφατα επιστημονικά άρθρα που αποτέλεσαν για εμένα μεγάλη πηγή έμπνευσης και γνώσης.

Οφείλω να ευχαριστήσω ιδιαίτερα και τον Δρ. Πολιτικό Μηχανικό, Παναγιώτη Κοσσιέρη, για την ουσιαστική του συμμετοχή στο στοχαστικό κομμάτι της εργασίας, καθώς και για τις πολύτιμες συμβουλές και το χρόνο που αφιέρωσε στις τηλεδιασκέψεις μας εν μέσω πανδημίας.

Θα ήθελα να ευχαριστήσω και τον επίκουρο καθηγητή κ. Ανδρέα Ευστρατιάδη, καθώς χάρη σε εκείνον ξεκίνησα την προηγούμενη χρονιά την πορεία μου στον τομέα της Μηχανικής Μάθησης, με την ερευνητική εργασία που παρουσιάσαμε στο -καθοριστικό για εμένα- συνέδριο της E.G.U. το 2019, αλλά και τον καθηγητή και πρώην κοσμήτορα της Σχολής κ. Δημήτρη Κουτσογιάννη που κάθε χρόνο δίνει την ευκαιρία στους φοιτητές να συμμετάσχουν σε αυτό το συνέδριο.

Θα ήθελα επίσης να ευχαριστήσω τον Ian Covert, Υποψήφιο Διδάκτορα στο Πανεπιστήμιο της Ουάσιγκτον, ο οποίος πρόθυμα μοιράστηκε μαζί μου μέρος του κώδικά που ανέπτυξε κατά την πρακτική του στη Google AI Healthcare.

Τέλος, ευχαριστώ το φίλο μου Βασίλη για τη στήριξή του καθώς και τους γονείς μου και την αδελφή μου Αθηνά για την αγάπη και τη συμπαράστασή τους.

Τσιάμη Λυδία-Μαρία

Αθήνα, Οκτώβριος 2020

ABSTRACT

Water distribution networks (WDN) deploy digital devices not only to monitor and control utility operations but also to increase automation and ultimately their efficiency. Although their digitalization is essential, it comes with a cost: it exposes the WDN to the risks of a Cyber-Physical System, i.e. cyber-attacks. The overall aim of this diploma thesis is to develop new and improve upon existing machine learning methods for cyber-physical attack detection on Water Distribution Networks. The innovation of this work resides in two main developments (a) the use of novel stochastic methods to generate the water demand timeseries needed to train existing machine learning models, in an effort to improve their overall performance in the presence of uncertainty and (b) the exploration and use of a novel family of machine learning methods that take both the spatial and temporal dimensions of a water network into account, in an effort to improve the ability of the model to represent the water network more accurately. To approach the first objective, we generate new, synthetic datasets for the study of cyber-physical attack detection on water distribution networks by performing simulations on a real medium size WDN under stochastically generated water demands. The second objective is approached by exploring the use of Spatio-Temporal Graph Neural Networks as cyber-physical attack detection tools. Finally, we test the detection performance of various ML algorithms (including SVDD, Autoencoder, Structural Convolutional Neural Networks) on our datasets and preexisting ones as well, and discuss.

Key words: Cyber security, Cyber-physical attacks, Water Distribution Systems, Machine Learning, Convolutional Neural Networks, Autoencoder, Support Vector Data Description classifier, Time-series, Stochastic methods, Water demands

ΕΚΤΕΝΗΣ ΠΕΡΙΛΗΨΗ / EXTENDED ABSTRACT IN GREEK

ΕΙΣΑΓΩΓΗ

Στα πλαίσια της εποχής ψηφιακού εκσυγχρονισμού, ο τομέας διαχείρισης υδατικών πόρων δεν θα μπορούσε να μείνει ανεπηρέαστος. Τα Δίκτυα Διανομής Νερού (ΔΔΝ) χρησιμοποιούν ψηφιακές συσκευές όχι μόνο για την εξασφάλιση της καλής λειτουργίας τους, αλλά και για να ενισχύσουν την αυτοματοποίησή τους και τελικώς να βελτιστοποιήσουν την λειτουργία τους. Παρόλο που η ψηφιοποίηση των ΔΔΝ αποτελεί πλέον προϋπόθεση για την καλή λειτουργία τους, ελλοχεύει συγχρόνως και ο κίνδυνος έκθεσης τους σε κυβερνοεπιθέσεις.

Δεδομένου ότι μια κυβερνοεπίθεση θα είχε καταστροφικές συνέπειες, τελευταία παρατηρείται ένα αυξανόμενο ερευνητικό ενδιαφέρον για την κυβερνοασφάλεια των Υδατικών Υποδομών. Μία από τις σημαντικότερες πτυχές αυτού του νέου ερευνητικού πεδίου είναι η διάγνωση κυβερνοεπιθέσεων σε ΔΔΝ. Στόχος είναι η ανάπτυξη εργαλείων που θα επιτρέπουν τη διάγνωση των κυβερνοεπιθέσεων έγκαιρα, δηλαδή προτού προλάβουν να προκληθούν μη αναστρέψιμες ζημιές στο δίκτυο.

Συχνά σε προβλήματα ανίχνευσης ανωμαλιών εφαρμόζονται μέθοδοι Μηχανικής Μάθησης με πολλά υποσχόμενα αποτελέσματα. Απαραίτητη προϋπόθεση για την εφαρμογή τους είναι η διαθεσιμότητα επαρκών δεδομένων. Στις περιπτώσεις που τα διαθέσιμα δεδομένα είναι περιορισμένα αποτελεί κοινή πρακτική να χρησιμοποιούνται σε συνδυασμό με συνθετικά. Έχει μάλιστα αποδειχθεί στο

[1] ότι στην επιστήμη των δεδομένων (Data Science) η χρήση συνθετικών δεδομένων μπορεί ακόμη και να αντικαταστήσει τη χρήση πραγματικών δεδομένων.

Δυστυχώς όμως στον τομέα των υδατικών πόρων, όπου η ψηφιοποίηση είναι σχετικά πρόσφατη, τα διαθέσιμα δεδομένα είναι κατά κανόνα περιορισμένα. Στις περιπτώσεις δε, που απαιτούνται δεδομένα για τον σχεδιασμό και την ανάλυση των υδατικών συστημάτων, είθισται η αξιοποίηση στοχαστικών μεθόδων.

Για την ανάλυση και προσομοίωση ενός ΔΔΝ είναι απαραίτητη η γνώση των αναμενόμενων καταναλώσεων και η επιστήμη υδατικών πόρων παρέχει αποτελεσματικά εργαλεία για τη στοχαστική μοντελοποίηση της κατανάλωσης του νερού. Υπό αυτό το πρίσμα, είναι εύλογο να υποθέσουμε ότι η εκπαίδευση μοντέλων Μηχανικής Μάθησης με δεδομένα που έχουν προκύψει από προσομοιώσεις με στοχαστικές ζητήσεις νερού θα μπορούσε να συνεισφέρει στον εντοπισμό επιθέσεων σε δίκτυα διανομής νερού.

Επιπλέον, επειδή τα Δίκτυα Διανομής Νερού έχουν εγγενώς δομή γράφου, είναι εξίσου εύλογο να υποτεθεί ότι τα Νευρωνικά Δίκτυα σε Γράφους (Graph Neural Networks) θα μπορούσαν να είναι χρήσιμα εργαλεία για τον εντοπισμό κυβερνοεπιθέσεων. Τα Νευρωνικά Δίκτυα σε Γράφους, εμπνευσμένα από τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks), είναι αλγόριθμοι βαθιάς μηχανικής μάθησης που είναι σε θέση να ενσωματώνουν στην αρχιτεκτονική τους τις χωρικές πληροφορίες ενός γράφου.

Με βάση τα παραπάνω, ο κύριος σκοπός της παρούσας διπλωματικής εργασίας είναι να αναπτυχθούν νέες αλλά και να βελτιωθούν οι υπάρχουσες μέθοδοι μηχανικής μάθησης για την ανίχνευση κυβερνοεπιθέσεων στα Δίκτυα Διανομής

Νερού. Η καινοτομία της εργασίας έγκειται σε δύο βασικούς παράγοντες: (α) στη χρήση στοχαστικών μεθόδων για τη δημιουργία συνθετικών χρονοσειρών ζήτησης νερού που είναι απαραίτητες για την εκπαίδευση των μοντέλων μηχανικής μάθησης, σε μια προσπάθεια βελτίωσης της συνολικής απόδοσής τους κάτω από τις συνθήκες αβεβαιότητας και (β) στην εξερεύνηση και χρήση μιας νέας κατηγορίας μεθόδων μηχανικής μάθησης που λαμβάνει υπόψη τόσο τις χωρικές όσο και τις χρονικές διαστάσεις ενός δικτύου νερού και αποσκοπεί στη βελτίωση της ικανότητας του μοντέλου να αναπαριστά με μεγαλύτερη ακρίβεια το δίκτυο νερού.

ΣΤΟΙΧΕΙΑ ΘΕΩΡΙΑΣ

Κατηγορίες Κυβερνοφυσικών επιθέσεων σε Δίκτυα Διανομής Νερού

Ένας τρόπος ταξινόμησης των κυβερνο-φυσικών επιθέσεων (cyber-attacks) σε ένα δίκτυο διανομής νερού είναι με βάση το τμήμα του δικτύου που στοχοποιείται. Τα στοιχεία ενός δικτύου διανομής νερού που ορίζονται ως ευάλωτα σε κυβερνοεπιθέσεις είναι: οι αισθητήρες (sensors), οι υδραυλικοί ενεργοποιητές (actuators), οι προγραμματιζόμενοι λογικοί ελεγκτές (Programmable Logic Controllers – PLCs), το σύστημα τηλελέγχου και τηλεχειρισμού (SCADA), καθώς και οι μεταξύ τους ασύρματες και ενσύρματες ζεύξεις (communication links).

Οι επιθέσεις διαχωρίζονται επίσης, όπως υποδηλώνει και το όνομά τους, σε φυσικές επιθέσεις και σε επιθέσεις στον κυβερνοχώρο. Οι αισθητήρες και οι υδραυλικοί ενεργοποιητές είναι ευάλωτοι σε φυσικές επιθέσεις, καθώς απαιτείται πρόσβαση στη φυσική υποδομή για να λάβει χώρα η επίθεση. Επιπλέον στις φυσικές επιθέσεις κατατάσσονται και οι επιθέσεις στις ενσύρματες ζεύξεις μεταξύ των συνιστωσών του δικτύου. Αν και επίθεση αυτού του είδους μπορεί να μοιάζει απίθανη, είναι δόκιμο να λαμβάνεται υπόψη κατά τη θωράκιση ενός δικτύου διανομής νερού, καθώς υπάρχει πάντα η περίπτωση ένας υδραυλικός ενεργοποιητής ή ένας αισθητήρας να βρίσκεται σε μια απομακρυσμένη (ή χωρίς επαρκή παρακολούθηση) περιοχή που να είναι προσβάσιμη σε έναν εισβολέα.

Οι επιθέσεις στον κυβερνοχώρο είναι οι επιθέσεις που γίνονται ενάντια στην ασύρματη ζεύξη μεταξύ των συνιστωσών του δικτύου. Όλες οι επιθέσεις ανεξάρτητα από το στόχο τους και από το αν γίνονται στον κυβερνοχώρο ή αν είναι φυσικές, ανήκουν σε μία από τις τρείς κατηγορίες:

Επιθέσεις **υποκλοπής** (eavesdropping attacks): στοχεύουν στην υποκλοπή ευαίσθητων πληροφοριών όπως η κατάσταση και ο τρόπος συμπεριφοράς του δικτύου διανομής νερού. Οι επιθέσεις υποκλοπής αποτελούν συνήθως το πρώτο στάδιο για έναν επιτιθέμενο και τον βοηθούν να σχεδιάσει πιο προηγμένες μορφές επιθέσεων.

Επιθέσεις **άρνησης εξυπηρέτησης** (Denial of Service – DoS attacks): καθιστούν το σύστημα μη διαθέσιμο ή παρεμποδίζουν την επικοινωνία των στοιχείων του δικτύου, άρα και την ομαλή λειτουργία του.

Επιθέσεις **εξαπάτησης** (deception attacks): έχουν σκοπό την τροποποίηση και μετάδοση εσφαλμένων πληροφοριών στο δίκτυο. Αυτή η επίθεση μπορεί όχι μόνο να διαταράξει την ομαλή λειτουργία του δικτύου αλλά και να χρησιμοποιηθεί ως εργαλείο συγκάλυψης των επιθέσεων που λαμβάνουν χώρα, από το σύστημα τηλελέγχου και τηλεχειρισμού (SCADA).

Η επιθέσεις συγκάλυψης (deception attacks) είναι και η βασική πρόκληση που έχουν να αντιμετωπίσουν οι αλγόριθμοι διάγνωσης επιθέσεων. Ένας καλός αλγόριθμος ανίχνευσης κυβερνοφυσικών επιθέσεων είναι αυτός που έχει τη δυνατότητα να διαγιγνώσκει συναφείς ανωμαλίες (contextual anomalies). Συναφείς ανωμαλίες είναι παρατηρήσεις οι οποίες αποκλίνουν σε ένα συγκεκριμένο περιβάλλον και μόνο σε αυτό, έχουν δηλαδή τιμές οι οποίες ανήκουν μεν στα προηγούμενα ιστορικά τους όρια, αλλά θεωρούνται ανώμαλες εντός ενός συγκεκριμένου χρονικού πλαισίου και με βάση τις παρατηρήσεις που έχουν προηγηθεί.

Μια τυπική μέθοδος εντοπισμού ανωμαλιών

Έστω ότι έχουμε ένα σύνοδο δεδομένων X και ότι η κατανομή των δεδομένων του περιέχει καθαρά και ανώμαλα δεδομένα:

$$p_{full}(x,y) \sim p(y=1)p(x|y=1) + p(y=0)p(x|y=0)$$
$$p_{normal}(x) \sim p(x|y=0)$$
$$p_{abnormal}(x) \sim p(x|y=1)$$

Ο στόχος των προβλημάτων ανίχνευσης ανωμαλιών είναι όσο το δυνατόν καλύτερη εκτίμηση των κατανομών $p_{normal}(x)$ and $p_{abnormal}(x)$.

Ένα χαρακτηριστικό του προβλήματος ανίχνευσης επιθέσεων σε ένα σύστημα διανομής νερού, είναι ότι οι επιθέσεις που έχουμε στην διάθεση μας αποτελούν μόνο ένα μικρό ποσοστό του συνόλου των επιθέσεων που θα μπορούσαν να συμβούν σε ένα δίκτυο, καθώς δεν είναι εφικτό να σχεδιαστεί και να μοντελοποιηθεί κάθε πιθανό σενάριο. Για αυτό, η πιο αποτελεσματική μέθοδος για τον εντοπισμό επιθέσεων σε Δίκτυα Διανομής νερού είναι μέσω ημι-επιβλεπόμενης μάθησης.

Στο πρόβλημα της ημι-επιβλεπόμενης μάθησης έχουμε διαθέσιμο ένα σύνολο δεδομένων που εμπεριέχει μόνο καθαρά δεδομένα (δηλαδή δεδομένα κανονικής λειτουργίας του δικτύου) και αυτό χρησιμοποιείται για την εκπαίδευση ενός αλγορίθμου μηχανικής μάθησης με στόχο των εντοπισμό των ανωμαλιών (επιθέσεων) σε ένα σύνολο ελέγχου (test dataset).

$$D_{train} = X_{train} \sim p_{normal}(x)$$
$$D_{test} = X_{test} \sim p_{full}(x)$$

Αφού ο αλγόριθμος χρησιμοποιήσει το σύνολο εκπαίδευσης για να μοντελοποιήσει την κατανομή των καθαρών δεδομένων, στη συνέχεια αναθέτει μια βαθμολογία σε κάθε δείγμα ανάλογα με το κατά πόσο μπορεί να θεωρηθεί ως ανωμαλία (anomaly score). Τελικά, το δείγμα επισημειώνεται ως αναμενόμενο ή μη, με βάση ένα όριο αποκοπής. Πιο συγκεκριμένα, ένα δείγμα χαρακτηρίζεται ως ανώμαλο, όταν η βαθμολογία του είναι μεγαλύτερη από ένα προκαθορισμένο όριο αποκοπής.

Αλγόριθμοι Μηχανικής Μάθησης για τον εντοπισμό επιθέσεων σε δίκτυα διανομής νερού

Παρακάτω παρουσιάζονται εν συντομία τα χαρακτηριστικά των αλγορίθμων μηχανική μάθησης που αξιοποιήθηκαν στην παρούσα διπλωματική για τη διάγνωση επιθέσεων σε δίκτυα νερού.

Support Vector Data Description Classifier

Ο ταξινομητής Support Vector Data Description (SVDD) δημιουργεί ένα σφαιρικό σύνορο γύρω από ένα σύνολο δεδομένων πολλών μεταβλητών χρησιμοποιώντας συναρτήσεις πυρήνα. Υπολογίζοντας την απόσταση ενός νέου δείγματος από το σφαιρικό αυτό όριο, μπορεί κανείς να αποφασίσει αν αυτό το δείγμα ανήκει στην κατανομή των δεδομένων εκπαίδευσης ή όχι. Ο ταξινομητής SVDD δημιουργεί το σφαιρικό σύνορο, λύνοντας το ακόλουθο πρόβλημα βελτιστοποίησης:

$$\min_{R,\alpha} R \; Subject\ to \; \|\varphi(x_i) - a\| \leq R$$

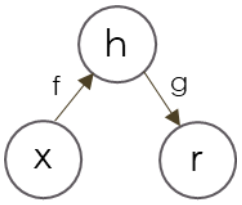
όπου R και a = μεταβλητές απόφασης και $\varphi(x_i)$ = μια συνάρτηση μετασχηματισμού των δεδομένων εισόδου σε δεδομένα υψηλότερης διάστασης. Μετά το στάδιο της εκπαίδευσης και αφού έχουν οριστεί οι τιμές των R_{opt} και α_{opt}, μπορεί να υπολογιστεί για κάθε νέο δείγμα η απόστασή του από το σφαιρικό όριο από τη σχέση:

$$DV = \left\|\varphi(x_i) - \alpha_{opt}\right\| - R_{opt}$$

Όταν η τιμή DV για ένα δείγμα είναι θετική ($DV > 0$), τότε το δείγμα βρίσκεται εκτός του σφαιρικού χώρου που ορίστηκε με βάση τα δεδομένα εκπαίδευσης και χαρακτηρίζεται ως ανωμαλία.

Αυτοκωδικοποιητές – Autoencoders

Ένας αυτοκωδικοποιητής είναι ένα νευρωνικό δίκτυο που προσπαθεί να αντιγράψει την είσοδό του στην έξοδό του. Με άλλα λόγια, εκπαιδεύεται στο να αναπαριστά ένα δεδομένο εισόδου x στην έξοδο του r μέσω μιας εσωτερικής αναπαράστασης h. Ο αυτοκωδικοποιητής αποτελείται από δύο μέρη: έναν κωδικοποιητή f (ο οποίος αναπαριστά το x σε μία μικρότερη διάσταση h, $h = f(x)$) και έναν αποκωδικοποιητή g (που παράγει μία ανακατασκευή r του h, $r = g(h)$).



Εικόνα 1: Βασική δομή ενός αυτοκωδικοποιητή.

Λόγω της αρχιτεκτονικής τους, οι αυτοκωδικοποιητές έχουν την ικανότητα να ανακαλύπτουν συσχετίσεις μεταξύ των δεδομένων και αποτελούν χρήσιμα εργαλεία στην ανίχνευση ανωμαλιών.

Structural Convolutional Neural Networks

Τα Δομικά Συνελικτικά Νευρωνικά Δίκτυα είναι ένας αλγόριθμος βαθιάς μηχανικής μάθησης, ο οποίος είναι εμπνευσμένος από τα νευρωνικά δίκτυα σε γράφους (Graph Neural Networks – GNNs) και τα μονοδιάστατα συνελικτικά νευρωνικά δίκτυα (1D CNNs). Τα SCNN έχουν τη δυνατότητα να μοντελοποιούν δεδομένα τα οποία έχουν χωροχρονικά χαρακτηριστικά ενσωματώνοντας στην αρχιτεκτονική τους τον πίνακα γειτνίασης (adjacency matrix) ενός γράφου. Αυτό το χαρακτηριστικό τους επιτρέπει να αποτυπώνουν καλύτερα τις συσχετίσεις μεταξύ των στοιχείων ενός

γράφου και να κάνουν ακριβέστερες προβλέψεις σε δεδομένα χρονοσειρών με δομή γράφου.

ΔΕΔΟΜΕΝΑ

Τα δεδομένα που χρησιμοποιήσαμε προέρχονται από ένα διαγωνισμό με θέμα την κυβερνοασφάλεια των συστημάτων διανομής νερού. Ο διαγωνισμός που ονομάζεται BATADAL είχε ως στόχο τη διάγνωση των επιθέσεων σε ένα πραγματικό δίκτυο διανομής νερού, που ονομάζεται C-Town.

Τα διαθέσιμα δεδομένα από αυτόν τον διαγωνισμό είναι τα ακόλουθα:

- Τρία σύνολα δεδομένων από τρεις διαφορετικές προσομοιώσεις, με καταγεγραμμένες ωριαίες μετρήσεις SCADA που αφορούν 43 διαφορετικές μεταβλητές του δικτύου.

- dataset03: σύνολο δεδομένων με μετρήσεις SCADA διάρκειας ενός έτους. Στο συγκεκριμένο σύνολο δεδομένων δεν περιέχεται καμία επίθεση.
- dataset04: σύνολο δεδομένων που περιέχει μετρήσεις από μια προσομοίωση διάρκειας 6 μηνών και περιλαμβάνει 7 κυβερνοεπιθέσεις.
- testdataset: σύνολο δεδομένων που περιέχει μετρήσεις διάρκειας 3 μηνών και περιλαμβάνει 7 διαφορετικές επιθέσεις.

-Ένας πίνακας που περιγράφει τα σενάρια επίθεσης των "dataset04" και "testdataset".

-Ένα αρχείο που ονομάζεται "ctown.inp", περιέχει τοπογραφικές, υδραυλικές και υδρολογικές πληροφορίες του C-Town και επιτρέπει την προσομοίωσή του σε περιβάλλον EPANET. Ανάμεσα στις πληροφορίες που περιέχονται σε αυτό το αρχείο είναι και:

- η μέση μηνιαία ζήτηση νερού σε κάθε κόμβο του δικτύου.
- μια τυπική διακύμανση ζήτησης νερού διάρκειας μιας εβδομάδας για κάθε μία από τις πέντε ζώνες του δικτύου.

ΜΕΘΟΔΟΛΟΓΙΑ

Η μεθοδολογία που ακολουθήσαμε για την επίτευξη του σκοπού της εργασίας χωρίζεται σε δύο μέρη:

Α. Στο πρώτο μέρος, ο στόχος είναι να δημιουργηθούν νέα σύνολα δεδομένων, αντίστοιχα με αυτά που είναι διαθέσιμα στο BATADAL. Η διαδικασία που ακολουθούμε είναι η εξής:

- Αρχικά χρησιμοποιώντας τη μεθοδολογία του [2], δημιουργούμε συνθετικές χρονοσειρές μοτίβων ζήτησης νερού. Με αυτή τη μεθοδολογία, προκύπτουν δύο διαφορετικά είδη μοτίβων ζήτησης: το ένα έχει προσομοιωθεί με χρήση της κατανομής Βήτα και το άλλο με βάση την κατανομή Γάμμα. Επιπλέον, κατασκευάζεται και ένα τρίτο μοτίβο ζητήσεων με μία απλοϊκή μέθοδο, προκειμένου να έχουμε ένα μέτρο σύγκρισης για τη συνεισφορά του κάθε μοτίβου στη διάγνωση επιθέσεων.
- Με τη βοήθεια του εργαλείου epanetCPA αναπαραγάγουμε τα σενάρια επιθέσεων του BATADAL και τρέχουμε νέες προσομοιώσεις με τα συνθετικά μοτίβα ζήτησης νερού, για να δημιουργήσουμε τα δικά μας σετ δεδομένων,

τα οποία απαιτούνται για την εκπαίδευση των αλγορίθμων μηχανικής μάθησης.

Β. Το δεύτερο μέρος αφορά τον εντοπισμό των επιθέσεων με χρήση αλγορίθμων μηχανικής μάθησης. Στην παρούσα διπλωματική χρησιμοποιούνται τρεις διαφορετικοί αλγόριθμοι μηχανικής μάθησης:

- Οι δύο είναι αλγόριθμοι [3], [4] που βασίζονται σε μεθοδολογίες που έχουν δημοσιευθεί και εφαρμοσθεί στα σύνολα δεδομένων του διαγωνισμού BATADAL. Η πρώτη μεθοδολογία χρησιμοποιεί έναν ταξινομητή SVDD, ενώ η δεύτερη κάνει χρήση ενός Αυτοκωδικοποιητή (Autoencoder).
- Ο τρίτος αλγόριθμος βασίζεται στα έργα των [5], [6] που μοντελοποιούν χρονοσειρές με δομή γράφου.

ΑΠΟΤΕΛΕΣΜΑΤΑ

Τα κυριότερα αποτελέσματα της μεθόδου μας συνοψίζονται στους παρακάτω πίνακες:

SVDD

Πίνακας 1: Αποτελέσματα SVDD – Η απόδοσή του υπολογίστηκε με βάση τη μετρική S όπως ορίστηκε στο διαγωνισμό BATADAL.

train batadal	train random	train beta	train gamma
test batadal	test batadal	test batadal	test batadal
The Train Score is 0.956 The Test Score is 0.954 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.932 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.919 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.885 The optimal L is 3 The optimal TH is 0.012
test random	test random	test random	test random
The Train Score is 0.956 The Test Score is 0.781 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.897 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.876 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.877 The optimal L is 3 The optimal TH is 0.012
test beta	test beta	test beta	test beta
The Train Score is 0.956 The Test Score is 0.805 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.922 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.922 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.906 The optimal L is 3 The optimal TH is 0.012
test gamma	test gamma	test gamma	test gamma
The Train Score is 0.956 The Test Score is 0.791 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.903 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.879 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.881 The optimal L is 3 The optimal TH is 0.012

Αυτοκωδικοποιητής

Πίνακας 2: Μέση απόδοση (μετά από 10 διαφορετικές διαδικασίες εκπαίδευσης) του αυτοκωδικοποιητή σε κάθε ένα από τα διαθέσιμα σύνολα δεδομένων.

Model	FP	FN	TP	Rec	Pre	F1 score	S score (BATADAL)
							dataset_r06
random	92	151	341	0.693 ± 0.043	0.795 ± 0.065	0.738 ± 0.022	0.82 ± 0.045
gamma	59	149	343	0.697 ± 0.071	0.856 ± 0.03	0.766 ± 0.041	0.825 ± 0.069
beta	84	144	348	0.707 ± 0.06	0.812 ± 0.053	0.753 ± 0.025	0.83 ± 0.048
batadal	370	117	375	0.761 ± 0.044	0.504 ± 0.017	0.606 ± 0.016	0.861 ± 0.018
							dataset_r03
random	58	87	320	0.785 ± 0.077	0.848 ± 0.037	0.813 ± 0.045	0.882 ± 0.029
gamma	49	93	314	0.772 ± 0.13	0.868 ± 0.013	0.81 ± 0.078	0.876 ± 0.047
beta	54	88	319	0.784 ± 0.082	0.86 ± 0.037	0.817 ± 0.04	0.886 ± 0.025
batadal	176	76	331	0.812 ± 0.06	0.653 ± 0.014	0.723 ± 0.024	0.877 ± 0.02
							dataset_g06
random	173	151	341	0.692 ± 0.041	0.668 ± 0.057	0.678 ± 0.029	0.815 ± 0.036
gamma	86	142	350	0.711 ± 0.074	0.81 ± 0.054	0.753 ± 0.03	0.829 ± 0.068
beta	124	141	351	0.714 ± 0.064	0.748 ± 0.059	0.727 ± 0.028	0.838 ± 0.043
batadal	535	125	367	0.745 ± 0.041	0.407 ± 0.009	0.526 ± 0.013	0.834 ± 0.026
							dataset_g03
random	84	106	301	0.739 ± 0.075	0.783 ± 0.033	0.758 ± 0.038	0.878 ± 0.03
gamma	47	113	294	0.723 ± 0.117	0.866 ± 0.034	0.782 ± 0.073	0.866 ± 0.044
beta	60	107	300	0.737 ± 0.072	0.836 ± 0.042	0.781 ± 0.036	0.874 ± 0.029
batadal	244	88	319	0.783 ± 0.065	0.565 ± 0.012	0.656 ± 0.031	0.804 ± 0.025
							dataset_b06
random	140	156	336	0.683 ± 0.043	0.713 ± 0.064	0.695 ± 0.029	0.837 ± 0.039
gamma	73	155	337	0.685 ± 0.075	0.826 ± 0.042	0.745 ± 0.041	0.826 ± 0.094
beta	92	143	349	0.709 ± 0.073	0.798 ± 0.054	0.747 ± 0.032	0.854 ± 0.055
batadal	461	104	388	0.789 ± 0.058	0.458 ± 0.011	0.579 ± 0.017	0.875 ± 0.015
							dataset_b03
random	64	103	304	0.747 ± 0.077	0.829 ± 0.051	0.783 ± 0.045	0.893 ± 0.024
gamma	45	105	302	0.742 ± 0.128	0.871 ± 0.016	0.796 ± 0.081	0.896 ± 0.038
beta	53	98	309	0.759 ± 0.083	0.859 ± 0.048	0.802 ± 0.045	0.9 ± 0.025
batadal	206	86	321	0.789 ± 0.064	0.61 ± 0.013	0.687 ± 0.027	0.886 ± 0.02
							batadal_06
random	288	114	378	0.768 ± 0.083	0.704 ± 0.251	0.698 ± 0.144	0.788 ± 0.038
gamma	499	113	379	0.77 ± 0.154	0.665 ± 0.312	0.642 ± 0.19	0.753 ± 0.104
beta	595	104	388	0.788 ± 0.107	0.52 ± 0.265	0.575 ± 0.147	0.764 ± 0.047
batadal	63	109	383	0.779 ± 0.049	0.861 ± 0.036	0.816 ± 0.025	0.821 ± 0.028
							batadal_03
random	137	88	319	0.783 ± 0.079	0.774 ± 0.199	0.757 ± 0.098	0.889 ± 0.021
gamma	236	90	317	0.78 ± 0.136	0.728 ± 0.262	0.705 ± 0.147	0.836 ± 0.096
beta	273	77	331	0.812 ± 0.084	0.637 ± 0.232	0.681 ± 0.116	0.842 ± 0.063
batadal	33	84	323	0.794 ± 0.059	0.909 ± 0.026	0.846 ± 0.034	0.914 ± 0.018

SCNN (Structural Convolutional Neural Network)

Πίνακας 3: Μέση απόδοση (μετά από 10 διαφορετικές διαδικασίες εκπαίδευσης) του μοντέλου SCNN σε κάθε ένα από τα διαθέσιμα σύνολα δεδομένων.

MODEL	FP	FN	TP	Recall	Precision	F1 score	S score (BATADAL)
							dataset_r06
random	42	172	320	0.65 ± 0.026	0.886 ± 0.028	0.749 ± 0.012	0.853 ± 0.034
gamma	28	193	299	0.607 ± 0.025	0.913 ± 0.013	0.729 ± 0.019	0.806 ± 0.051
beta	66	172	320	0.651 ± 0.032	0.835 ± 0.059	0.729 ± 0.01	0.835 ± 0.029
batadal	542	104	388	0.788 ± 0.042	0.425 ± 0.058	0.549 ± 0.041	0.907 ± 0.008
							dataset_r03
random	43	107	300	0.737 ± 0.036	0.875 ± 0.014	0.8 ± 0.02	0.922 ± 0.013
gamma	36	119	288	0.708 ± 0.05	0.89 ± 0.009	0.788 ± 0.03	0.912 ± 0.017
beta	49	102	305	0.75 ± 0.036	0.862 ± 0.023	0.801 ± 0.017	0.922 ± 0.014
batadal	246	48	359	0.882 ± 0.031	0.597 ± 0.047	0.711 ± 0.026	0.936 ± 0.005
							dataset_g06
random	84	167	326	0.662 ± 0.023	0.796 ± 0.027	0.722 ± 0.011	0.883 ± 0.008
gamma	41	175	317	0.643 ± 0.011	0.886 ± 0.013	0.746 ± 0.005	0.855 ± 0.029
beta	80	159	333	0.678 ± 0.025	0.808 ± 0.039	0.736 ± 0.011	0.885 ± 0.016
batadal	590	104	388	0.789 ± 0.033	0.402 ± 0.046	0.531 ± 0.035	0.903 ± 0.009
							dataset_g03
random	47	102	305	0.75 ± 0.042	0.867 ± 0.017	0.803 ± 0.023	0.929 ± 0.01
gamma	36	120	287	0.705 ± 0.055	0.889 ± 0.008	0.785 ± 0.035	0.916 ± 0.019
beta	53	92	315	0.774 ± 0.037	0.858 ± 0.025	0.813 ± 0.015	0.935 ± 0.008
batadal	321	43	364	0.895 ± 0.015	0.536 ± 0.05	0.669 ± 0.037	0.928 ± 0.007
							dataset_b06
random	53	175	317	0.644 ± 0.031	0.857 ± 0.032	0.734 ± 0.017	0.889 ± 0.018
gamma	31	199	293	0.595 ± 0.021	0.904 ± 0.023	0.717 ± 0.012	0.833 ± 0.055
beta	52	165	328	0.666 ± 0.034	0.865 ± 0.034	0.751 ± 0.015	0.895 ± 0.02
batadal	543	102	390	0.793 ± 0.029	0.425 ± 0.051	0.551 ± 0.038	0.905 ± 0.008
							dataset_b03
random	51	107	300	0.738 ± 0.045	0.856 ± 0.011	0.792 ± 0.025	0.927 ± 0.011
gamma	46	123	284	0.698 ± 0.054	0.861 ± 0.006	0.77 ± 0.034	0.914 ± 0.017
beta	50	99	308	0.756 ± 0.032	0.859 ± 0.011	0.804 ± 0.018	0.932 ± 0.008
batadal	225	58	349	0.858 ± 0.022	0.614 ± 0.054	0.714 ± 0.031	0.933 ± 0.005
							batadal06
random	24	238	255	0.517 ± 0.041	0.916 ± 0.015	0.66 ± 0.033	0.855 ± 0.015
gamma	19	257	235	0.477 ± 0.03	0.925 ± 0.018	0.629 ± 0.024	0.842 ± 0.012
beta	39	222	270	0.549 ± 0.038	0.878 ± 0.045	0.674 ± 0.021	0.862 ± 0.017
batadal	65	181	311	0.632 ± 0.042	0.83 ± 0.031	0.716 ± 0.02	0.891 ± 0.01
							batadal03
random	30	163	244	0.6 ± 0.052	0.891 ± 0.007	0.716 ± 0.038	0.883 ± 0.014
gamma	28	177	231	0.566 ± 0.042	0.893 ± 0.006	0.692 ± 0.032	0.877 ± 0.011
beta	34	158	250	0.613 ± 0.046	0.881 ± 0.013	0.722 ± 0.031	0.887 ± 0.012
batadal	45	122	285	0.7 ± 0.038	0.866 ± 0.023	0.773 ± 0.019	0.906 ± 0.009

ΣΥΜΠΕΡΑΣΜΑΤΑ

Μερικά από τα βασικότερα συμπεράσματα αυτής της εργασίας είναι τα ακόλουθα:

Η επιλογή του συνόλου εκπαίδευσης σχετίζεται άμεσα με την απόδοση του αλγόριθμου. Αυτό παρατηρήθηκε ειδικά στην περίπτωση του αυτοκωδικοποιητή, όπου η χρήση στοχαστικών συνόλων δεδομένων για εκπαίδευση, βελτίωσε την απόδοσή τους στον εντοπισμό επιθέσεων.

Η απόδοση των αυτοκωδικοποιητών εξαρτάται σε μεγάλο βαθμό από το σύνολο δεδομένων με βάση το οποίο θα ρυθμίσουμε το κατώφλι ανωμαλίας. Αντίθετα στα SCNN παρατηρήθηκε ότι η απόδοσή τους ήταν αρκετά καλή χωρίς να χρειάζεται να γίνει πρώτα αλλαγή στο σύνολο ρύθμισης κατωφλιού.

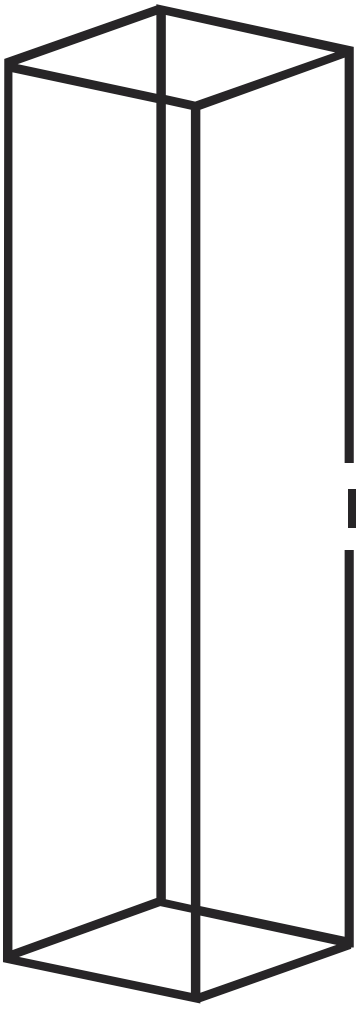
Η επιλογή μίας μόνο μετρικής απόδοσης έχει πολύ μεγάλη σημασία για έναν αλγόριθμο εντοπισμού ανωμαλιών. Καταρχήν επιτρέπει να συγκρίνουμε μεταξύ τους διαφορετικά μοντέλα. Επιπλέον όμως, καθορίζει και την τελική απόδοση ενός μοντέλου, όπως είδαμε στην περίπτωση του SVDD που απέτυχε να διαγνώσει επιθέσεις όταν η μετρική απόδοσής του για την επιλογή κατωφλιού ανωμαλίας, ήταν αυτή που είχε προταθεί από το διαγωνισμό BATADAL.

Τα πρώτα αποτελέσματα δείχνουν ότι τα μοντέλα SCNN φαίνεται να είναι λιγότερο ευαίσθητα (σε σχέση με τους αυτοκωδικοποιητές) στις εντονότερες διακυμάνσεις που έχουν οι στοχαστικές ζητήσεις, καθώς παρατηρούνται μικρότερες διαφορές μεταξύ στοχαστικών και μη στοχαστικών μοντέλων.

CONTENTS

LIST OF TABLES

LIST OF FIGURES



INTRODUCTION

INTRODUCTION

General Context

In the era of unprecedented technological advancements, the water sector is going under digital transformation. Water distribution networks (WDN) deploy digital devices not only to monitor and control utility operations, but to increase automation and ultimately their efficiency. Although water digitalization is essential, it comes with a cost: it exposes the WDN to the risks of a Cyber-Physical System, i.e. cyber-attacks.

Since the impact of a potential attack could be enormous, the research interest on the security of Water Infrastructures is growing. One critical aspect of ongoing research is the ability to detect attacks [7]. Even if an attacker hacks the WDN, a lot of damage could be avoided if the attack is promptly detected.

Machine Learning methods are implemented in anomaly detection problems with promising results. The creation of reliable ML models requires a vast amount of data for training. This is not the case in the water sector, whom digitalization is recent and the data availability is poor. For decades instead, water resource systems analysis is relying on stochastic methods to tackle the hydrological randomness and uncertainty [8].

When real data are limited, the use of synthetic data in Machine Learning is frequent. It has been proven that the use of synthetic data can replace the use of real data in data science [1]. Furthermore, stochastic modelling of water demand is dominating in water resources design and management [2]. In this respect, we evaluate whether training Machine Learning models with stochastically generated water demands will improve their performance in detecting anomalous behavior in the network. Taking into account that synthetic timeseries are providing a more realistic representation of water demands, we expect the ML model to generalize better to unknown data.

Since a WDN has graph structure, we also implement Graph Neural Networks in the detection of cyber-physical attacks and compare them with baseline methods [3], [4]. Graph Neural Networks, inspired by Convolutional Neural Networks, are able to take into account the spatial information of arbitrary graph structures. Our intension is to examine whether their inherent ability to understand relations between the nodes of a graph, i.e. the nodes of a Water Distribution Network, makes them valuable tools in anomaly detection. This is because, when a hacker takes control of the WDN, the data transmitted are altered in an effort to conceal the attack.

Aim

The overall aim of this diploma thesis is to develop new and improve upon existing machine learning methods for cyber-physical attack detection on Water Distribution Networks (WDN). The innovation of this work resides in two main developments (a) the use of novel stochastic methods to generate the water

demand timeseries needed to train existing machine learning models, in an effort to improve their overall performance in the presence of uncertainty and (b) the exploration and use of a novel family of machine learning methods that take both the spatial and temporal dimensions of a water network into account, in an effort to improve the ability of the model to represent the water network more accurately.

Thesis Structure

The thesis is structured into six chapters as follows:

In the **first** chapter we present a brief overview of similar to the subject, state-of-the-art research. We focus on methods of generating stochastically, water demands, detecting cyber-physical attacks on water distribution networks and present some of the research done on spatiotemporal problems with the use of Graph Neural Networks.

In the **second** chapter we make a presentation of the basic theoretical tools used and we provide some insight on the rationale behind approaching an anomaly detection problem.

In the **third** chapter we present the available datasets and the water distribution network we applied our methodology on.

In the **fourth** chapter we provide the methodology outline of this dissertation

In the **fifth** chapter we present the first part of our methodology which relies on the generation of synthetic water demand timeseries, the simulation of cyber-physical attacks and ultimately the creation of a new set of datasets aiming to capture the presence of uncertainty in water distribution networks.

In the **sixth** chapter we apply three different machine learning algorithms to detect the attacks contained in the available datasets and report their performance.

In the **seventh** chapter we present the major conclusions drawn from the models' performance

LITERATURE REVIEW

1. LITERATURE REVIEW

1.1 Stochastic methods for water demand estimation

Creating a model for water distribution systems that is consistent between the observed data of the real network and the simulated data from a hydraulic analysis model, is key into creating robust anomaly detection algorithms. One of the most important input components in a WDS model is water demands, but their estimation is usually complicated due to the stochastic behavior of water consumption. To represent the random nature of water requirements, a common practice is the use of models that generate water demands stochastically.

Kossieris et al. in their paper [2] describe the most essential stochastic methods for water demand modelling and one of the most widespread ones involves the use of pulse-based models. Based on the assumption that residential demand (or the demand of household water appliances) can be described by a rectangular pulse, Poisson rectangular pulse (PRP) [9]–[11] and Poisson-cluster processes [12]–[14] have been used to generate synthetic water demands at fine temporal and spatial scale. Then synthetic demand records can be obtained by aggregating the pulses of those fine resolution data.

Apart from pulse-based methods, Gargano [15] proposed a method for the probabilistic representation of the daily trend of residential water demand for different number of users, using a mixed-type distribution to describe the whole process. Furthermore, Alvisi et al. [12] using a bottom up-approach, employed random polynomial processes along with reordering techniques to enable the generation of synthetic water demand data which are statistically consistent (in terms of mean, variance and spatio-temporal correlations) with the observed time series at lower and higher spatial and temporal scales.

Finally, Kossieris et al. [13], [16] proposed a method based on Nataf-type simulation models [17], [18] that combines the widely used class of linear stochastic models (e.g. autoregressive models) with the concept of Nataf's joint distribution model to enable the explicit reproduction of the marginal distribution and the dependence structure of the process. One of the advantages of this methodology is that it allows the accounting of important marginal properties such as tail behavior and hence the reproduction of extremes. This is very useful in the design of WDS that require characterization of peak flows at different temporal resolutions.

The benefit of realistic reproduction of extremes can also be transferred in anomaly detection problems, where creating a model that is able to distinguish normal peaks from outliers is essential. Hence, in this dissertation the Kossieris et al. [2] method is chosen to stochastically generate water demands.

1.2 Cyber-physical attacks on Water Distribution Systems

Before creating effective protection methods against cyber-physical attacks on Water Distribution Networks, it is important to understand the nature of CPAs and the network's response to them. For that reason, there is an emerging scientific interest in developing tools to assess the effect of cyber-physical threats on the hydraulic behavior of water distribution systems.

Perelman et al. [19] first presented an approach to assess the vulnerability of small scale water networks, while Adepu et al. [20] investigated cyber-physical attacks in the context of a laboratory testbed to obtain the response of an operational water distribution system.

Taormina et al. [21] approached the problem by creating simulation-based tool, named epanetCPA, to assess the risks associated to CPAs. The authors presented a modeling framework consisting of two main components, namely an attack model that characterizes a broad range of attacks on cyber components (e.g., sensors, PLCs, and SCADA) and a MATLAB toolbox (epanetCPA) that automatically implements in EPANET all attacks based on the attack model.

Nikolopoulos et al. [22] introduced a Python-based modeling platform for stress-testing WDNs under CPAs aiming to aid risk management practices. This modeling platform, named RISKNOUGHT, by incorporating the interconnection between cyber and physical processes, allows to simulate WDS's response in a higher fidelity and a more realistic way than simulation solutions that mostly focus on the outcome of a cyber operation and the state of cyber-component.

1.3 Detecting cyber-physical attacks on Water Distribution Systems

The area of anomaly detection and intrusion detection in Industrial Control Systems has been widely studied. When it comes to the water distribution sector, a recent example is the Battle of the Attack Detection Algorithms (BATADAL), an international competition on water distribution system cyber-attack detection [7]. In that competition seven teams demonstrated their solutions on a simulated dataset. The best results were shown by the authors of [23] who proposed a model-based approach that employed EPANET to simulate the hydraulic processed of the water distribution systems, and then used the error between the EPANET simulated values and the available observations to detect anomalous behavior. The limitation of this approach is that in real world problems creating a precise system model is hindered by various factors, such as the inherent variability of demand patterns and the uncertainties of the hydraulic model.

Another team that achieved a high score in the competition is Abokifa et al. [24] that introduced a three-stage detection method with each stage targeting at a different type of anomaly. More specifically, after checking for violations in any of the actuator rules, they used statistical fences to detect simple outliers, ANNs for contextual anomalies and finally they detected global anomalies via PCA decomposition. A similar approach took also Giacomoni et al. [25] who also verified

the integrity of the actuator rules and SCADA data and separated data into normal and anomalous by performing principal component analysis (PCA).

Moreover Brentan et al. [26] used Recurrent Neural Networks to forecast tank water levels and by comparing their predictions with the available observations, detected when the system was under attack. Chandy et al. [27] approached the problem with a combination of control rules verification and the use of a Convolutional Variational Autoencoder that calculated the reconstruction probability of the data: the lower the probability, the higher the chance of the data being anomalous. Pasha et al. 's method [28] involved three interconnected modules. These modules focused on consistency checks, pattern recognition, and hydraulic and system relationships. Lastly, Aghashahi et al. [29] implemented a two-stage method which included dimensionality reduction from the multidimensional observed time series data to a four-dimensional feature vector that was then passed to a classifier to detect attacks (Random Forest).

After the BATADAL competition finished, the datasets provided are still publicly available and are one of the few cases of open-source datasets available for the research in cyber security of water networks. Those datasets gave a springboard to more researchers to study cyber-physical attacks on WDNs. A few examples include Taormina et al. [4] who used an Autoencoder (AE) and Kravchik et al. [30] that experimented also with an AE and 1D Convolutional Neural Networks (CNNs), PCA-Reconstruction and frequency domain analysis. Moreover, Ramotsoela et al. [31] proposed an ensemble technique that focuses mostly on traditional Machine Learning algorithms for anomaly detection by using both density-based and Quadratic Discriminant Analysis (QDA). Kadosh et al. [3] examined the use of support vector data description (SVDD) method along with a feature selection methodology.

Finally, in this work some of the key methodologies in cyber-physical attack detection in WDS have been mentioned. The methodologies mentioned are a good starting point to gain insight into the problem of CPA detection and although there are various other approaches that have been proposed for that particular matter, conducting a more rigorous survey is out of the scope of this dissertation.

1.4 Graph Neural Networks

Recently, there is an increasing number of applications regarding data generated from non-Euclidean domains that can be represented as graphs with complex interconnections and dependencies between its components. Although deep learning has been very successful in capturing Euclidean data, the complexity of graphs has imposed challenges on the existing deep learning algorithms. As a result, new definitions and operations that generalize the existing deep learning methods (such as CNNs, RNNs and Autoencoders) have been rapidly developed to handle the complexity of graph data. These new generalizations are the base of a new type of deep learning method, that operates on relational data i.e. graphs, named Graph Neural Networks (GNNs).

There are numerous different types of architectures of Graph Neural Networks most of them cited in [32]. In this dissertation we only mention the GNN architecture we found most relevant to our topic i.e. GNN architectures applicable on spatio-temporal graph data.

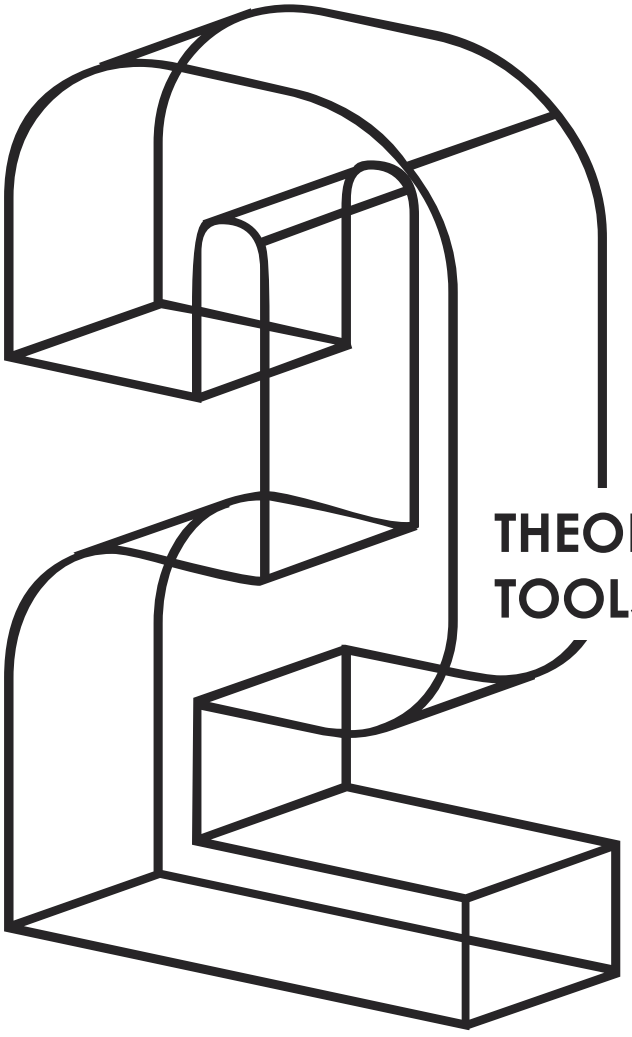
This category is the CNN-based, Spatio-Temporal Graph Neural Network (STGNN). STGNNs model graphs that are dynamic in terms of their node inputs, while assuming interdependency between connected nodes. STGNNs have already been implemented successfully in many problems including traffic prediction, human skeleton movement prediction, and human brain networks.

Accurate traffic forecasting is essential in a smart transportation system. Since the traffic condition of one road depends on its adjacent road's conditions, it is critical to incorporate spatial dependency when performing traffic speed forecasting. STGNNs allow to capture both the spatial and temporal dependencies of a graph simultaneously, by considering the traffic network as a spatial-temporal graph, where the nodes are sensors installed on roads, the edges are measured by the distance between pairs of nodes, and each node has the average traffic speed within a window as dynamic input-features. Yu et al. in their paper [33] propose a type of Spatio-Temporal Convolutional Networks for traffic forecasting and evaluate their model on real-world traffic datasets with promising results.

In human kinematics, capturing the motion of a human is not straightforward as it is subject to the constraints of the human body. Teh et al. in their paper [5] model human hand motion by representing as a graph the human joints which are linked by skeletons. Modelling the human hand as a graph allowed them to predict its motion and led them to propose a Structural Convolutional Neural Network (SCNN) architecture for time series data with arbitrary graph structure.

Last but not least, Covert et al. [34] proposed the Temporal Graph Convolutional Network (TGCN) to automatically detect seizures from electroencephalograms (EEGs). By modelling the electrodes placed on a patient's scalp as a graph, the researchers used TGCNs to detect from EEGs when precisely the seizures occur and the parts of the brain that are most involved.

Water Distribution Networks have an inherent spatio-temporal structure and from the above it is evident that STGNNs seem like a potentially useful tool to model them with. As far as we know, to this day Graph Neural Networks have not been applied yet in cyber-physical attack detection on WDS.



THEORETICAL TOOLS

2. THEORETICAL TOOLS

2.1 Types of cyber-physical attacks on Water Distribution Systems

A cyber-physical system such as a water distribution network has three basic security goals: operational integrity, availability and confidentiality. Each security goal can be targeted by a specific type of cyber-physical attack. Operational integrity, which means that the system components are able to function as intended and provide a barrier between the water in the system and external threats, can be compromised with *deception* attacks that alter the information sent or received by sensors, actuators or controllers. Availability denotes that the system is ready for use upon demand and it can be threatened by *Denial of Service* attacks (DoS) which occur when an attacker renders the system unavailable. Finally, confidentiality, which relates to keeping sensitive information safe from unauthorized users, is susceptible to eavesdropping attacks. *Eavesdropping* attacks are essentially the act of stealthily accessing sensitive information of a WDS such as the system's state and behavior [21].

According to [21], cyber-physical attacks on WDS can also be classified on the basis of the element being attacked i.e. the *target*. The elements that could potentially be under attack in a WDS are sensors, actuators, PLCs and SCADA, as well as the communication links connecting them.

As their name implies attacks can also be either *cyber* or *physical*. Sensors and actuators are only susceptible to physical attacks since the attacker needs to have direct physical access to the target to damage, manipulate or replace it. Attacks on the connection between different elements of the network can as well be physical, if the connection is hardwired. Although a physical attack might be unlikely it should be considered when securing a WDS in case an actuator or sensor is in a remote (or poorly monitored) area that might be accessible to an attacker.

Attacks on the connection link between different elements of the network, when the connection is wireless are considered to be cyber-attacks. All attacks no matter the target or whether they are cyber or physical, could be deception, denial of service or eavesdropping attacks.

To elaborate, let's assume a cyber-attack targeted to the connection link between two PLCs. If one PLC monitors the water level from a tank and transmits it to the other PLC which controls a pump on the basis of the tank's water level, then when the connection is interrupted and the content shared between them is manipulated (deception), a disruption on the normal pumping operations is caused. The attacker may also eavesdrop the communication or prevent one of the PLCs from sending/receiving the sensor reading by flooding the communication channel with traffic (denial of service).

Another example, is a cyber-attack targeted to the connecting link between PLC and SCADA. Again, the communication can be manipulated, eavesdropped or

temporarily interrupted by flooding the communication channel. As a result, incomplete or altered information reaches the SCADA. What is important is that the adversary might resort to this attack to conceal other actions from human operators or event detection algorithms implemented at SCADA level.

Attack concealment is the key challenge attack detection algorithms face. A good CPA detection algorithm is not one that identifies outliers per se, but one that is able to uncover contextual anomalies. Contextual anomalies are suspicious observations that even if their magnitude is well within the previous historic bounds, are anomalous within a specific temporal context based on their previous observations.

The figure below is an extract from [21] and illustrates the different types of cyber-physical attacks on a simple water distribution system consisting of one pump, one valve, one tank and a few demand nodes.

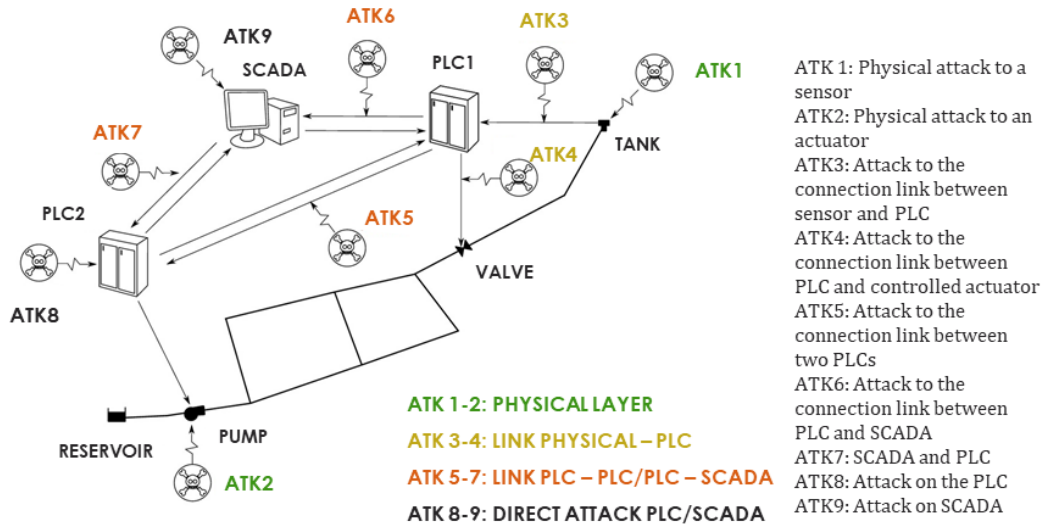


Figure 1: Graphical representation of types of cyber-physical attacks on a water distribution system

2.2 A common anomaly detection approach

We assume there is a dataset X, that its distribution contains normal and anomalous data:

$$p_{full}(x,y) \sim p(y=1)p(x|y=1) + p(y=0)p(x|y=0)$$
$$p_{normal}(x) \sim p(x|y=0)$$
$$p_{abnormal}(x) \sim p(x|y=1)$$

In anomaly detection our goal is to make the best possible estimation of the $p_{normal}(x)$ and $p_{abnormal}(x)$ distributions.

When it comes to the problem of detecting attacks in a Water Distribution System, we only have available a small portion of the entire (anomalous) set of cyber-physical attacks that could threaten it- it is not plausible to identify and model all possible attack scenarios. Thus, the most effective way of detecting anomalies on a WDS is by utilizing semi-supervised algorithms.

Semi-supervised learning is when the training set available contains only normal points and the task is to identify the anomalous points in a test set. This is also called novelty detection:

$$D_{train} = X_{train} \sim p_{normal}(x)$$
$$D_{test} = X_{test} \sim p_{full}(x)$$

After using D_{train} to model the distribution of normal data, the algorithm is able to assign to each data point an anomaly score. It is expected that most observations will get low scores while the anomalous hopefully will have higher anomaly score. To make a final decision whether a point is anomalous or not, the detection algorithm needs a score threshold. The threshold will separate the normal data from the anomalous and it must be determined how high the anomaly score should be for the data to be considered anomalous. For instance, if we train an algorithm to forecast the SCADA readings of a WDN at a time t and we calculate the error between the observed readings and the predicted ones, how large should be the error in order to issue an attack alarm?

Determining a threshold is one of the key challenges in anomaly detection. A threshold score that is too low might catch most anomalies, but might also lead to a high rate of false detections. Too many false detections become a destruction, waste time and are overwhelming. The person responsible for dealing with potential attacks might also become habituated to the alarm raising the danger that they will not respond appropriately to a true attack -it's a case of the danger of "crying wolf". On the other side setting a threshold that is too high although it will decrease the number of false detections, it might also miss attacks. However, missing CPAs on a WDN, might have enormous long-term consequences. As a result, the decision is a trade-off between true and false detection.

With a few positive samples (attacks), like in the case of semi-supervised learning, and the appropriate classification metric, it is possible to find a satisfactory threshold. In the following section we present in detail the most common classification metrics.

2.2.1 Anomaly Detection Metrics

To assess how well a model performs and to compare its performance with other models it is important to determine an evaluation metric. In binary classification problems, such as anomaly detection where the goal is to define whether a system is under attack or not, there are numerous metrics available to use.

Choosing a single-number metric speeds up our ability to make a decision when selecting among a large number of models and to fine-tune the anomaly detection threshold. It gives a clear preference ranking and therefore a clear direction progress. However, each evaluation metric gives us a different perspective of the algorithm's performance, so the choice of a single-number metric is not straightforward and it must be aligned with the predefined research objectives.

Below, we present some basic classification metrics along with some of the limitations each one faces.

Confusion Matrix

In binary classification problems, a confusion matrix is a 2×2 matrix that offers detailed information about a model's performance. A confusion matrix summarizes a model's performance on a specific dataset by depicting the correlation between the actual label and the model's classification i.e. False Negatives, True Negatives, False Positives and True Positives. More specifically, in a binary classification problem, like determining if a system is under attack or not, we define as:

- **True positive:** When the system is under attack and the model's prediction is also that the system is under attack.
- **True negative:** When the system is not under attack and the model's prediction is also that the system is not under attack.
- **False positive:** When the system is not under attack and the model's prediction is that the system is under attack.
- **False negative:** When the system is under attack and the model's prediction is that the system is not under attack.

Confusion matrices contain sufficient information to calculate a variety of performance metrics, like precision and recall (see Figure below).

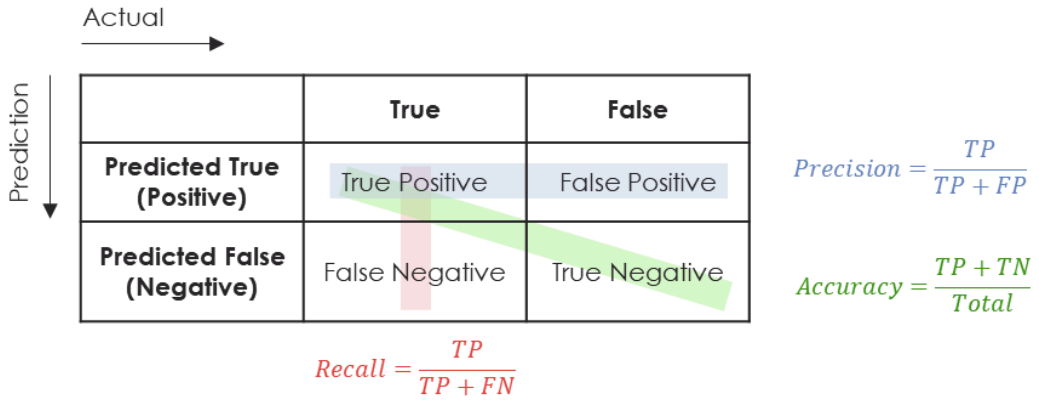


Figure 2: Confusion matrix, precision, accuracy and recall

Recall

Recall is a measurement that describes how many of the "true" predictions for all data points were actually "true". In other words, it measures the model's ability to correctly classify the state of the water distribution system as under attack. Recall is also known as sensitivity or true positive rate.

$$Recall = \frac{TP}{TP + FN}$$

Recall can be a deceiving metric in case an algorithm issues every instance as an attack. In that case the algorithm has detected every attack instance. There are no false negatives. Such model would return a recall score of 1.0, but contribute little.

Precision

Precision is a measurement that describes how many of the true predictions are actually true. In other words, it measures the model's ability to prevent false alarms.

$$Precision = \frac{TP}{TP + FP}$$

Precision is slightly harder to be deceiving. This can happen if an algorithm issues correctly only one alarm. Because no false positives are generated and the numerator is above zero, this gives maximum precision 1.0.

Accuracy

Accuracy is a measurement that describes how many predictions were correctly classified over the entire dataset.

$$Accuracy = \frac{TP + TN}{Total}$$

Accuracy can be a deceiving metric in imbalanced classification datasets. For example, in anomaly detection problems data points labeled as "under attack" (Positives) are significantly less than "under normal conditions" points (Negatives). As a result, even when failing to detect all Positive instances, i.e. not detecting any of the attacks, the accuracy score will still have a relatively high value.

From the above more metrics can be derived, such as the following:

Specificity

Specificity or True Negative Rate is another metric that determines the model's ability to avoid false alarms. It is similar to recall, but instead of the proportion of true positives to all of the true data points, it's the proportion of false positives to all of the false data points. In our case, it is the ration between the number of timesteps correctly classified as safe conditions and the total number of time steps during which the system is in safe conditions.

$$Specificity = \frac{TN}{TN + FP}$$

The drawback of specificity in imbalanced classification datasets is that it determines as more important True Negatives. True Negatives in anomaly detection problems are the majority class of the datasets, thus specificity presents very small variance and makes it hard to capture the differences in an algorithm's performance.

F-score

F-score is the weighted harmonic mean of precision and recall:

$$F_\beta = (1 + \beta^2)\frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

where β determines the balance between precision and recall, with high values favoring recall.

When $\beta = 1$ the F-score is called F1-score and it considers equally recall and precision. In other words, it takes into account both how well the model makes true predictions that are actually true (i.e. how many of the issued alarms were actually "under attack" labels) and how many of the total true predictions that model correctly predicted (i.e. how many "under-attack" labels out of the total were correctly detected).

The advantage of F-score is that, when using it, the focus is given to true positives, false positives and false negatives, while no attention is given to the majority class i.e. the true negative group.

The main disadvantage of F score is that one is unable to distinguish low recall from low-precision models when using solely F-score as an evaluation metric.

2.3 Machine Learning Algorithms

2.3.1 Feedforward Neural Networks

Suppose that (x,y) are points of a function f, where y=f(x). An Artificial Neural Network (ANN) is a deep learning algorithm that given some training examples (x,y) learns to approximate the function f.

The ANN's structure consists of neurons (depicted as nodes) that are organized into layers. A neuron's function is to receive inputs from n sources and then generate one output value. The neuron calculates the output by applying an activation function (nonlinear transformation) to a weighted sum of input values. Then it sends its output to m succeeding neurons.

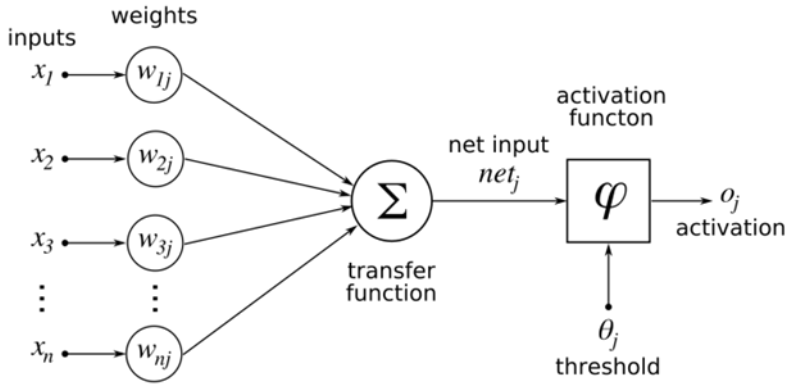


Figure 3: Structure of an artificial neuron

One example activation function is the ReLU (Rectified Linear Unit) function which is as follows:

If $net_j = \sum_{k=0}^{n} w_{kj} x_k + b$ is the weighted sum of the input and o_j is the output of the neuron, then

$$o_j = \begin{cases} net_j, & net_j \geq \theta_j \\ 0\ , & net_j < \theta_j \end{cases}, where\ \theta_j = 0$$

There are many activation functions that can be used in a neuron (see Figure below) and they are essential in neural networks, as they introduce non-linear properties into the network. This way a neural network can understand more complex patterns and give more accurate results.

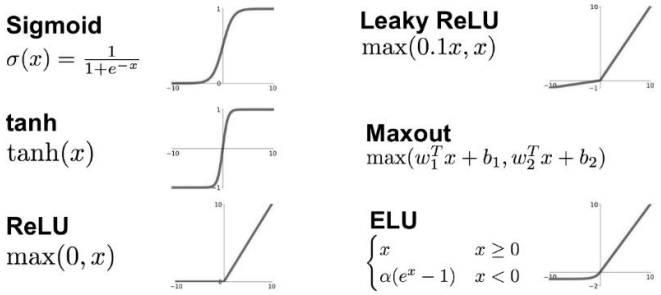


Figure 4: Types of commonly used activation functions in deep learning.

A set of neurons forms a layer and usually in an ANN there are three kinds of layers: the input, hidden and output layer. The input layer is the first layer of the network and the one that receives the input data (x). The output layer is the one that receives the ANN's predictions (outputs or $\hat{y}$) and the hidden layer is a layer between the input and the output. A hidden layer typically contains an activation function for training. When a neural network has more than one hidden layers it is called a deep neural network.

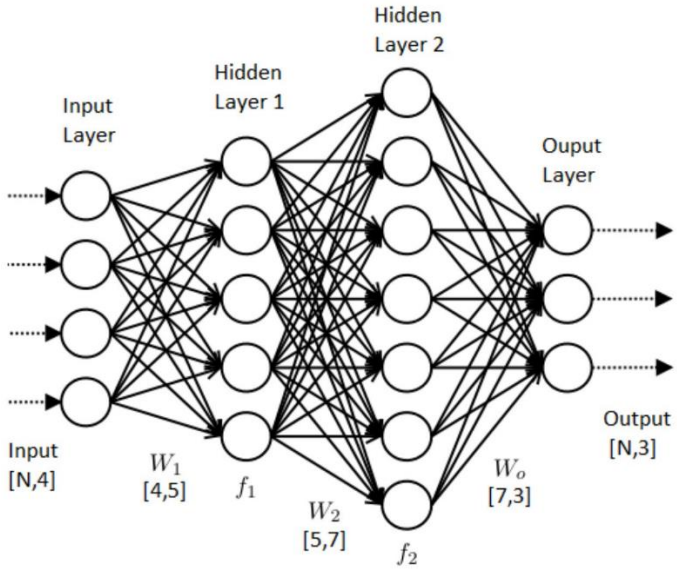


Figure 5: Typical architecture of an artificial neural network

As a result, information x flows through the layers of an ANN, until it reaches radically transformed the last layer which it outputs a prediction $\hat{y}$. The goal of an ANN is to adjust its parameters i.e. the weights of the neurons, until its prediction $\hat{y}$ is as close as possible to the objective y. Neural networks with different architectures can be designed, i.e. width (number of neurons in a layer) and depth (number of hidden layers) etc., until complex non-linear functions are learned.

To measure the performance of a neural network a loss/cost function is defined. The loss function usually depicts the error of the predicted value. Depending on the nature of the problem a different cost function can be chosen (mean squared error, mean absolute error, cross entropy loss). During training, a neural network

determines through an iterative process the ideal weights for each input feature that minimize the cost function.

The algorithm used to minimize the cost function is called gradient descent. In gradient descent the model's parameters are adjusted iteratively until finding the ones that minimize the loss. The gradients of loss are calculated with a process called backpropagation. In backpropagation, first the output values of each node are calculated in a forward pass and then the partial derivative of the error with respect to each parameter is calculated in a backward pass through the graph.

2.3.2 The Convolution Operation

The convolution operation is a two-step mathematical operation between an input matrix and a convolutional filter/kernel. The convolution involves the following:

1. Element wise multiplication of the kernel and a slice of the input matrix.
2. Summation of all the values in the resulting product matrix

The output is called an activation map and has the same shape as the convolution filter. The activation map consists of the results of the convolutional operations.

For example, given a 5 × 5 input matrix and a 3 × 3 kernel:

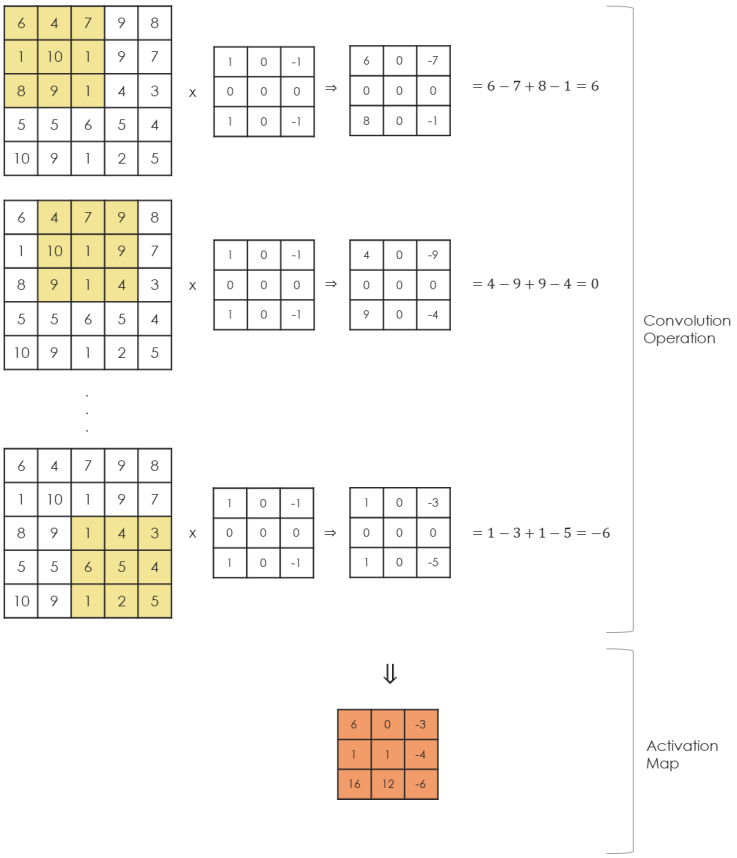
6	4	7	9	8
1	10	1	9	7
8	9	1	4	3
5	5	6	5	4
10	9	1	2	5

Input matrix

1	0	-1
0	0	0
1	0	-1

Kernel

We perform each convolution operation between a 3 × 3 slice of the input matrix and the kernel



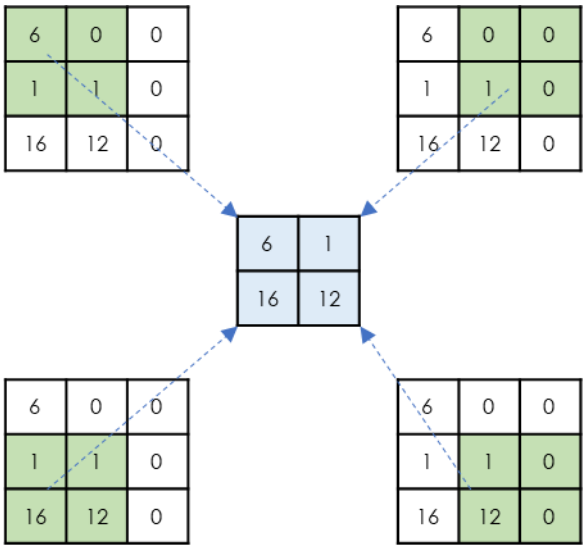
From the above, a convolution is a specialized kind of linear operation and the kernel has the same rank as the input matrix. For instance, if the input matrix is a 28×28 matrix, then the kernel can be any 2d matrix with a shape smaller than 28×28.

2.3.3 Convolutional Neural Network

Convolutional Neural Networks are neural networks that use the convolution operation in place of general matrix multiplication (see "Feedforward Neural Networks") in at least one of their layers. A typical convolutional layer has three components:

1. *Convolution stage:* This layer performs several convolution operations to produce the activation maps.
2. *Detector stage:* An activation function is applied to each element of each activation map.
3. *Pooling stage:* The pooling function is used to modify the output of the convolutional layer further. Pooling, like convolution, divides the matrix into slices and usually keeps either the maximum or average value across the pooled area. As a result, the output matrix is reduced to a smaller matrix.

For example, suppose the pooling operation divides the convolutional matrix into 2x2 slices with a 1x1 stride. As the following diagram illustrates, four pooling operations take place. Imagine that each pooling operation picks the maximum value of the four in that slice:



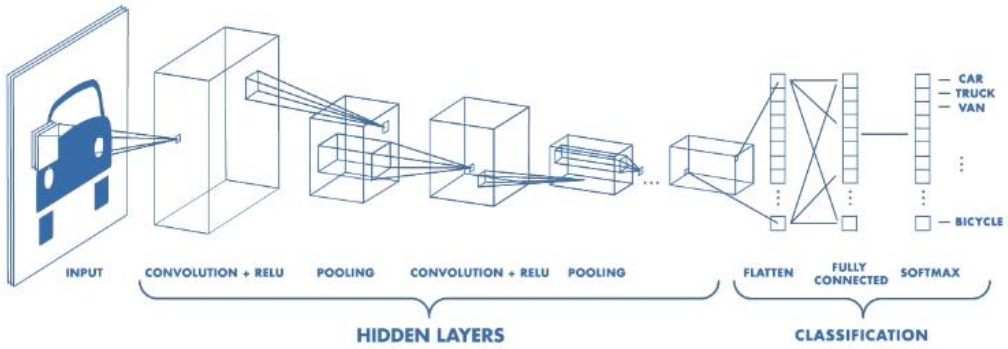


Figure 6: Typical architecture of a Convolutional Neural Network

To further elaborate, in the Figure above it is depicted a typical convolutional neural network. The input layer is an image matrix followed by two convolutional layers (Convolution-ReLU-Pooling). Their output is then reshaped to flatten out its spatial dimensions resulting in a 1D matrix. That matrix is then used as an input to an ordinary feedforward neural network.

The convolutional kernels' weights are parameters of the CNN and are obtained via training. Without convolutions a neural network would have to learn a weight for each input unit. With convolutional filters the algorithm has to learn only the weights for each filter, meaning that fewer parameters are stored and the memory requirements to train the model are reduced.

The use of the convolution operation is a characteristic of CNNs that allows them to "leverage three important ideas that improve a machine learning system: sparse interactions, parameter sharing and equivariant representations"[35]. These characteristics have made CNNs very successful in processing data with grid-like topography, like time series data (they can be considered as one-dimensional grid data), and images (two-dimensional grids of pixels).

2.3.4 Introduction to graphs and adjacency matrices

A graph is data structure for representing relationships. It comprises of nodes (or vertices) and edges connected together to represent relational information. Formally, a graph G is a tuple consisting of a finite set of $V(G)$ of vertices and a finite set $E(G)$ of edges, where each edge is an unordered pair of vertices. An edge between two nodes u and v is often denoted as (u,v).

For example, the graph illustrated below comprises of 5 nodes and 6 edges. A graph is called a directed graph if each edge is associated with a direction (see Figure 7 below). A directed edge can be considered as a one-way street. On the other side, an undirected graph is a graph that the connection order doesn't matter and it can be thought as a graph where each edge is directed.

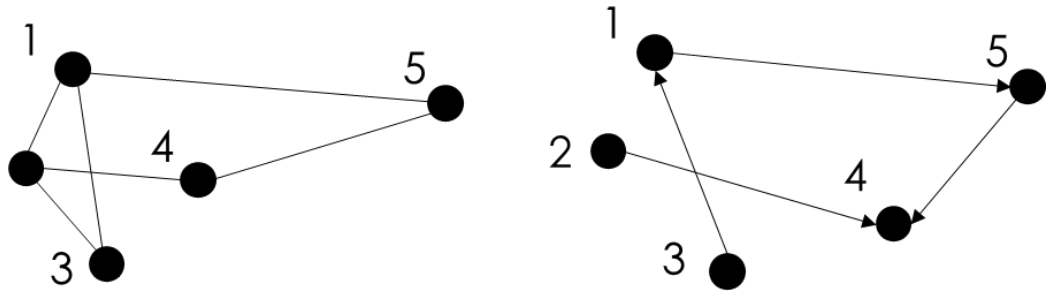
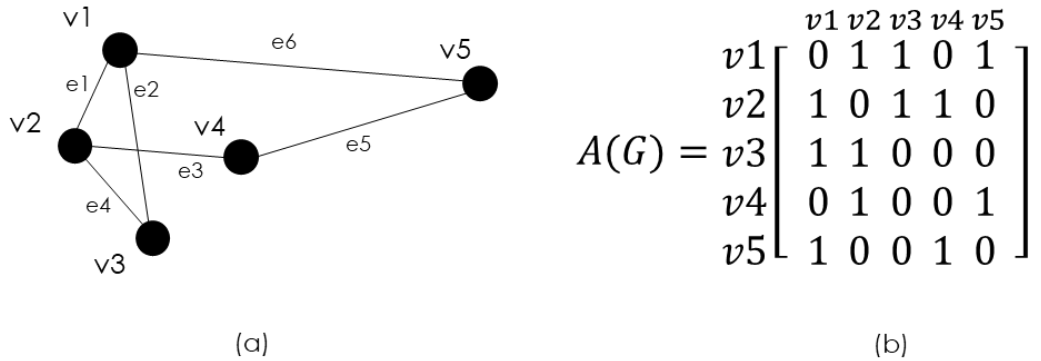


Figure 7: (a) Undirected graph and (b) directed graph

Now, let us suppose that G is a graph with the vertex set $V = \{v_1, v_2, \dots, v_n\}$ and the edge set $E = \{e_1, e_2, \dots, e_m\}$. The adjacency matrix $A(G)$ of G is an $n \times n$ matrix $A(G) = [a_{ij}]$ in which a_{ij} indicates the number of edges joining two vertices v_i and v_j. The Figure below illustrates a graph (a) and its adjacency matrix (b).



$$A(G) = \begin{array}{c} \\ v1 \\ v2 \\ v3 \\ v4 \\ v5 \end{array} \begin{array}{c} \begin{array}{ccccc} v1 & v2 & v3 & v4 & v5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 8: (a) A graph G with 5 nodes and 6 edges and (b) the adjacency matrix of graph G

2.3.5 Structural Convolutional Neural Networks

Teh et al. have proposed the Structural Convolutional Neural Network (SCNN) for general graph-structured CNNs for time series analysis [5]. In their paper they describe their model as a "neural network architecture that defines and uses specialized convolutional kernel with an arbitrary definable adjacency matrix."

First are defined the following for a graph with F number of nodes and the adjacency matrix $\vec{A} \in \mathbb{R}^{F \times F}$:

$\vec{y}^{l-1} = \left[\vec{y}_1^{l-1}, \dots, \vec{y}_F^{l-1}\right]^T$, previous layer's output

$\vec{y}^{l} = \left[\vec{y}_1^{l}, \dots, \vec{y}_F^{l}\right]^T$, current layer's output

$\vec{W}^{l} = \left[\vec{W}_1^{\,l}, \dots, \vec{W}_F^{\,l}\right]^T$, current layer's weights

$\vec{b}^{l} = \left[\vec{b}_1^{l}, \dots, \vec{b}_F^{l}\right]^T$, current layer's biases

where

$$\vec{y}^{-1} \in \mathbb{R}^{T \times F \times N},$$
$$\vec{y}^{l} \in \mathbb{R}^{T-(t-1) \times F \times M},$$
$$\vec{W}^{l} \in \mathbb{R}^{F \times t \times F \times N \times M},$$
$$\vec{b}^{l} \in \mathbb{R}^{F \times M},$$
$$\vec{y}_i^{l-1} \in \mathbb{R}^{T \times 1 \times N}, \forall\, i = 1, \dots, F,$$
$$\vec{y}_i^{l} \in \mathbb{R}^{(T-(t-1)) \times 1 \times N}, \forall\, i = 1, \dots, F,$$
$$\vec{W}_i^{l} \in \mathbb{R}^{t \times F \times N \times M}, \forall\, i = 1, \dots, F,$$
$$\vec{b}_i^{l} \in \mathbb{R}^{1 \times M}, \forall\, i = 1, \dots, F.$$

The kernel is made up of F sub-kernels and each of the sub-kernels i, which corresponds to node I, has weights W_i^l with the dimension of $t \times F \times N \times M$. The sub-kernels are slid across the temporal dimension of the input producing an output of $\left(T-(t-1)\right) \times 1 \times M$ for each node i. The output is then passed through an activation function g to produce:

$$\vec{y}_i^{l} = g\left(\vec{W}_i^{l} * \vec{y}^{l-1} + \vec{b}_i^{l}\right)$$

$$\vec{W}_i^{l} = \begin{bmatrix} \vec{w}_{i1}^{l} \\ \cdot \\ \cdot \\ \cdot \\ \vec{w}_{iF}^{l} \end{bmatrix}$$

where $*$ is the convolution operation,

$$\vec{W}_i^{l} * \vec{y}^{l-1} = \sum_{j=1}^{F} \vec{w}_{ij}^{l} * \vec{y}_j^{l-1}$$

and $\vec{w}_{ij}^{l}$ is the sub-kernel weights for the i node with its j neighbor.

2.3.6 Temporal Graph Convolutional Networks (TGCN)

Covert et al. proposed in their paper [34] a model (TGCN) that takes into account "the graph topology of a structural time series and applies feature extractors that are localized and shared over both the temporal and spatial dimensions of the input".

Inspired by Graph neural networks (GNNs) that use neighborhood aggregation schemes, their model is based on a proposed spatio-temporal convolutional (STC) layer described below:

Suppose that the input layer l is $h^{l-1} \in \mathbb{R}^{T^{l-1} \times p \times c^{l-1}}$, where T^{l-1} is the number of time points at the previous layer, and $h_i^{l-1} \in \mathbb{R}^{T^{l-1} \times c^{l-1}}$, represents the hidden features associated with sequence i. The STC layer can be used with two different propagation rules.

Both rules begin by applying a 1D convolution (denoted by $*$) with filter W_{int}^l to each sequence of hidden features h_i^{l-1}, resulting in an intermediate set of features denoted by a_i^l. Note that filter W_{int}^l is shared across all sequences in the layer l. The two rules differ in how they handle the aggregation of features from a node and its neighbors.

Rule A aggregates features from the node's neighborhood including the node itself, and then applies nonlinearity g. The aggregation operation (e.g. mean, max) is performed along the spatial dimension, so that the temporal and channel dimension are retained.

$$\boldsymbol{Rule\ A}$$

$$\begin{aligned} a_i^l &= W_{int}^l * h_i^{l-1} \\ z_i^l &= AGGREGATE\left(\{a_j^l for\ j\ in\ N(i)\}\right) \\ h_i^l &= g\left(z_i^l\right) \end{aligned}$$

Rule B first aggregates features across a node's neighbors, and then combines these features with the node's own features by concatenating them and passing them through an additional nonlinear operation, parameterized by W_{comb}^l. This prevents the node's feature from being "diluted" by the features from its neighboring nodes.

$$\boldsymbol{Rule\ B}$$

$$\begin{aligned} a_i^l &= W_{int}^l * h_i^{l-1} \\ z_i^l &= AGGREGATE\left(\{a_j^l for\ j\ in\ N(i) \setminus i\}\right) \\ h_i^l &= g_2\left(W_{comb}^l * g_1\left([z_i^l, a_i^l]\right)\right) \end{aligned}$$

The neighborhood of node i is defines as $N(i) = \{j \ \ s.t. \ \ A_{ij} = 1\}$, i.e. it's the set of nodes that have and edge to i. The only parameter for rule A is the convolutional kernel $W_{comb}^l \in \mathbb{R}^{t_2^l \times c^l \times c^{l-1}}$. The two parameters for rule B are W_{int}^l, and the second convolutional kernel $W_{comb}^l \in \mathbb{R}^{t_2^l \times c^l \times (2 * c^l)}$. In rule A the hyperparameters are the choice of nonlinearity g, the temporal kernel size t^l, and the number of channels c^l. Rule B has the additional hyperparameter t_2^l, which could simply be set to 1 or t^l, as well the possibility of a second nonlinearity. For the two rules, note that filters are shared both spatially and temporally.

The adjacency matrix of the sequences is used when we aggregate features across $N(i)$. An interesting property of STC layers is the independence between the number of parameters and the input adjacency matrix, which allows a TGCN model to accept inputs wit arbitrary graph topologies.

In the paper discussed, the authors also propose incorporating information not only from nodes that are directly connected, but also from nearby nodes that are reachable within k steps. To obtain the k-step reachability matrix, they use the operation $A(k) = \mathbb{1}(A^k)$ where A^k is the adjacency matrix to the kth power, $\mathbb{1}(\cdot)$ is an element-wise indicator function. Setting $k > 1$ enables information to spread through the graph using fewer layers.

2.3.7 Autoencoders

An autoencoder is a neural network that is trained to attempt to copy its input to its output. Internally, it has a hidden layer h that describes a code used to represent the input. The network may be viewed as consisting of two parts: an encoder function $h = f(x)$ and a decoder that produces a reconstruction $r = g(h)$.

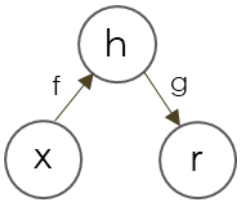


Figure 9: The general structure of an Autoencoder

If an autoencoder succeeds in simply learning to set $g(f(x)) = x$ everywhere, then it is not especially useful. Instead, autoencoders are designed to be unable to learn to copy perfectly. Usually they are restricted in ways that allow them to copy only approximately, and to copy only input that resembled the training data. Because the model is forced to prioritize which aspects of the input should be copies, it often learns useful properties of the data.

Copying the input to the output may sound useless, but we are typically not interested in the output of the decoder. Instead, we hope that training the autoencoder to perform the input copying task will result in h taking on useful properties.

One way to obtain useful features from the autoencoder is to constrain h to have a smaller dimension than x. An autoencoder whose code dimension is less that the input dimension is called, undercomplete. Learning an undercomplete representation forces the autoencoder to capture the most salient features of the training data.

Unfortunately, if the encoder and decoder are allowed too much capacity, the autoencoder can learn to perform the copying task without extracting useful information about the distribution of the data.

Autoencoders can be used for a wide variety of applications. Commonly, they are used in problems like dimensionality reduction, data denoising, feature extractions and anomaly detection.

2.3.8 Support Vector Data Description Classifier

The Support Vector Data Description (SVDD) classifier is designed to create a tight spherical boundary around any numerical multidimensional data using Kernel functions. By calculating the relative location of a new observation to the spherical boundary, one can decide whether the observation belongs to the fitted data or is considered an outlier. The SVDD classifier requires two tuning parameters. The first is the C parameter. When $C \geq 1$, it means that no outliers are expected in the training dataset. Then, for $C \geq 1$, the SVDD solves the following optimization problem:

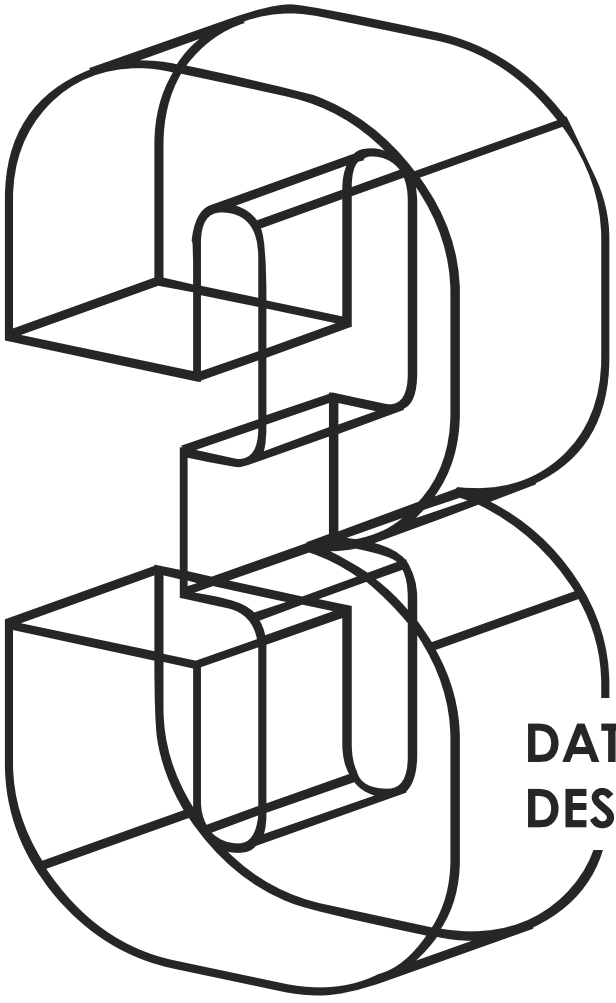
$$\min_{R,\alpha} R \; Subject\ to \; \|\varphi(x_i) - a\| \leq R$$

where R and a=decision variables; and $\varphi(x_i) =$ function mapping the data to a higher-dimensional space. Herein, $\varphi =$ radial basis function: $\sum_{i,j} \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right)$ for a pair of samples in the datasets x_i, x_j ; and $\gamma =$ second tuning parameter for the SVDD algorithm. In the prediction stage after the optimal decision variable R_{opt}, α_{opt} are determined, the decision value (DV) for any test instance x is given as detailed in the following equation:

$$DV = \left\|\varphi(x_i) - \alpha_{opt}\right\| - R_{opt}$$

where the boundary for the classification is obtained when $DV = 0$.

SVDD is useful for obtaining a geometric description of data and in most applications also for detecting outliers. SVDD is used in domains where most of the data are in one class. Applications of SVDD include equipment prognostics, cybersecurity and intrusion detection, fraud identification and others.



DATA DESCRIPTION

3. DATA DESCRIPTION

3.1 The network of C- Town

The network of C-Town was first introduced in the "Battle of the Water Calibration Networks"[36]. It is based on a real-world water distribution network and consists of one reservoir, seven cylindrical tanks (T1-T7), one actuated valve (V2), 429 pipes, 388 nodes, and 5 pumping stations (S1-S5) for a total of 11 pumps (PU1-PU11).

Control rules of the network are programmed into the Programmable Logic Controllers (PLCs). C-Town has nine PLCs that record, receive and send information about tanks, pumps and valves based on their control logic. Table 1 showcases the sensors and actuators controlled by the PLCs in C-Town. PLCs share with each other information related to their control rules and send their records to the Supervisory Control And Data Acquisition system (SCADA). SCADA's role is to coordinate the operations and store the readings provided by the nine PLCs. Since SCADA is a remote operating system, it is part of the cyber network of C-town, thus making the WDN susceptible to cyber-attacks.

Table 1: Sensors and actuators monitored/controlled by the PLCs in C-Town, Source:[7]

PLC	Sensor	Actuators (Controlling sensor)
PLC1	-	PU1 (T1), PU2(T1)
PLC2	T1	-
PLC3	T2	V2(T2), PU4(T3), PU5(T3), PU6(T4), PU7(T4)
PLC4	T3	-
PLC5	-	PU8(T5), PU9(-), PU10(T7), PU11(T7)
PLC6	T4	-
PLC7	T5	-
PLC8	T6	-
PLC9	T7	-

The hydraulic behavior of C-town can be simulated by using EPANET, an open source software that performs hydraulic analysis on WDSs. EPANET is a demand-driven analysis tool, and to perform simulations the water demand at each node of the network at each time step is required. The water demand of a node at each time step is calculated by multiplying the base demand of a node by its corresponding demand pattern.

Base demand is the average or nominal demand for water at each node, while the demand pattern is a time pattern used to characterize time variation in demand at each node. The demand pattern provides multipliers that are applied to the base demand to determine actual demand in a given time period. A network can have different base demands at each node and multiple demand patterns. C-Town is divided into five DMAs (District Metered Areas), each with its own demand pattern.

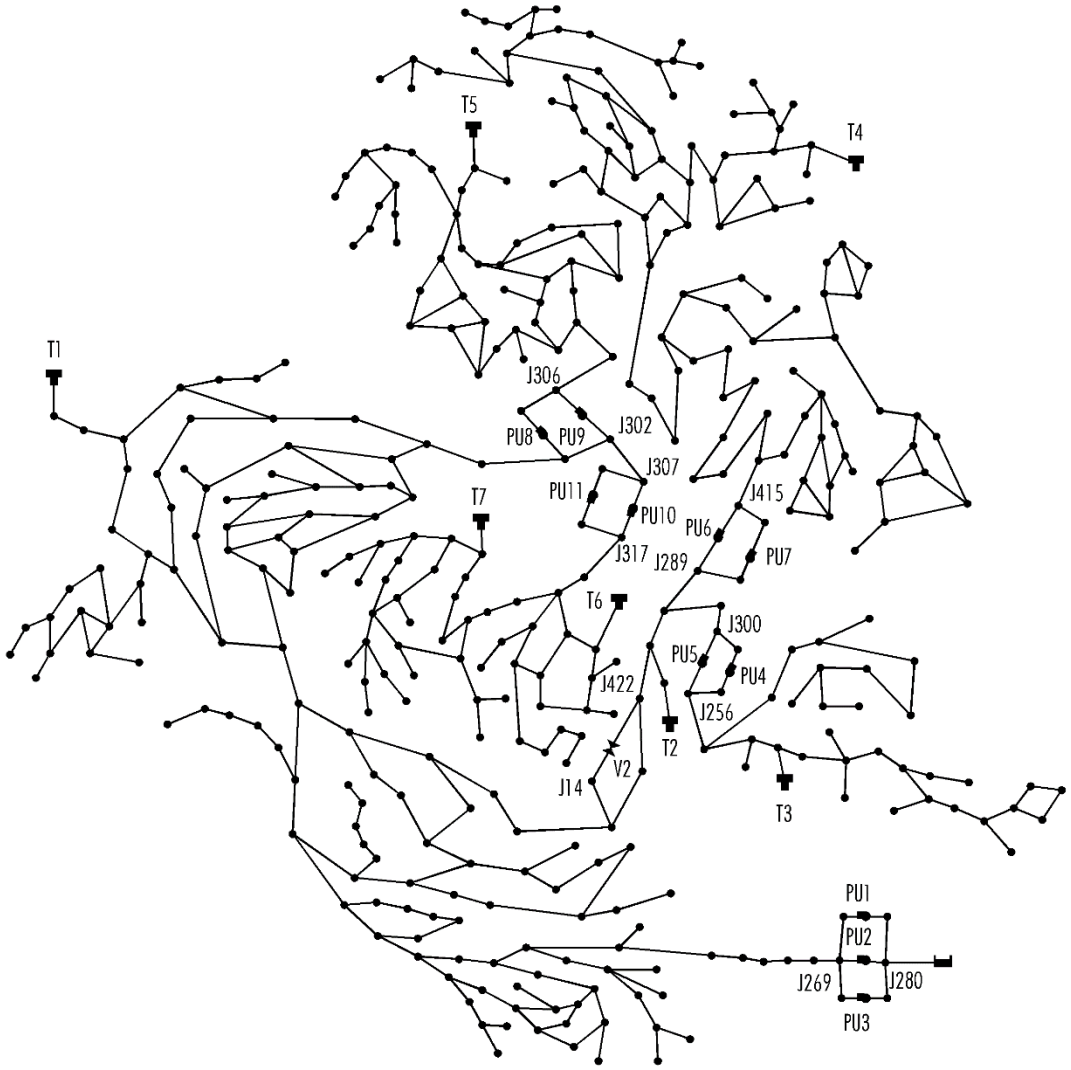


Figure 10: The network of C-Town

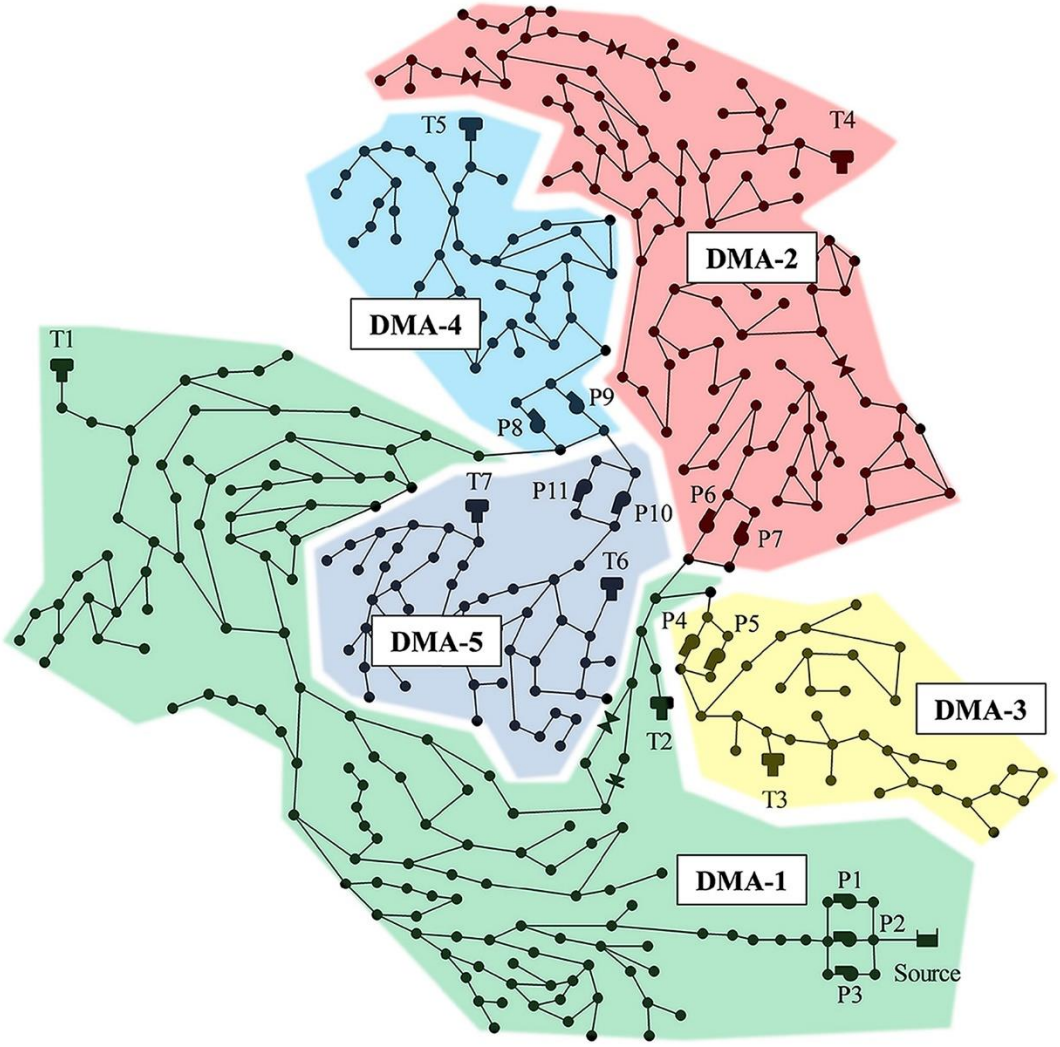


Figure 11: District Metered Areas (DMAs) of C-Town

3.2 The BATADAL Competition (BATle of the Attack Detection ALgorithms)

C-Town was again featured as a benchmark network in the BATtle of the Attack Detection ALgorithms (BATADAL), an international competition on cyber security of water distribution systems held at the World Environmental and Water Resources Congress (Sacramento, California, May 21-25, 2017) [7]. One of the outcomes of this competition is the openly available resources for the study of CPAs on WDNs, which are the following:

-Three datasets with SCADA readings from three different simulations:

dataset03: this dataset contains readings from a one-year EPANET simulation. During this simulation the network was only under safe conditions.

dataset04: this dataset has SCADA readings with a span of 6 months, containing 7 attacks.

testdataset: a 3-month dataset with 7 different attacks.

All three datasets contain hourly SCADA readings from 43 variables of the network. These variables are:

- water level at all seven tanks of the network (T1-T7)
- status and flow of all 11 pumps (PU1-PU11) and the actuated valve (V2) of the network
- pressure at 24 pipes of the network (corresponding to the inlet and outlet pressure of the pumps and the actuated valve)

-A table describing each attack scenario featured in the datasets "dataset04" and "testdataset"

-An EPANET input file named "ctown.inp" containing information (topographical and hydrological) about the distribution system. Among the information in the C-Town EPANET input file are:

- Monthly water base demands given at each node of the network.
- One week of hourly demand patterns for each of the five DMAs.

The SCADA readings provided in the available datasets, have been simulated using demand patterns different from the ones included in the "ctown.inp" EPANET file and are not publicly available. Furthermore, the datasets containing attacks have been simulated using epanetCPA, a MATLAB toolbox that allows to design a variety of cyber-attacks and simulate with EPANET the hydraulic response of the WDN - [21],[37], [38].

3.3 The challenge of the competition

The BATADAL contestants were asked to create algorithms to detect all the attacks contained in the datasets presented above. The algorithms' performance was evaluated with a combination of metrics: time-to-detection and classification accuracy.

The first metric, time-to detection is the following:

$$S_{TTD} = 1 - \frac{1}{N_A}\sum_{i=1}^{N_A}\frac{TTD_i}{\Delta T_i}$$

where $N_A = number\ of\ attacks$, $\frac{TTD_i}{\Delta T_i}$ = the time to detection of attack i as a ratio of the total attack duration ΔT_i.

The second metric, classification performance is defined as:

$$S_{CM} = \frac{TPR + TNR}{2}$$

where $TPR = True\ Positive\ Rate$, i.e. the ratio of true positives (TP) to the sum of true positives (TP) and false negatives (FN) and $TNR = True\ Negative\ Rate$, i.e. the ratio of true negatives (TN) to the sum of true negatives (TN) and false positives (FP).

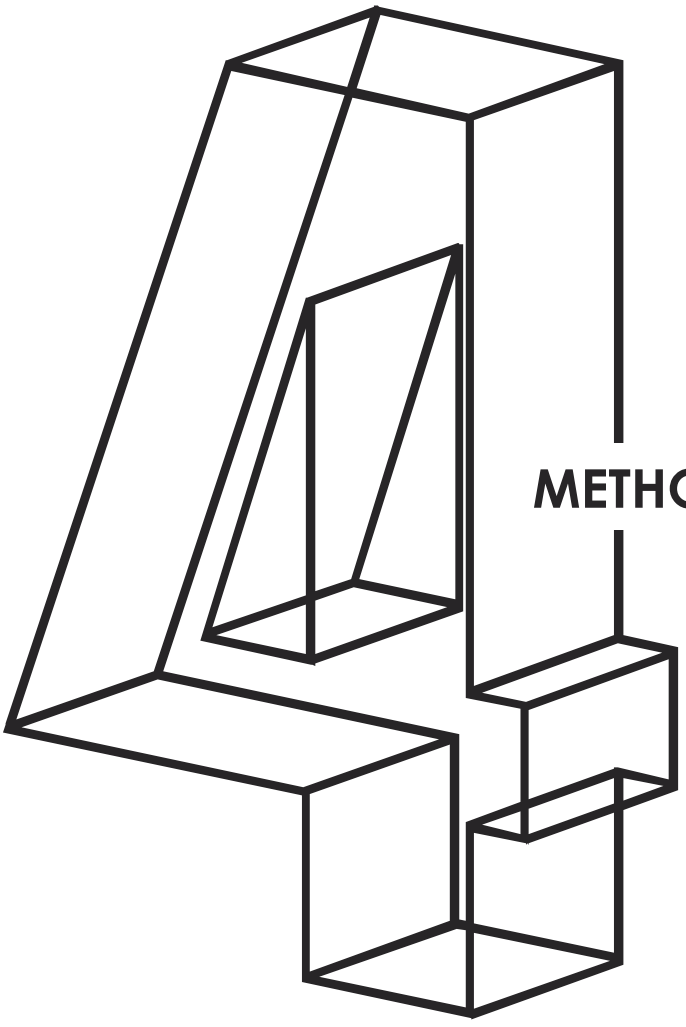
The two metrics are combined into a single score:

$$S = \gamma S_{TTD} + (1 - \gamma)S_{CM}$$

where $\gamma = 0.5$ is a coefficient that determines the relevant importance of the two evaluation metrics (here considered equally important).

A naïve algorithm that predicts the system to be always in safe conditions gets a score S equal to 0.25 ($S_{TTD} = 0, S_{CM} = 0.5$). On the other hand, flagging the system as always under attack yields a value of S equal to 0.75 ($S_{TTD} = 1, S_{CM} = 0.5$). This showcases the fact that S is biased towards attack identification, since the consequences of failing to disclose an attack are deemed more costly than issuing false alarms.

From the above, an algorithm with a score S larger that 0.75 means that it performed better than the naïve detection mechanisms.



METHODOLOGY

4. METHODOLOGY

4.1 Methodology Outline

The goal of this dissertation is to contribute to the research of cyber-physical attacks detection on simulation-based data on water distribution systems. We do that by:

(a) Generating new, more realistic datasets for the study of cyber-physical attack detection on water distribution networks. To do that, we perform simulations using stochastically generated demand patterns.
(b) Evaluating the performance of published algorithms for CPA detection on WDNs on our, more realistic datasets. We also compare the algorithms' performance on our datasets with their performance on naive, non-realistic datasets. Moreover, we also report the algorithms' performance on pre-existing, non-realistic datasets to evaluate each algorithm holistically.
(c) Exploring the use of Spatio-Temporal Graph Neural Networks on the CPA detection on WDNs problem.

To apply this methodology we build upon the simulated datasets featured on BATADAL, an international competition on cyber-security of water distribution networks [7].

The practical application of the methodology is divided into two parts:

The first part is about creating the new, more realistic datasets. To do that we:

A. Generate synthetic demand patterns using Kossieris et al. methodology [2]
B. Design attack scenarios similar to the ones featured in BATADAL [7].
C. Run simulations to obtain the hydraulic behavior of C-Town [38].

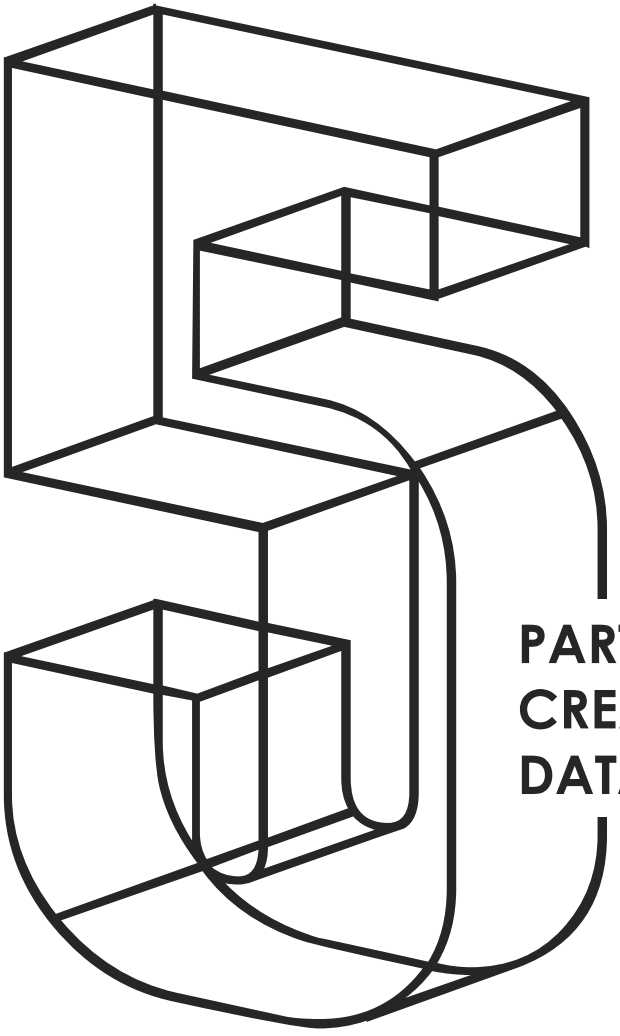
The second part is about detecting attacks using different machine learning algorithms. We use three different approaches:

A. Two published machine learning approaches that have been implemented successfully to the BATADAL datasets by Taormina et al. and Kadosh et al. The first applies an Autoencoder to detect the attacks [4], while the other an SVDD classifier [3].
B. A third approach that hasn't been implemented to a water distribution network's CPA detection problem yet. It is based on the works of Teh et al.[5] and Covert et al [34] on graph-structured time series data.

4.2 Software and Code Repositories

Most of our research relies on open source software. Synthetic water demands were generated using anySim[1], an R package developed by Tsoukalas and Kossieris [39]. The attack scenarios were simulated in epanetCPA[2], a MATLAB toolbox developed by Taormina et al. [7], [21]. The SVDD code is available for MATLAB in the published article [3]. The code for the Autoencoder[3] was developed by Taormina et al [4] using the Keras library. Finally, the code for the SCNN [5] and TGCN [34] layers was kindly provided by Ian Covert, PhD student at University of Washington. The code for our model, based on SCNNs, was written using the deep-learning framework PyTorch.

[1] https://github.com/itsoukal/anySim
[2] https://github.com/rtaormina/epanetCPA
[3] https://github.com/rtaormina/aeed



PART I: CREATING NEW DATASETS

5.PART I: CREATING NEW DATASETS

5.1 Generating synthetic demands

The synthetic demand patterns are generated using the methodology proposed by Kossieris et al. [2]. This stochastic modelling strategy builds upon Nataf's joint distribution model and was chosen because it preserves both the distributional and dependence properties of a process. Based on the available hourly one-week demand patterns of C-Town, synthetic demand patterns are generated by selecting an appropriate probability and autocorrelation function that resemble the marginal and dependence properties of demand patterns of BATADAL datasets. The generation of synthetic series was conducted via anySim an R package, developed by Tsoukalas and Kossieris, [39] which implements the aforementioned methodology.

To evaluate the performance of ML models with the use of stochastically generated demand patterns, we also need a non-stochastically generated dataset. Since the demand patterns that the BATADAL datasets originated from, are not publicly available, a new benchmark dataset was created: To create a naïve demand pattern with n-week duration, the one-week hourly demand patterns available from the ctown.inp file are repeated n-times and each of the hourly demands are multiplied with a random[4] number between 0.90 and 1.10. The resulting demand pattern is essentially n noisy versions of the available one-week demand pattern, concatenated into a single n-week demand pattern.

To be more precise, the goal is to create datasets corresponding to the ones available in BATADAL, meaning that we need to run 12, 6 and 3-month simulations on epanetCPA. Simulations with a specific duration require demand patterns of equal length, thus we also need 3 demand pattern datasets with a horizon of 12, 6 and 3 months respectively.

To gain more insight of the effect of stochastically generated demand patterns in the CPA detection algorithms, two different categories of synthetic demand patterns are generated. The distinguishing feature of each DP category is the marginal distribution from which it was generated. The first category of synthetic demand patterns is generated using the Beta probability distribution function, while the second category using the Gamma probability distribution function. Demand patterns simulated using the Beta distribution resemble a lot more the available one-week demand patterns, while demand patterns generated using the Gamma distribution function present larger fluctuations. Considering the non-stochastic dataset:

$3\ demand\ pattern\ categories * 3\ dataset\ durations\ (i.e. 12, 6\ and\ 3\ months) =$
$9\ different\ demand\ pattern\ datasets.$

The resulting demand pattern datasets are summarized in the table below:

[4] Using the RANDBETWEEN() formula in MS Office Excel

Table 2: Summarization of the generated demand pattern datasets

Name	Duration (months)	Generation Environment	Probability Distribution	Generation Method
dempat_b12	12	anySim	Beta	Stochastic
dempat_b06	6	anySim	Beta	Stochastic
dempat_b03	3	anySim	Beta	Stochastic
dempat_g12	12	anySim	Gamma	Stochastic
dempat_g06	6	anySim	Gamma	Stochastic
dempat_g03	3	anySim	Gamma	Stochastic
dempat_r12	12	MS Excel	Random[1]	Non-stochastic
dempat_r06	6	MS Excel	Random[1]	Non-stochastic
dempat_r03	3	MS Excel	Random[1]	Non-stochastic

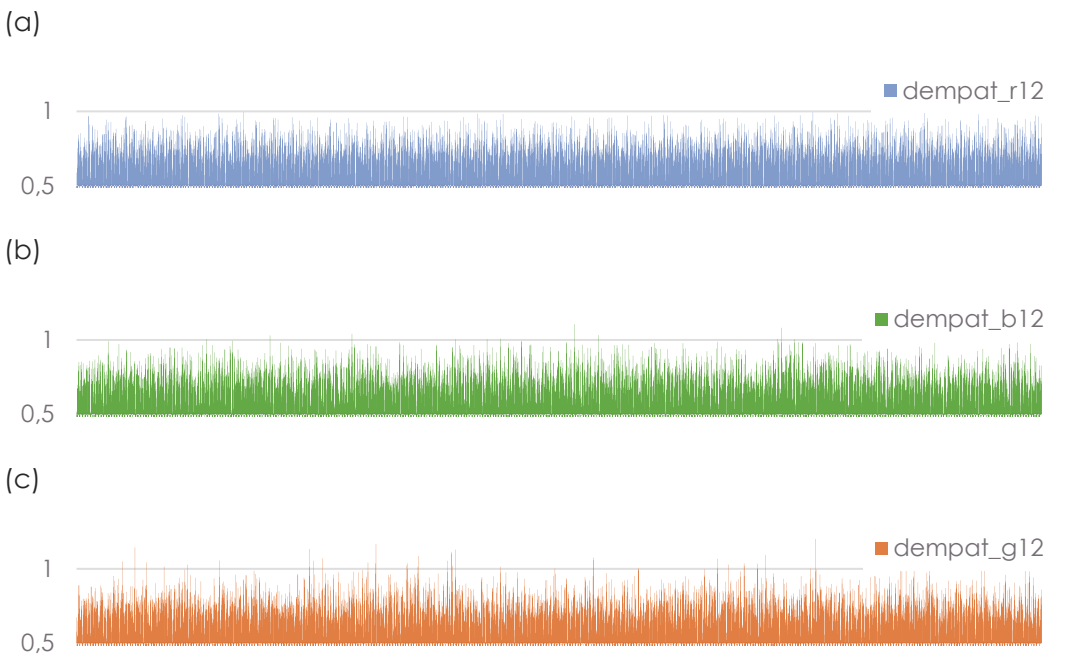


Figure 12: Hourly water demand variation in DMA_1 of C-Town during a year across the different demand pattern categories. (a) dempat_r12 – demand pattern generated non-stochastically, (b) dempat_b12 – demand pattern generated using the Beta distribution as the marginal distribution function, (c) dempat_g12 – demand pattern generated using the Gamma distribution function.

The charts above depict three demand pattern datasets from three different demand pattern categories. Note how the demand pattern that was generated non-stochastically has a fairly regular motif, while the stochastically generate demand patterns present larger fluctuations and higher demand peaks.

5.2 Generating cyber-physical attacks and running simulations

To create attack scenarios similar to the ones featured in BATADAL, we use epanetCPA [21]. First, the available attack descriptions were translated into executable by epanetCPA attack scenarios. Then, having available the attack scenarios, demand patterns and the C-Town EPANET input file for each dataset, we run the epanetCPA simulations to obtain the hydraulic response of the network. Note that the simulated SCADA readings during an attack are altered in an attempt to conceal the physical impact of the attack to the network thus making the CPA detection a challenging task.

The table below shows that the attacks in BATADAL include malicious activation of hydraulic actuators, change of actuator settings and deception attacks. Deception attacks are used to manipulate the information sent or received by sensors and PLCs and to alter the information received by SCADA.

Table 3: Attack scenarios featured in the BATADAL datasets.

Identifier	Starting time (dd/mm/YY HH)	Ending time (dd/mm/YY HH)	Duration (h)	Attack Description	SCADA concealment (altered readings)
1	13/09/16 23	16/09/16 00	50	Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.	Replay attack (L_T7).
2	26/09/16 11	27/09/16 10	24	Like Attack 1	Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).
3	09/10/16 09	11/10/16 20	60	Attacker alters L_T1 readings arrings to PLC2 with a constant low level. PLC1 recieves the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.	Polyline to offset L_T1 increase.
4	29/10/16 19	02/11/16 16	94	As in Attack 3	Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).
5	26/11/16 17	29/11/16 04	60	Working speed of PU7 reduces to 0.9 of nominal speed causes lower water levels in T4	-
6	06/12/16 07	10/12/16 04	94	As in Attack 5, but speed reduced to 0.7	Replay attack (L_T4).
7	14/12/16 15	19/12/16 04	110	As in attack 6	Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).

Identifier	Starting time (dd/mm/YY HH)	Ending time (dd/mm/YY HH)	Duration (h)	Attack Description	SCADA concealment (altered readings)
8	16/01/17 09	19/01/17 06	70	Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.	Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).
9	30/01/17 08	02/02/17 00	65	Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.	Polyline to offset L_T2 increase.
10	09/02/17 03	10/02/17 09	31	Malicious activation of Pump PU3.	-
11	12/02/17 01	13/02/17 07	31	As in Attack 10.	-
12	24/02/17 05	28/02/17 08	100	As in Attack 9.	Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)
13	10/03/17 14	13/03/17 21	80	Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.	Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.
14	25/03/17 20	27/03/17 01	30	Alteration of T4 signal arriving at PLC6.	-

In BATADAL each dataset has its own attack scenarios:

"dataset03" has no attacks (12-month dataset)

"dataset04" has 7 attacks (1-7) and has a 6-month duration

"testdataset" has 7 attacks (8-14) with a 3-month duration

As a result, the duration of each simulation defines the attack scenarios it will contain. One-year simulations do not contain any attacks, 6-month simulations contain attacks 1-7 and 3-month simulations contain attacks 8-14. The starting time and duration of each attack is the same as in the BATADAL datasets.

Therefore, the resulting simulations are nine with six of them containing attacks and three representing the hydraulic behavior of the network under normal conditions. In the table below are summarized the basic characteristics of all the datasets in our disposal.

Table 4: Available datasets summarization

Dataset	Duration (months)	Attacks	Demand Pattern Category	Source
dataset_r12	12	None	non-stochastic	epanetCPA simulation
dataset_r06	6	1-7	non-stochastic	epanetCPA simulation
dataset_r03	3	8-14	non-stochastic	epanetCPA simulation
dataset_b12	12	None	stochastic (Beta)	epanetCPA simulation
dataset_b06	6	1-7	stochastic (Beta)	epanetCPA simulation
dataset_b03	3	8-14	stochastic (Beta)	epanetCPA simulation
dataset_b012	12	None	stochastic (Gamma)	epanetCPA simulation
dataset_g06	6	1-7	stochastic (Gamma)	epanetCPA simulation
dataset_g03	3	8-14	stochastic (Gamma)	epanetCPA simulation
dataset03	12	None	non-stochastic	BATADAL
dataset04	6	1-7	non-stochastic	BATADAL
testdataset	3	8-14	non-stochastic	BATADAL

To further elaborate on the process of translating an attack scenario into an epanetCPA one and to showcase what the output of each simulation is, some of the key steps taken are described below:

First a table is constructed that breaks down the basic characteristics of each attack. This is a critical step towards implementing the attack scenarios described on BATADAL.

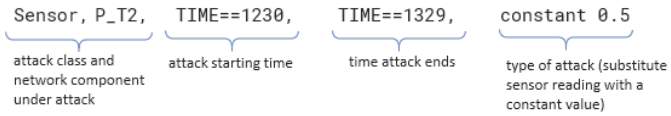
Table 5: Basic characteristics of each attack scenario in terms of Target-Action-Effect

ATTACK	TARGET	ACTION	EFFECT	epanetCPA ATTACK CATEGORY
1	SCADA to PLC9	Alter PU10, PU11 activation levels	PU10, PU11 off (T7 level decreases)	Attack On Control
	PLC9 to SCADA	Replay T7 level from the previous 48 h to SCADA	SCADA deception	Attack On Communication
2	SCADA to PLC9	Alter PU10, PU11 activation levels	PU10, PU11 off (T7 level decreases)	Attack On Control
	PLC9 to SCADA	Replay T7 level from the previous 48 h to SCADA	SCADA deception	Attack On Communication
	PLC5 to SCADA	Replay PU10 & PU11 flow and status from the previous 48 h to SCADA	SCADA deception	Attack On Communication
3	PLC2 to PLC1	Report T1_level = 0.5 m (low level) to PLC1	PU1 & PU2 on (T1 level increases)	Attack On Communication
	PLC2 to SCADA	Report T1 level with a -2.0 m offset	SCADA deception	Attack On Communication
4	PLC2 to PLC1	Report T1_level = 0.5 m (low level) to PLC1	PU1 & PU2 on (T1 level increases)	Attack On Communication
	PLC1 to SCADA	Replay T1 level, PU1 & PU2 flow and status, and J269 & J280 pressure from the previous 48 h to SCADA	SCADA deception	Attack On Communication
5	PU7	Speed of PU7 reduced to 0.9 of its nominal speed	T4 level decreases	Attack On Actuator
6	PU7	Speed of PU7 reduced to 0.7 of its nominal speed	T4 level decreases	Attack On Actuator

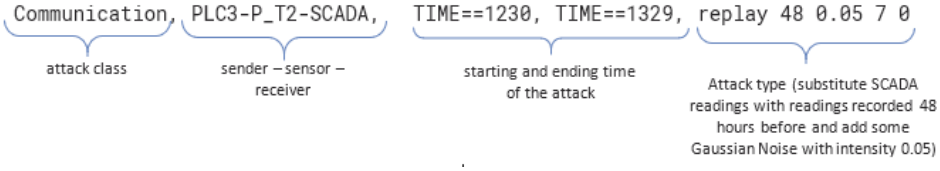
ATTACK	TARGET	ACTION	EFFECT	epanetCPA ATTACK CATEGORY
	PLC6 to SCADA	Replay T4 level from the previours 48 h	SCADA deception	Attack On Communication
7	PU7	Speed of PU7 reduced to 0.7 of its nominal speed	T4 level decreases	Attack On Actuator
	PLC2 to SCADA	Replay T1 level from the previous 48 h	SCADA deception	Attack On Communication
	PLC1 to SCADA	Replay PU1 & PU2 flow and status from the previous 48 hours	SCADA deception	Attack On Communication
8	PLC3	Alter PU4 & PU5 activation levels	PU4 & PU5 off (T3 level decreases)	Attack On Control
	PLC4 to SCADA	Replay T3 level from the previous 48 h	SCADA deception	Attack On Communication
	PLC3 to SCADA	Replay PU4 & PU5 flow and status from the previous 48 h	SCADA deception	Attack On Communication
9	Sensor T2	Report T2 level = 0.5 m (low level)	PLC3 deception	Attack On Sensor
	PLC3 to SCADA	Report T2 level with a -2.0 m offset	SCADA deception	Attack On Communication
10	PU3	Turn PU3 on	PU3 on	Attack On Actuator
11	PU4	Turn PU3 on	PU3 on	Attack On Actuator
12	Sensor T2	Report T2 level = 0.5 m (low level)	V2 on	Attack On Sensor
	PLC3 to SCADA	Replay T2 level, V2 flow and status and J422 & J14 pressure from the previous 48 h	SCADA deception	Attack On Communication
13	PLC5	Alter PU10, PU11 activation levels	PU10, PU11 switch on/off continuously	Attack On Control
	PLC9 to SCADA	Replay T7 level from the previous 48 h to SCADA	SCADA deception	Attack On Communication
	PLC5 to SCADA	Replay PU10 & PU11 flow and status and J317 & J307 pressure from the previous 48 h to SCADA. The replay attack of J307 terminates earlier	SCADA deception	Attack On Communication
14	PLC6 to PLC3	Alter T4 signal arriving to PLC6	T6 level increasing	Attack On Communication

Then each action is translated into epanetCPA executable scenarios. For instance, the attack #12 is implemented using two attack categories on epanetCPA:

1. Attack on a Sensor: The sensor monitoring the water level of Tank 2, transmits a constant low level equal to 0.5 m. According to the C-Town control rules Valve V2 is forced to open.



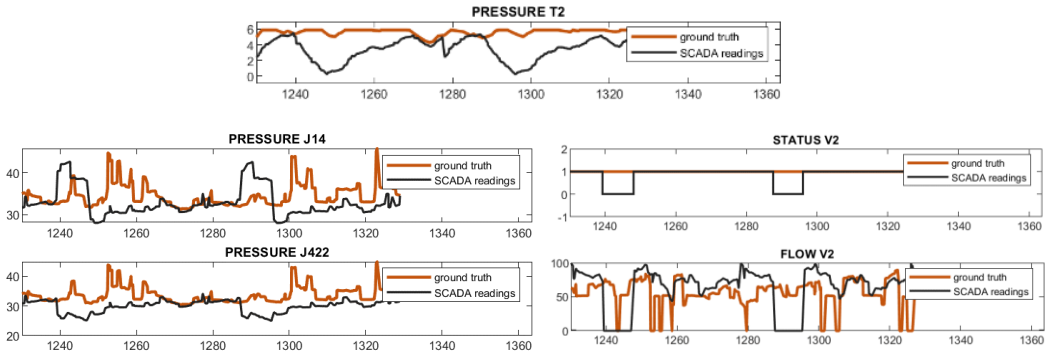
2. Attack on Communication: The measurements sent by the sensors reporting L_T2, F_V2, S_V2, P_J422 and P_J14, to SCADA, are substituted with data recorded during the same hour, two days before.



Similarly:

```
Communication, PLC3-F_V2-SCADA,   TIME==1230, TIME==1329, replay 48 0 120 0
Communication, PLC3-P_J422-SCADA, TIME==1230, TIME==1329, replay 48 0 120 0
Communication, PLC3-P_J14-SCADA,  TIME==1230, TIME==1329, replay 48 0 120 0
```

Finally, having available the demand patterns and the attack scenarios we can run the epanetCPA simulations. The effect of the attacks on the WDS can be showcased with an extract from "dataset_g03" during attack #12:



For more information, all attack scenarios and their effect on each dataset are available in Appendix.

PART II: DETECTING ATTACKS

6. PART II: DETECTING ATTACKS

In this dissertation the cyber-physical attack detection is done utilizing three different approaches. More specifically:

A. Two published machine learning approaches that have been implemented successfully to the BATADAL datasets by Taormina et al. and Kadosh et al. The first applies an Autoencoder to detect the attacks [4], while the other an SVDD classifier [3].
B. A third approach that hasn't been implemented to a water distribution network's CPA detection problem yet. It is based on the works of Teh et al.[5] and Covert et al [34] on graph-structured time series data.

6.1 Support Vector Data Description Classifier

Kadosh et al. proposed a one-class cyber-attack detection system (OCDS) using a Support Vector Data Description Classifier (SVDD) [3]. SVDD creates a spherical boundary around a numeric multi-dimensional dataset. By training SVDD with data under normal conditions, a boundary is created that can be used to separate anomalies (cyber-attacks) from data that are under normal condition operation.

The performance of the classification algorithm relies heavily on the chosen group of features. For that reason, the authors train the model using features of the WDS that are carefully selected based on the physical understanding of its topology. To do that, C-Town's five DMAs are utilized to assemble five groups of features. Each DMA's group of features is used as an input to a different classifier, meaning that a different model is constructed for each DMA.

To detect situations where pumps' control rules are violated (given that the control rules are a function of tanks' volumes), the authors added two extra features in their model. The first is the amount of storage in the DMA tanks at each time. While the second is the average inflow for each DMA. Average inflow in a DMA is used to avoid the overestimation of a DMA's inflow due to the sparse SCADA readings available (every 1 hour).

Each feature is defined below as:

$$amount\ of\ storage\ at\ time\ t\colon \Delta V_t = V_{t+1} - V_t,$$

where,

$V_t\colon the\ tank\ volume\ at\ time\ t$,

$V_t\colon sum\ of\ the\ volumes\ in\ case\ of\ more\ than\ on\ tanks\ in\ a\ DMA.$

and

$$average\ inflow\colon \bar{Q}_t^{in} = 0.5 \cdot \left(\bar{Q}_{t+1}^{in} + \bar{Q}_t^{in}\right)$$

where,

$\bar{Q}_t^{in}\colon inflow\ for\ the\ DMA\ at\ time\ t.$

Furthermore, to tackle the varying magnitudes and correlations of the demands during the day, each hour of the day has a dedicated classifier. That means that $5\ DMAs \times 24\ hours\ per\ day = 120$ different classifiers were trained. The DMA-based classifiers take into account the spatial variability, while the hourly-based consider the temporal variability of the WDS.

Each SVDD constructs a boundary for each DMA based on the attack-free training dataset. After the training, when we input an unknown to the SVDD sample, it outputs the distance (DV) of that sample from the boundary. If the distance is positive it means that the sample is out of the boundary, i.e. anomalous. A different distance is calculated for each DMA. To detect the attacks the maximum distance across the five DMAs is kept. Then given the fact that an adjacent sequence of positive distances increases the likelihood of an attack the authors use a moving average lag L, to derive a smoother aggregated DV. When the aggregated distance is above a threshold, TH, then the sample is identified as under attack. Both L and TH are hyperparameters of the SVDD. To choose the values of these hyperparameters the authors apply an automatic numerical process that choses the combination of the hyperparameters that perform better in detecting the attacks of the 6-month dataset. Finally, they test their algorithm using the "test_dataset" that has a 3-month horizon and seven attacks.

The metric score used to characterize the performance and to tune the L and TH hyperparameters, is the S score as defined in BATADAL.

In our experiments we evaluate the performance of the SVDD when trained with stochastically and non-stochastically generated datasets. More specifically, we train four different SVDD classifiers (equal to the number of attack-free datasets we have available) and evaluate their performance on the four 3-month test sets (dataset_r03/b03/g03 and batadal_03).

Table 6 shows the performance of each SVDD classifier on the test datasets. The first thing to notice is that (based on the S score) the classifier trained with the "batadal" dataset, fails to generalize on our datasets. Furthermore, the classifier trained with the non-stochastically generated dataset "random" seems to perform generally better on all test datasets.

However, after a visual inspection of the results, we notice that, the SVDD classifier, no matter the training set used, is prone to issuing a very large number of false positives. The reason behind this phenomenon might be that either the SVDD classifier is not a robust algorithm for anomaly detection regarding this particular problem, or that the S score used to optimize the hyperparameters is a misleading metric.

For future research we propose that the hyperparameters TH and L are tuned using the F1 score and for now, we deem this algorithm as unsuccessful.

Table 6: SVVD models' performance

train batadal	train random	train beta	train gamma
test batadal	test batadal	test batadal	test batadal
The Train Score is 0.956 The Test Score is 0.954 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.932 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.919 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.885 The optimal L is 3 The optimal TH is 0.012
test random	test random	test random	test random
The Train Score is 0.956 The Test Score is 0.781 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.897 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.876 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.877 The optimal L is 3 The optimal TH is 0.012
test beta	test beta	test beta	test beta
The Train Score is 0.956 The Test Score is 0.805 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.922 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.922 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.906 The optimal L is 3 The optimal TH is 0.012
test gamma	test gamma	test gamma	test gamma
The Train Score is 0.956 The Test Score is 0.791 The optimal L is 11 The optimal TH is 0.018	The Train Score is 0.929 The Test Score is 0.903 The optimal L is 8 The optimal TH is 0.013	The Train Score is 0.924 The Test Score is 0.879 The optimal L is 10 The optimal TH is 0.011	The Train Score is 0.919 The Test Score is 0.881 The optimal L is 3 The optimal TH is 0.012

6.2 **Autoencoder**

The second ML model used is an Autoencoder (AE) for CPA detection as proposed by Taormina et al. [4]. In the paper, the researchers use an AE to detect the attacks of BATADAL. Both the encoder and decoder of the proposed AE are two deep fully connected ANNs. In brief, the methodology outline is the following:

The goal of the AE is to learn a representation of the network under safe conditions. Consequently, it is trained to reconstruct its input based on the attack-free dataset (dataset03). Since the autoencoder is trained only on attack-free instances, when the input contains anomalies, i.e. concealed attacks, the reconstruction error is expected to be larger. By defining an error threshold ($theta$), the network is classified as "under attack" when the threshold is surpassed by the average reconstruction error of a window of length n hours (smoothing window).

The AE's architecture consists of a feedforward neural network acting as an encoder and another feedforward neural network acting as a decoder. The depth and width of each FNN layer is a function between the number of hidden layers (nl) and the compression factor (cf). The compression factor is the ratio of the size of the input layer (in this case 43, for 43 different SCADA readings) to that of the midmost hidden layer hosting the encoded representation. The attack-free dataset is split into train and validation set. The validation dataset is used to monitor the algorithm's performance during training and to avoid overfitting.

Two important hyperparameters of the proposed algorithm are the threshold $theta$ and the window length n. The threshold $theta$ is associated with different percentiles of the error distribution of the validation dataset. When the autoencoder outputs consecutively in a window length of n hours a reconstruction error greater than $theta$, then the network is classified as under attack.

Taormina et al. tested quite a few autoencoder architectures and various combinations of $theta$ and n to detect the CPAs of the BATADAL datasets. To compare the performance of the different architectures tested they referred to the F1 score, instead of the BATADAL S score.

Applying the Autoencoder on our datasets

In this study four different autoencoder models are trained. Each model is trained using one of the attack-free datasets available. Two thirds of the set are used for training and the rest for validation. A development test containing attacks is also used to fine-tune the threshold $theta$. The train and development datasets have to be originated from demand patterns drawn from the same distribution. For instance, if the autoencoder is trained with the dataset "dataset_g12" which has been simulated using demand patterns with marginal distribution the "Gamma distribution", then the threshold theta should be selected using the corresponding 6-month dataset, "dataset_g06". The rest of the datasets containing attacks are kept as test datasets to evaluate each autoencoder's performance. All models use a window length of 6 hours before issuing an alarm. This is based on the assumption

that we wish to know that the network is under attack, no later than six hours after the attack begins[5].

All AEs have the same number of hidden layers and compression factor to obtain comparable results. The recommended from the paper [4] values of nH and cf are used as default.

The only parameter different across the four AEs is the threshold $theta$. To select the $theta$ value, first each AE is trained. Then the F1 score of the development test is calculated for different values of $theta$. The $theta$ value corresponding to the highest F1 score for the development dataset is selected.

The table below summarizes the basic characteristics of each AE:

Table 7: Architecture of each Autoencoder model

Model's Name	Training Dataset	Development Dataset (to select $theta$ value)	nH	cf
random	dataset_r12	dataset_r06	5	2
beta	dataset_b12	dataset_b06	5	2
gamma	dataset_g12	dataset_g06	5	2
batadal	dataset03	dataset04	5	2

From now on beta and gamma AEs will also be referred (for clarity reasons) as AEs trained with stochastically generated data, and random and batadal as AEs trained with non-stochastically generated data.

Evaluating the models' performance

Evaluating the performance of each AE is not as straightforward as it seems. Neural Networks, like AEs, are non-deterministic algorithms that present a variance in their training performance. This is because the final outcome of the training depends among others, on the random weight initialization, mini batch randomization etc. Berg et al. in their paper [40] argue that the training variance is a phenomenon that should not be ignored when evaluating the performance of an algorithm. For that reason and to draw safer conclusions, each AE is trained 10 times and is evaluated in two stages. First by comparing the average performance of each AE and then by assessing the best one (out of the 10 trainings) from each category. The performance of the AEs is evaluated using the F1 score, instead of the BATADAL S score. S score is biased towards attack identification and is insensitive in false alarms. Given that a model should not issue many false alarms, S score is not an insightful

[5] Generally, larger windows allow for more confident predictions regarding the state of the network and decrease the number of false alarms issued due to outliers. However, a prompt attack detection is essential. The 6-hour window here is selected arbitrarily and not optimized, on the basis that the shortest attack scenario lasts 24 hours and for the shake of comparing the algorithms' performance with each other. A novel approach would be to select a window length after considering both the cost of detecting an attack n hours after it starts and the cost of responding to m false alarms.

metric when it comes to that matter, thus we report it only for comprehensive reasons.

Table 8: Mean performance of AE models.

Model	FP	FN	TP	Recall	Precision	F1 score	S score BATADAL
random	102	126	324	0.723 ± 0.07	0.773 ± 0.081	0.744 ± 0.058	0.854 ± 0.046
gamma	60	126	323	0.721 ± 0.102	0.85 ± 0.04	0.775 ± 0.063	0.853 ± 0.066
beta	78	120	329	0.735 ± 0.075	0.819 ± 0.061	0.771 ± 0.046	0.864 ± 0.045

To begin with, Table 8 shows the average performance of each model on our datasets (dataset_g06/ g03/ r06/ r03/ b06/ b03). It seems that AE gamma (i.e. trained with dataset_g12/ threshold chosen based on dataset g06) has the highest F1 score. However, when taking into account the standard deviations the betta and gamma models appear to behave quite similarly.

The gamma model appears to also have the highest precision, even when considering the standard deviation. With respect to recall, beta seems to perform better when it comes to detecting attacks.

As for the average FP, we get a sense that random tends to issue the most false alarms, and although beta and gamma are not much better at detecting True Positives (attacks), they issue less False Positives.

After getting a sense of the average performance of each model, we proceed with the examination of the average performance of each model on each test dataset.

Table 9 : Average Performance of the AE. Threshold has been finetuned for each one based on the f1 score

Model	FP	FN	TP	Rec	Pre	F1 score	S score (BATADAL)
							dataset_r06
random	92	151	341	0.693 ± 0.043	0.795 ± 0.065	0.738 ± 0.022	0.82 ± 0.045
gamma	59	149	343	0.697 ± 0.071	0.856 ± 0.03	0.766 ± 0.041	0.825 ± 0.069
beta	84	144	348	0.707 ± 0.06	0.812 ± 0.053	0.753 ± 0.025	0.83 ± 0.048
batadal	370	117	375	0.761 ± 0.044	0.504 ± 0.017	0.606 ± 0.016	0.861 ± 0.018
							dataset_r03
random	58	87	320	0.785 ± 0.077	0.848 ± 0.037	0.813 ± 0.045	0.882 ± 0.029
gamma	49	93	314	0.772 ± 0.13	0.868 ± 0.013	0.81 ± 0.078	0.876 ± 0.047
beta	54	88	319	0.784 ± 0.082	0.86 ± 0.037	0.817 ± 0.04	0.886 ± 0.025
batadal	176	76	331	0.812 ± 0.06	0.653 ± 0.014	0.723 ± 0.024	0.877 ± 0.02
							dataset_g06
random	173	151	341	0.692 ± 0.041	0.668 ± 0.057	0.678 ± 0.029	0.815 ± 0.036
gamma	86	142	350	0.711 ± 0.074	0.81 ± 0.054	0.753 ± 0.03	0.829 ± 0.068
beta	124	141	351	0.714 ± 0.064	0.748 ± 0.059	0.727 ± 0.028	0.838 ± 0.043
batadal	535	125	367	0.745 ± 0.041	0.407 ± 0.009	0.526 ± 0.013	0.834 ± 0.026
							dataset_g03
random	84	106	301	0.739 ± 0.075	0.783 ± 0.033	0.758 ± 0.038	0.878 ± 0.03
gamma	47	113	294	0.723 ± 0.117	0.866 ± 0.034	0.782 ± 0.073	0.866 ± 0.044
beta	60	107	300	0.737 ± 0.072	0.836 ± 0.042	0.781 ± 0.036	0.874 ± 0.029
batadal	244	88	319	0.783 ± 0.065	0.565 ± 0.012	0.656 ± 0.031	0.804 ± 0.025
							dataset_b06
random	140	156	336	0.683 ± 0.043	0.713 ± 0.064	0.695 ± 0.029	0.837 ± 0.039
gamma	73	155	337	0.685 ± 0.075	0.826 ± 0.042	0.745 ± 0.041	0.826 ± 0.094
beta	92	143	349	0.709 ± 0.073	0.798 ± 0.054	0.747 ± 0.032	0.854 ± 0.055
batadal	461	104	388	0.789 ± 0.058	0.458 ± 0.011	0.579 ± 0.017	0.875 ± 0.015
							dataset_b03
random	64	103	304	0.747 ± 0.077	0.829 ± 0.051	0.783 ± 0.045	0.893 ± 0.024
gamma	45	105	302	0.742 ± 0.128	0.871 ± 0.016	0.796 ± 0.081	0.896 ± 0.038
beta	53	98	309	0.759 ± 0.083	0.859 ± 0.048	0.802 ± 0.045	0.9 ± 0.025
batadal	206	86	321	0.789 ± 0.064	0.61 ± 0.013	0.687 ± 0.027	0.886 ± 0.02
							batadal_06
random	288	114	378	0.768 ± 0.083	0.704 ± 0.251	0.698 ± 0.144	0.788 ± 0.038
gamma	499	113	379	0.77 ± 0.154	0.665 ± 0.312	0.642 ± 0.19	0.753 ± 0.104
beta	595	104	388	0.788 ± 0.107	0.52 ± 0.265	0.575 ± 0.147	0.764 ± 0.047
batadal	63	109	383	0.779 ± 0.049	0.861 ± 0.036	0.816 ± 0.025	0.821 ± 0.028
							batadal_03
random	137	88	319	0.783 ± 0.079	0.774 ± 0.199	0.757 ± 0.098	0.889 ± 0.021
gamma	236	90	317	0.78 ± 0.136	0.728 ± 0.262	0.705 ± 0.147	0.836 ± 0.096
beta	273	77	331	0.812 ± 0.084	0.637 ± 0.232	0.681 ± 0.116	0.842 ± 0.063
batadal	33	84	323	0.794 ± 0.059	0.909 ± 0.026	0.846 ± 0.034	0.914 ± 0.018

At first sight, we can see that our datasets fail to generalize on the BATADAL datasets and vice versa. This is not unexpected as our datasets and the BATADAL have been simulated using completely different demand patterns.

Moreover, when focusing on the new datasets (r06/ r03/ g06/ g03/ b06/ b03), it is noticeable that the models trained with stochastically generated datasets (gamma & beta) have the highest F1 score on all datasets, even on those that are not stochastically generated (i.e. dataset_r06/ r03).

Again, AE gamma has the highest precision across all datasets, even when taking into consideration the standard deviation. This suggests that it tends to issue less

false alarms. Despite this, gamma model has the worst recall score, due to its high variance between runs.

Although beta model tends to issue more false alarms than gamma, it has the best average performance on all test datasets. This happens because beta model issues less false alarms than random, and it detects more "attack" instances than gamma does on each dataset.

Table 10: The best performing models (according to their performance on their development set)

Model	FP	FN	TP	Recall	Precision	F1 score	S score (BATADAL)
dataset_r06							
random	37	149	343	0.697	0.903	0.787	0.787
gamma	45	136	356	0.724	0.888	0.797	0.853
beta	79	121	371	0.754	0.824	0.788	0.872
batadal	425	94	398	0.809	0.484	0.605	0.876
dataset_r03							
random	37	66	341	0.838	0.902	0.869	0.900
gamma	51	60	347	0.853	0.872	0.862	0.902
beta	50	52	355	0.872	0.877	0.874	0.908
batadal	199	42	365	0.897	0.647	0.752	0.911
dataset_g06							
random	104	148	344	0.699	0.768	0.732	0.792
gamma	63	128	364	0.740	0.852	0.792	0.862
beta	116	118	374	0.760	0.763	0.762	0.867
batadal	590	89	403	0.819	0.406	0.543	0.868
dataset_g03							
random	56	97	310	0.762	0.847	0.802	0.873
gamma	35	88	319	0.784	0.901	0.838	0.884
beta	47	77	330	0.811	0.875	0.842	0.895
batadal	257	46	361	0.887	0.584	0.704	0.852
dataset_b06							
random	67	149	343	0.697	0.837	0.761	0.813
gamma	49	135	357	0.726	0.879	0.795	0.887
beta	77	110	382	0.776	0.832	0.803	0.908
batadal	524	67	425	0.864	0.448	0.590	0.872
dataset_b03							
random	41	86	321	0.789	0.887	0.835	0.910
gamma	44	76	331	0.813	0.883	0.847	0.917
beta	46	63	344	0.845	0.882	0.863	0.925
batadal	231	50	357	0.877	0.607	0.718	0.916
batadal_06							
random	158	97	395	0.803	0.714	0.756	0.831
gamma	216	84	408	0.829	0.654	0.731	0.832
beta	955	56	436	0.886	0.313	0.463	0.803
batadal	73	77	415	0.843	0.850	0.847	0.849
batadal_03							
random	87	72	335	0.823	0.794	0.808	0.906
gamma	101	65	342	0.840	0.772	0.805	0.909
beta	436	46	361	0.887	0.453	0.600	0.843
batadal	34	55	352	0.865	0.912	0.888	0.938

Table 10 shows the performance of the best model from each run, chosen based on the performance on their development set.

The models trained with stochastically generated datasets have once again the highest F1-score.

In this case, the gamma could be considered as the best model across all datasets, as it discloses a high number of TP, while it always issues less false alarms than beta.

Table 11: Average performance of all the AE models when their threshold-tuning set is drawn from the same distribution as the test set.

Model	Average of FP	Average of FN	Average of TP	Average of recall	Average of precision	Average of f1_score	Average of S
batadal_06				**THRESHOLD TUNING SET**			
batadal	65	114	378	0.768	0.854	0.809	0.821
random	36	138	354	0.719	0.908	0.802	0.776
gamma	33	146	346	0.703	0.913	0.793	0.777
beta	44	144	349	0.708	0.889	0.787	0.773
batadal_03				**TEST DATASET**			
batadal	28	80	327	0.803	0.921	0.858	0.914
beta	22	94	313	0.768	0.935	0.842	0.904
gamma	16	101	306	0.751	0.951	0.839	0.888
random	18	101	306	0.753	0.946	0.837	0.904

Model	Average of FP	Average of FN	Average of TP	Average of recall	Average of precision	Average of f1_score	Average of S
dataset_r06				**THRESHOLD TUNING SET**			
beta	78	130	362	0.737	0.828	0.778	0.858
gamma	73	148	344	0.699	0.829	0.757	0.828
random	99	151	341	0.694	0.786	0.734	0.815
batadal	251	160	332	0.675	0.651	0.617	0.777
dataset_r03				**TEST DATASET**			
beta	57	68	339	0.832	0.858	0.844	0.903
gamma	50	82	325	0.798	0.869	0.830	0.890
random	56	89	318	0.782	0.855	0.814	0.883
batadal	127	156	251	0.617	0.678	0.587	0.759

Model	Average of FP	Average of FN	Average of TP	Average of recall	Average of precision	Average of f1_score	Average of S
dataset_b06				**THRESHOLD TUNING SET**			
beta	88	133	359	0.730	0.810	0.766	0.875
gamma	83	142	350	0.711	0.819	0.757	0.859
random	148	151	341	0.692	0.711	0.696	0.844
batadal	443	98	394	0.801	0.501	0.593	0.855
dataset_b03				**TEST DATASET**			
beta	55	79	328	0.806	0.858	0.831	0.914
gamma	51	97	310	0.763	0.861	0.806	0.901
random	66	117	290	0.712	0.821	0.755	0.881
batadal	203	91	316	0.776	0.625	0.663	0.862

Model	Average of FP	Average of FN	Average of TP	Average of recall	Average of precision	Average of f1_score	Average of S
dataset_g06				**THRESHOLD TUNING SET**			
random	80	187	305	0.620	0.823	0.695	0.745
gamma	86	150	342	0.696	0.807	0.743	0.822
beta	80	141	351	0.713	0.821	0.760	0.827
batadal	21	262	230	0.468	0.918	0.620	0.584
dataset_g03				**TEST DATASET**			
random	45	189	218	0.535	0.846	0.630	0.788
gamma	46	119	289	0.709	0.866	0.774	0.865
beta	43	114	293	0.720	0.874	0.783	0.866
batadal	16	344	64	0.156	0.797	0.261	0.482

All the previous results show the performance of the models when their alarm threshold is tuned based on a development set that is from the same demand pattern category as the training set (e.g. train with batadal_12 and tune threshold with batadal_06 etc.). But what would happen to the model's performance if the threshold was tuned using a different development set?

A realistic assumption is that in ML problems, we usually know the test set we would like to do well on. Assuming that dataset r03/ b03/ g03 & batadal_03 are the test sets we care about and that for each test set we have a corresponding development set at our disposal (dataset_r06/ b06/ g06 & batadal_06), we train each model with one of the training sets, but we tune the anomaly threshold with the development set that corresponds to the test set we care about.

More specifically, assuming that we want to do well on the dataset "batadal_03", we will train four different models using the four different training datasets, but this time we will tune their anomaly threshold using the "batadal_06" dataset.

Table 11 shows the average performance of all the models when their threshold is tuned with a set drawn from the same distribution as the test set.

Note that, now that the development set has changed, all of our datasets generalize well on the BATADAL datasets. Although none of them surpasses the performance of the model trained with BATADAL datasets, all of the models perform quite similarly.

In the case of dataset_r03, beta and gamma models outperform the random model both in terms of precision as well as recall.

Finally, when it comes to datasets_g03 and dataset_b03 beta and then gamma models have the quite similar performances, while random performs noticeably worse in terms of recall (i.e. doesn't disclose as well the attacks).

6.3 Structural Convolutional Neural Networks

Detecting cyber-physical attacks with a model that apart from the temporal, considers the spatial structure of a WDS is an idea worth exploring for various reasons. First of all, capturing both the spatial and temporal features of the network as well as the correlations between them if done successfully is expected to improve the model's ability to detect contextual anomalies, thereby its ability to detect deception attacks. Furthermore, compared to algorithms that rely only on temporal information, the inclusion of the network's structure is expected to contribute into creating a more robust algorithm. This is because, structure is an inherent characteristic of the network and a form of prior knowledge, while the temporal information in a WDS are not only limited, but also accompanied by great uncertainty.

One way to include the spatial information in a machine learning algorithm is to consider the WDS as a graph-structured network. This is possible due to the innate interconnection between its components, which allow us to depict the water distribution network as a graph where its nodes (tanks, junctions etc.) are linked with edges (pipes). A graph-structured network can then be modelled using Graph Neural Networks (GNNs), a neural network type that operates on graphs. A special kind of GNNs are the Temporal Graph Neural Networks, which leverage the spatio-temporal information of time series data with an arbitrary graph topology.

In this dissertation, based on the work of Teh et al. [5] and Covert et al. [34], it is explored whether the inclusion of the graph-structure of C-Town will improve a model's performance in detecting cyber-physical attacks. The basic concept is the following: The model predicts the current SCADA measurements given n prior measurements. If the model's prediction is not close to the observed measurements (based on a predefined threshold) then an attack alarm is raised.

The methodology is divided into the following stages:

- The available event-free measurements are divided into subsequences using a sliding window with length n hours and a one-hour step size. The subsequences are then used as an input to the GNNs while the output is the observed measurements at the next time step.
- Based on the map of C-Town an adjacency matrix is created to describe the connections only between the sensors whose measurements are available.
- GNNs are trained to predict the measurements of the next hour
- Based on a hold-out dataset that contains attacks, a threshold theta is chosen to detect the attacks.
- Finally, the model's performance is evaluated using the remaining test datasets.

Choosing a window size

To obtain subsequences and use them as input data, a popular method is to take a window of a fixed size and slide it over the available datasets. The observations within each window represent the different samples of our training dataset.

Determining a window size is not straightforward. It depends on the network's response time to water demand changes. By doing some initial experiments we

choose a window size of 8 hours, assuming that after including 8 hours of previous observations, prior time points provide no additional information. Using a window size of less than 8 hours defeats the purpose of using a temporal model that operates on time series for the reason that, the subsequences would be too short to model.

A way to make predictions with more recent data would be to use measurements at a finer time scale, which in our case such measurements were not available.

Generally, a novel way to choose the optimal window size would be by examining the correlations coefficients between the demand patterns and all the variables of the network.

C-Town's Adjacency Matrix

The adjacency matrix of C-Town is extracted directly from the available map, where junctions, tanks, pumps and valves can be represented as nodes and pipes as the edges that define whether the nodes are adjacent or not.

Given that the available measurements are only from some of the network's variables, it is not possible to use the adjacency matrix of the whole town. As a result, a new, condensed adjacency matrix is created that describes the connections only between the nodes whose observations are available.

The resulting adjacency matrix will allow to embed into the model the topology of the WDS.

The adjacency matrix allows the exchange of information mainly between nodes that are connected with a direct edge. To allow the model to incorporate information from nodes that are reachable within k steps, the k-step reachability adjacency matrix is used in the model instead. To obtain it from A, the operation used is $A(k) = \mathbb{1}\,(A^k)$ where A^k is the adjacency matrix raised to the kth power, $\mathbb{1}(\cdot)$ is an element-wise indicator function, and $A(0) = I$. Setting $k > 1$ enables information to spread through the graph using fewer layers, setting k = 0 creates a layer that operates on each sequence separately. Preliminary results showed that A^2 tends to yield better results that A^1, so the 2-step reachability matrix was used on all the models described below.

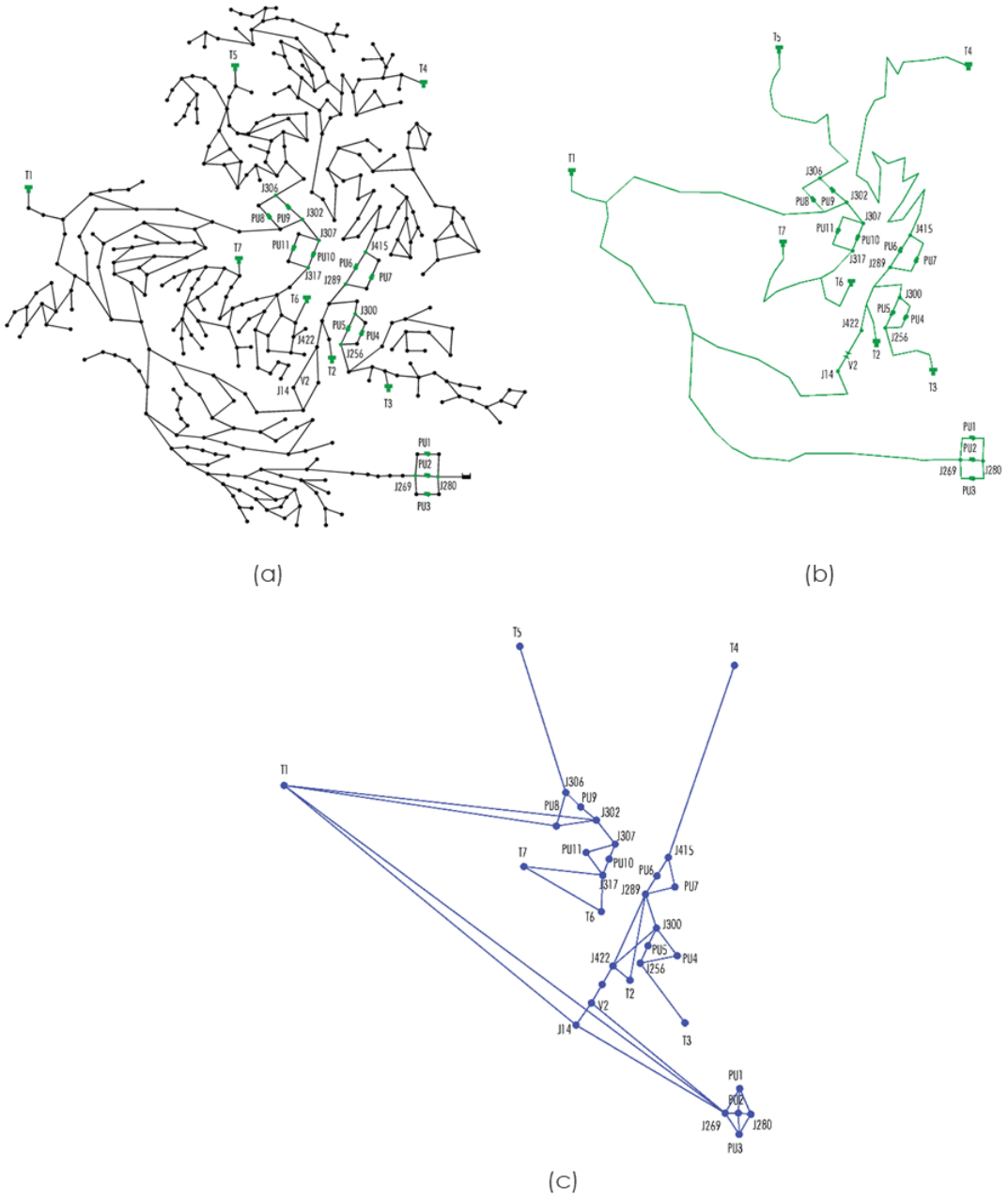


Figure 13: (a) The available variables' measurements of C-Town, (b) The resulting condensed network created based on the available variables, (c) The resulting (with 1-step reachability) Graph of the network.

Training the model

To learn the prediction task the architecture includes three stacked SCNN layers with 32, 64 and 128 output channels correspondingly. The output of the last SCNN layer is then flattened and passed in a fully connected layer with 150 neurons and then its output is passed to a final linear layer to get the next hour forecast. All layers use the ReLu activation function (except the linear layer, of course). The model is trained with the subsequences of length W (W=8 hours) taken in mini-batches of size B (B=16) from the training inputs X_{train} and targets Y_{train}. 30% of the training data (i.e. the one-year dataset without events) are held as a validation set. The model is trained by using the Adam optimizer to minimize the mean squared error (MSE) loss. Early stopping is applied to prevent overfitting of the model and to reduce overall time required for the training process. For this purpose, the MSE on the validation set is tracked. In most cases, 10-20 epochs are sufficient to reach a minimum of the validation error.

Detecting cyber-physical attacks

After the prediction model is trained, the validation time series X_{valid} is passed through the model and a tensor $\hat{Y}$ is predicted. Then the prediction errors $E = Y - \hat{Y}$ are calculated where Y contains the observed/target values for the next hour.

Estimating a multivariate Gaussian Distribution

One approach of statistically detecting anomalies, is to assume that the prediction errors are roughly Gaussian distributed and the parameters $(\vec{\mu}, S)$ of a multivariate Gaussian distribution can be estimated [41]. For that reason, the covariance matrix S and the mean vector $\vec{\mu}$ are calculated for the prediction errors matrix E.

Given $\vec{\mu}, S$ we can compute the Mahalanobis distance of a vector $\vec{x}$ from the mean vector $\vec{\mu}$. The squared Mahalanobis distance is defined as:

$$D_M(\vec{x}) = (\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})$$

Anomaly Detection

For data points that represent the network's status when not under attack, the corresponding Mahalanobis distance will be comparably small, since they are located close to the mean of the distribution. On the other side, when the network is under attack, the error vectors $\vec{e}$ are expected to have large values in one or more dimensions. Hence, the Mahalanobis distance can be used as a cyber-physical attack indicator. By specifying a distance threshold, all instances that their error has a Mahalanobis distance larger than the threshold will be flagged as anomalous.

While other researchers, use the average error across all parameters [4] or monitor the error of each sensor individually [24] we decided that the Mahalanobis distance is more fitting to detect attacks. This is because the average error does not take into consideration the different error range each sensor has, thus it is more sensitive to variables with higher error, while on the other hand, monitoring the error of each sensor individually requires to fine tune multiple thresholds to define an attack rather than just one global threshold.

Choosing an attack threshold

Depending on the choice of the threshold, more or less points will be classified as anomalous. If the threshold is set too small, the algorithm will likely produce many false detections. If the threshold is chosen too large, some attacks might be missed. In an attempt to avoid both scenarios, two different thresholds are incorporated in our methodology. The first is $threshold_{lower}$, that monitors the error in a window of length n=6hours and when more than 5 out of 6 instances' errors surpass the threshold, an alarm is issued. The second is $threshold_{upper}$, that issues an alarm immediately when the error of an instance is greater than it. To fine-tune the value of each threshold, we try different values until we maximize the F1 score and the precision correspondingly on a hold-out set that contains attacks.

Results

Like in the case of AEs, we define four different models based on their training set. Each model was trained 10 times and we report the performance of these models by the mean and standard deviation of their performance.

Table 12: Comparison between the SCNN models. Each entry shows the mean and standard deviation across 10 runs

MODEL	FP	FN	TP	Recall	Precision	F1 score	S score (BATADAL)
random	53	138	311	0.697 ± 0.0567	0.856±0.0364	0.767±0.0375	0.901±0.0329
gamma	36	155	295	0.660 ± 0.0607	0.890±0.0206	0.756±0.0364	0.873±0.0554
beta	59	131	318	0.712 ± 0.0584	0.848±0.0392	0.772±0.0375	0.901±0.0385

Table 12 shows the average performance of each model on our datasets (dataset_g06/g03/r06/r03/b06/b03). It seems that the model beta has the highest F1 score.

Gamma model although has the highest precision it also has the lowest recall. That might indicate that it is the least sensitive among all models in disclosing attacks but the least prone in issuing false alarms.

Beta seems to perform better, but similarly to random model. Beta has the highest recall score among all models, meaning that it is the model that detects most of the "attack" instances on average.

Table 13: Average performance and standard deviation of SCNN across 10 trainings

MODEL	FP	FN	TP	Recall	Precision	F1 score	S score (BATADAL)
							dataset_r06
random	42	172	320	0.65 ± 0.026	0.886 ± 0.028	0.749 ± 0.012	0.853 ± 0.034
gamma	28	193	299	0.607 ± 0.025	0.913 ± 0.013	0.729 ± 0.019	0.806 ± 0.051
beta	66	172	320	0.651 ± 0.032	0.835 ± 0.059	0.729 ± 0.01	0.835 ± 0.029
batadal	542	104	388	0.788 ± 0.042	0.425 ± 0.058	0.549 ± 0.041	0.907 ± 0.008
							dataset_r03
random	43	107	300	0.737 ± 0.036	0.875 ± 0.014	0.8 ± 0.02	0.922 ± 0.013
gamma	36	119	288	0.708 ± 0.05	0.89 ± 0.009	0.788 ± 0.03	0.912 ± 0.017
beta	49	102	305	0.75 ± 0.036	0.862 ± 0.023	0.801 ± 0.017	0.922 ± 0.014
batadal	246	48	359	0.882 ± 0.031	0.597 ± 0.047	0.711 ± 0.026	0.936 ± 0.005
							dataset_g06
random	84	167	326	0.662 ± 0.023	0.796 ± 0.027	0.722 ± 0.011	0.883 ± 0.008
gamma	41	175	317	0.643 ± 0.011	0.886 ± 0.013	0.746 ± 0.005	0.855 ± 0.029
beta	80	159	333	0.678 ± 0.025	0.808 ± 0.039	0.736 ± 0.011	0.885 ± 0.016
batadal	590	104	388	0.789 ± 0.033	0.402 ± 0.046	0.531 ± 0.035	0.903 ± 0.009
							dataset_g03
random	47	102	305	0.75 ± 0.042	0.867 ± 0.017	0.803 ± 0.023	0.929 ± 0.01
gamma	36	120	287	0.705 ± 0.055	0.889 ± 0.008	0.785 ± 0.035	0.916 ± 0.019
beta	53	92	315	0.774 ± 0.037	0.858 ± 0.025	0.813 ± 0.015	0.935 ± 0.008
batadal	321	43	364	0.895 ± 0.015	0.536 ± 0.05	0.669 ± 0.037	0.928 ± 0.007
							dataset_b06
random	53	175	317	0.644 ± 0.031	0.857 ± 0.032	0.734 ± 0.017	0.889 ± 0.018
gamma	31	199	293	0.595 ± 0.021	0.904 ± 0.023	0.717 ± 0.012	0.833 ± 0.055
beta	52	165	328	0.666 ± 0.034	0.865 ± 0.034	0.751 ± 0.015	0.895 ± 0.02
batadal	543	102	390	0.793 ± 0.029	0.425 ± 0.051	0.551 ± 0.038	0.905 ± 0.008
							dataset_b03
random	51	107	300	0.738 ± 0.045	0.856 ± 0.011	0.792 ± 0.025	0.927 ± 0.011
gamma	46	123	284	0.698 ± 0.054	0.861 ± 0.006	0.77 ± 0.034	0.914 ± 0.017
beta	50	99	308	0.756 ± 0.032	0.859 ± 0.011	0.804 ± 0.018	0.932 ± 0.008
batadal	225	58	349	0.858 ± 0.022	0.614 ± 0.054	0.714 ± 0.031	0.933 ± 0.005
							batadal06
random	24	238	255	0.517 ± 0.041	0.916 ± 0.015	0.66 ± 0.033	0.855 ± 0.015
gamma	19	257	235	0.477 ± 0.03	0.925 ± 0.018	0.629 ± 0.024	0.842 ± 0.012
beta	39	222	270	0.549 ± 0.038	0.878 ± 0.045	0.674 ± 0.021	0.862 ± 0.017
batadal	65	181	311	0.632 ± 0.042	0.83 ± 0.031	0.716 ± 0.02	0.891 ± 0.01
							batadal03
random	30	163	244	0.6 ± 0.052	0.891 ± 0.007	0.716 ± 0.038	0.883 ± 0.014
gamma	28	177	231	0.566 ± 0.042	0.893 ± 0.006	0.692 ± 0.032	0.877 ± 0.011
beta	34	158	250	0.613 ± 0.046	0.881 ± 0.013	0.722 ± 0.031	0.887 ± 0.012
batadal	45	122	285	0.7 ± 0.038	0.866 ± 0.023	0.773 ± 0.019	0.906 ± 0.009

At first sight, we can see that the batadal model fails to generalize on our data. On the other side, SCNN models trained with our data do well on the batadal datasets even when their anomaly threshold is fine-tuned using a dataset that is drawn from a different distribution.

Note how that was not the case in the corresponding AE models. Their performance on the batadal datasets improved only after changing the threshold tuning dataset.

This might indicate that either SCNNs present less variance to the development set or that the Mahalanobis distance helps to capture better the differences of the prediction error between anomalous and normal data.

When it comes to the performance of the models on our datasets, beta tends to perform the best on almost all test datasets (based on the F1 score). Beta also presents smaller variation (considering the standard deviations), thus making it a more stable model.

Gamma distribution has the highest precision on all datasets, but the lowest recall.

Beta has the highest recall score, but issues more false alarms than random. Generally, someone might say that the differences in performance between random and beta are minor and that they perform similarly on average.

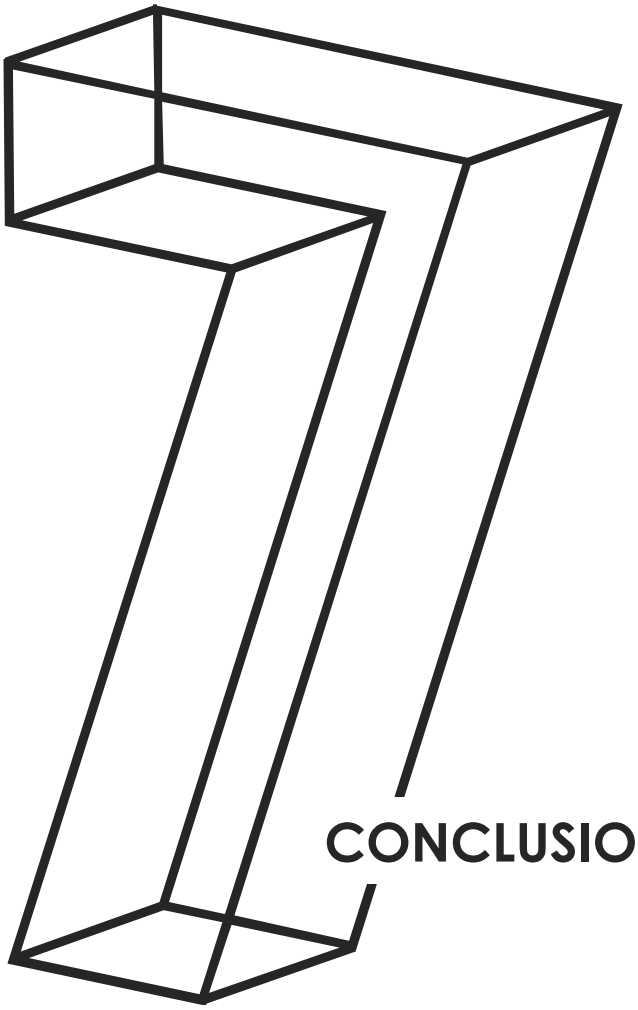
Table 14 : Performance of the best SCNN models (chosen based on their performance on the development set).

Row Labels	Sum of FP	Sum of FN	Sum of TP	Sum of Recall	Sum of Precision	Sum of F1 score	Sum of S
							dataset_r06
random	50	150	342	0.695	0.872	0.774	0.895
gamma	36	184	308	0.626	0.895	0.737	0.820
beta	93	152	340	0.691	0.785	0.735	0.885
batadal	708	74	418	0.850	0.371	0.517	0.914
							dataset_r03
beta	55	75	332	0.816	0.858	0.836	0.946
gamma	43	86	321	0.789	0.882	0.833	0.941
random	51	87	320	0.786	0.863	0.823	0.939
batadal	308	31	376	0.924	0.550	0.689	0.937
							dataset_g06
beta	83	143	349	0.709	0.808	0.755	0.906
gamma	48	167	325	0.661	0.871	0.751	0.868
random	99	142	350	0.711	0.780	0.744	0.888
batadal	776	85	407	0.827	0.344	0.486	0.897
							dataset_g03
beta	62	65	342	0.840	0.847	0.843	0.950
random	48	80	327	0.803	0.872	0.836	0.943
gamma	43	87	320	0.786	0.882	0.831	0.939
batadal	403	39	368	0.904	0.477	0.625	0.919
							dataset_b06
beta	60	143	349	0.709	0.853	0.775	0.920
random	58	148	344	0.699	0.856	0.770	0.919
gamma	42	181	311	0.632	0.881	0.736	0.880
batadal	738	97	395	0.803	0.349	0.486	0.895
							dataset_b03
beta	52	79	328	0.806	0.863	0.834	0.944
random	52	80	327	0.803	0.863	0.832	0.944
gamma	53	85	322	0.791	0.859	0.824	0.940
batadal	336	46	361	0.887	0.518	0.654	0.924
							batadal06
batadal	102	137	355	0.722	0.777	0.748	0.911
beta	49	188	304	0.618	0.861	0.720	0.885
random	30	204	288	0.585	0.906	0.711	0.874
gamma	22	236	256	0.520	0.921	0.665	0.861
							batadal03
random	35	118	289	0.710	0.892	0.791	0.912
batadal	65	103	304	0.747	0.824	0.784	0.916
beta	37	125	282	0.693	0.884	0.777	0.909
gamma	29	136	271	0.666	0.903	0.767	0.902

Table 14 shows the performance of the best SCNN models (chosen based on their performance on the development set).

We notice that gamma is inferior to random and beta models, and although it is the most precise it fails to disclose as many TP as the other models

We would argue that when it comes to the relative comparison between the best models, no model seems to outperform significantly the others.



CONCLUSIONS

7.CONCLUSIONS

To summarize, in this dissertation we approached the problem of cyber-physical attack detection on water distribution systems with the use of machine learning algorithms. To train the different models we used two kinds of datasets. Datasets that were stochastically generated in terms of the water demand variation and datasets with fairly regular and consistent demand patterns.

Some of the major conclusions of this study are the following:

- The training dataset is directly correlated to the algorithm's performance. We noticed that in the case of the Autoencoder algorithm, stochastically generated training datasets tend to improve its performance. More specifically, training with data generated from the Beta distribution improved the algorithm in terms of issuing less false positives and detecting more true positives. However, training with stochastically generated datasets isn't always reliable, like in the case of training with data generated from Gamma distribution which made all algorithms less sensitive in detecting attacks.
- AEs performance is very sensitive to the choice of the anomaly threshold. We observed, that by just changing the threshold tuning set the AEs performance on the BATADAL datasets changed dramatically.
- The choice of a single evaluation metric is of great importance. First of all, it allows us to evaluate the different models' performance. Moreover, in the case where the threshold is tuned based on the value of a metric score, the metric score we choose can have a great impact on the model's performance. We saw that in the case of the SVDD classifier where by choosing TH based on the maximum S score from BATADAL, failed to have a good performance.
- Before choosing an evaluation metric, first we have to set a clear objective about what makes an algorithm have a good performance in the framework of Water Systems' security. The problem of detecting cyber-physical attacks on a water distribution network is very particular. The dataset classes ("attack"/"no attack") are imbalanced and while there are many metrics in machine learning that tackle the problem of imbalanced datasets, not all "attack" instances are equally important in a CPA dataset. That means that it should not be equally important to detect "attack" instances towards the end of an attack vs. at the beginning of it. The same concept applies also in the detection of FP. For example, a model that issues a false alarm for 24 hours during a whole day is not performing equally well with a model that issued 24 hours of false alarms in the course of a month.
- Preliminary results of the SCNN models showed that they have less variance to the fluctuations of the stochastically generated data, as models trained with different datasets didn't have substantial differences in performance like in the case of AE.
- SCNNs performed better than AEs in the test datasets. Although AEs and SCNNs have similar F1-score, SCNNs presented lower variance between runs, thus making them more stable.

To conclude, our rationale behind using stochastically generated datasets in the problem of WDS CPA detection was twofold. Firstly, using novel stochastic methods to generate the water demand timeseries allowed us go a step further towards creating more realistic simulation scenarios. This is because stochastically generated water demands let us incorporate (to an extent) into the CPA detection problem, the uncertainty associated with the variability and stochastic nature one of the key components of urban water systems. Moreover, it lets us set a clear direction towards anomaly detection progress. Given that stochastically generated data allow to incorporate and study a large number of alternative scenarios, extending the "bounded horizon" of observed data, it is of greater value to create algorithms that perform well on them. With a novel stochastic method and a robust model that performs well on stochastically generated data it is possible to open the path for domain adaptation and robust learning in the water sector.

The second reason behind incorporating stochastically generated datasets to our experiments was to observe the performance of different anomaly detection algorithms. The core of machine learning is to create algorithms that learn from a training set of data and our experiments showed that not only the performance of the aforementioned CPA detection algorithms depends to an extent on the data used during training, but that there are training datasets that are more valuable than others, because they improve and generalize the algorithm's performance.

BIBLIOGRAPHY

[1] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," *Proc. - 3rd IEEE Int. Conf. Data Sci. Adv. Anal. DSAA 2016*, pp. 399–410, 2016, doi: 10.1109/DSAA.2016.49.

[2] P. Kossieris, I. Tsoukalas, C. Makropoulos, and D. Savic, "Simulating marginal and dependence behaviour of water demand processes at any fine time scale," *Water (Switzerland)*, vol. 11, no. 5, 2019, doi: 10.3390/w11050885.

[3] N. Kadosh, A. Frid, and M. Housh, "Detecting cyber-physical attacks in water distribution systems: one-class classifier approach," *J. Water Resour. Plan. Manag.*, vol. 146, no. 8, pp. 1–13, 2020, doi: 10.1061/(ASCE)WR.1943-5452.0001259.

[4] R. Taormina and S. Galelli, "Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems," *J. Water Resour. Plan. Manag.*, vol. 144, no. 10, pp. 1–15, 2018, doi: 10.1061/(ASCE)WR.1943-5452.0000983.

[5] T. Teh, C. Auepanwiriyakul, J. A. Harston, and A. A. Faisal, "Generalised Structural CNNs (SCNNs) for time series data with arbitrary graph topology," 2018, [Online]. Available: http://arxiv.org/abs/1803.05419.

[6] V. L. Do, L. Fillatre, and I. Nikiforov, "A statistical method for detecting cyber/physical attacks on SCADA systems," *2014 IEEE Conf. Control Appl. CCA 2014*, pp. 364–369, 2014, doi: 10.1109/CCA.2014.6981373.

[7] R. Taormina *et al.*, "Battle of the Attack Detection Algorithms: Disclosing cyber attacks on water distribution networks," *J. Water Resour. Plan. Manag.*, vol. 144, no. 8, 2018, doi: 10.1061/(ASCE)WR.1943-5452.0000969.

[8] J. B. Marco, R. Harboe, and J. D. Salas, Eds., *Stochastic Hydrology and its Use in Water Resources Systems Simulation and Optimization*. Dordrecht: Springer Netherlands, 1993.

[9] S. G. Buchberger and L. Wu, "Model for Instantaneous Residential Water Demands," *J. Hydraul. Eng.*, vol. 121, no. 3, pp. 232–246, Mar. 1995, doi: 10.1061/(ASCE)0733-9429(1995)121:3(232).

[10] S. G. Buchberger and G. J. Wells, "Intensity, Duration, and Frequency of Residential Water Demands," *J. Water Resour. Plan. Manag.*, vol. 122, no. 1, pp. 11–19, Jan. 1996, doi: 10.1061/(ASCE)0733-9496(1996)122:1(11).

[11] E. Creaco, P. Kossieris, L. Vamvakeridou-Lyroudia, C. Makropoulos, Z. Kapelan, and D. Savic, "Parameterizing residential water demand pulse models through smart meter readings," *Environ. Model. Softw.*, vol. 80, pp. 33–40, Jun. 2016, doi: 10.1016/j.envsoft.2016.02.019.

[12] S. Alvisi, M. Franchini, and A. Marinelli, "A Stochastic Model for Representing Drinking Water Demand at Residential Level," *Water Resour. Manag.*, vol. 17, no. 3, pp. 197–222, 2003, doi: 10.1023/A:1024100518186.

[13] P. Kossieris, C. Makropoulos, E. Creaco, L. Vamvakeridou-Lyroudia, and D. A. Savic, "Assessing the Applicability of the Bartlett-Lewis Model in Simulating Residential Water Demands," *Procedia Eng.*, vol. 154, pp. 123–131, 2016, doi: 10.1016/j.proeng.2016.07.429.

[14] P. Kossieris, "Multi-scale stochastic analysis and modelling of residential water demand processes," National Technical University of Athens, 2020.

[15] R. Gargano, C. Tricarico, G. del Giudice, and F. Granata, "A stochastic model for daily residential water demand," *Water Supply*, vol. 16, no. 6, pp. 1753–1767, Dec. 2016, doi: 10.2166/ws.2016.102.

[16] Kossieris, "Multi-scale stochastic analysis and modelling of residential water demand processes," *PhD thesis, Dep. Water Resour. Environ. Eng.*, p. 304, 2020.

[17] I. Tsoukalas, C. Makropoulos, and D. Koutsoyiannis, "Simulation of Stochastic Processes Exhibiting Any-Range Dependence and Arbitrary Marginal Distributions," *Water Resour. Res.*, vol. 54, no. 11, pp. 9484–9513, Nov. 2018, doi: 10.1029/2017WR022462.

[18] I. Tsoukalas, A. Efstratiadis, and C. Makropoulos, "Stochastic Periodic Autoregressive to Anything (SPARTA): Modeling and Simulation of Cyclostationary Processes With Arbitrary Marginal Distributions," *Water Resour. Res.*, vol. 54, no. 1, pp. 161–185, Jan. 2018, doi: 10.1002/2017WR021394.

[19] L. Perelman and S. Amin, "A network interdiction model for analyzing the vulnerability of water distribution systems," *HiCoNS 2014 - Proc. 3rd Int. Conf. High Confid. Networked Syst. (Part CPS Week)*, pp. 135–144, 2014, doi: 10.1145/2566468.2566480.

[20] S. Adepu, V. R. Palleti, G. Mishra, and A. Mathur, "Investigation of Cyber Attacks on a Water Distribution System," vol. 0, no. 0, pp. 1–23, 2019, [Online]. Available: http://arxiv.org/abs/1906.02279.

[21] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, and A. Ostfeld, "Characterizing cyber-physical attacks on water distribution systems," *J. Water Resour. Plan. Manag.*, vol. 143, no. 5, pp. 1–12, 2017, doi: 10.1061/(ASCE)WR.1943-5452.0000749.

[22] D. Nikolopoulos, G. Moraitis, D. Bouziotas, A. Lykou, G. Karavokiros, and C. Makropoulos, "Cyber-Physical Stress-Testing Platform for Water Distribution Networks," *J. Environ. Eng. (United States)*, vol. 146, no. 7, pp. 1–21, 2020, doi: 10.1061/(ASCE)EE.1943-7870.0001722.

[23] M. Housh and Z. Ohar, "Model-based approach for cyber-physical attack detection in water distribution systems," *Water Res.*, vol. 139, no. March, pp. 132–143, 2018, doi: 10.1016/j.watres.2018.03.039.

[24] A. A. Abokifa, K. Haddad, C. Lo, and P. Biswas, "Real-time identification of cyber-physical attacks on water distribution systems via machine learning-based anomaly detection techniques," *J. Water Resour. Plan. Manag.*, vol. 145, no. 1, pp. 1–13, 2019, doi: 10.1061/(ASCE)WR.1943-5452.0001023.

[25] M. Giacomoni, N. Gatsis, and A. Taha, "Identification of Cyber Attacks on

Water Distribution Systems by Unveiling Low-Dimensionality in the Sensory Data," in *World Environmental and Water Resources Congress 2017*, pp. 660–675.

[26] B. M. Brentan *et al.*, "On-Line Cyber Attack Detection in Water Networks through State Forecasting and Control by Pattern Recognition," in *World Environmental and Water Resources Congress 2017*, pp. 583–592.

[27] S. E. Chandy, A. Rasekh, Z. A. Barker, and M. Ehsan Shafiee, "Cyberattack detection using deep generative models with variational inference," *J. Water Resour. Plan. Manag.*, vol. 145, no. 2, 2019, doi: 10.1061/(ASCE)WR.1943-5452.0001007.

[28] M. F. K. Pasha, B. Kc, and S. L. Somasundaram, "An Approach to Detect the Cyber-Physical Attack on Water Distribution System," in *World Environmental and Water Resources Congress 2017*, pp. 703–711.

[29] M. Aghashahi, R. Sundararajan, M. Pourahmadi, and M. K. Banks, "Water distribution systems analysis symposium-battle of the attack detection algorithms (BATADAL)," *World Environ. Water Resour. Congr. 2017 Int. Perspect. Hist. Heritage, Emerg. Technol. Student Pap. - Sel. Pap. from World Environ. Water Resour. Congr. 2017*, no. May 2017, pp. 101–108, 2017, doi: 10.1061/9780784480595.010.

[30] M. Kravchik and A. Shabtai, "Efficient Cyber Attacks Detection in Industrial Control Systems Using Lightweight Neural Networks and PCA," pp. 1–18, 2019, [Online]. Available: http://arxiv.org/abs/1907.01216.

[31] D. T. Ramotsoela, G. P. Hancke, and A. M. Abu-Mahfouz, "Attack detection in water distribution systems using machine learning," *Human-centric Comput. Inf. Sci.*, vol. 9, no. 1, 2019, doi: 10.1186/s13673-019-0175-8.

[32] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. XX, no. Xx, pp. 1–21, 2020, doi: 10.1109/tnnls.2020.2978386.

[33] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2018-July, pp. 3634–3640, 2018, doi: 10.24963/ijcai.2018/505.

[34] I. Covert *et al.*, "Temporal Graph Convolutional Networks for Automatic Seizure Detection," pp. 1–19, 2019, [Online]. Available: http://arxiv.org/abs/1905.01375.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[36] A. Ostfeld *et al.*, "Battle of the water calibration networks," *J. Water Resour. Plan. Manag.*, vol. 138, no. 5, pp. 523–532, 2012, doi: 10.1061/(ASCE)WR.1943-5452.0000191.

[37] R. Taormina, S. Galelli, H. C. Douglas, N. O. Tippenhauer, E. Salomons, and A. Ostfeld, "Modeling Cyber-Physical Attacks on Water Networks with epanetCPA," *WDSA/CCWI Jt. Conf. Proc.*, vol. 1, 2018.

[38] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, and A. Ostfeld, "Simulation of cyber-physical attacks on water distribution systems with

EPANET," *Cryptol. Inf. Secur. Ser.*, vol. 14, pp. 123–130, 2016, doi: 10.3233/978-1-61499-617-0-123.

[39] I. Tsoukalas, P. Kossieris, and C. Makropoulos, "Simulation of non-gaussian correlated random variables, stochastic processes and random fields: Introducing the anysim r-package for environmental applications and beyond," *Water (Switzerland)*, vol. 12, no. 6, 2020, doi: 10.3390/w12061645.

[40] E. Van Den Berg, B. Ramabhadran, and M. Picheny, "Training variance and performance evaluation of neural networks in speech," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 2287–2291, 2017, doi: 10.1109/ICASSP.2017.7952564.

[41] M. Thill, S. Däubener, W. Konen, and T. Bäck, "Anomaly detection in electrocardiogram readings with stacked LSTM networks," *CEUR Workshop Proc.*, vol. 2473, pp. 17–25, 2019.

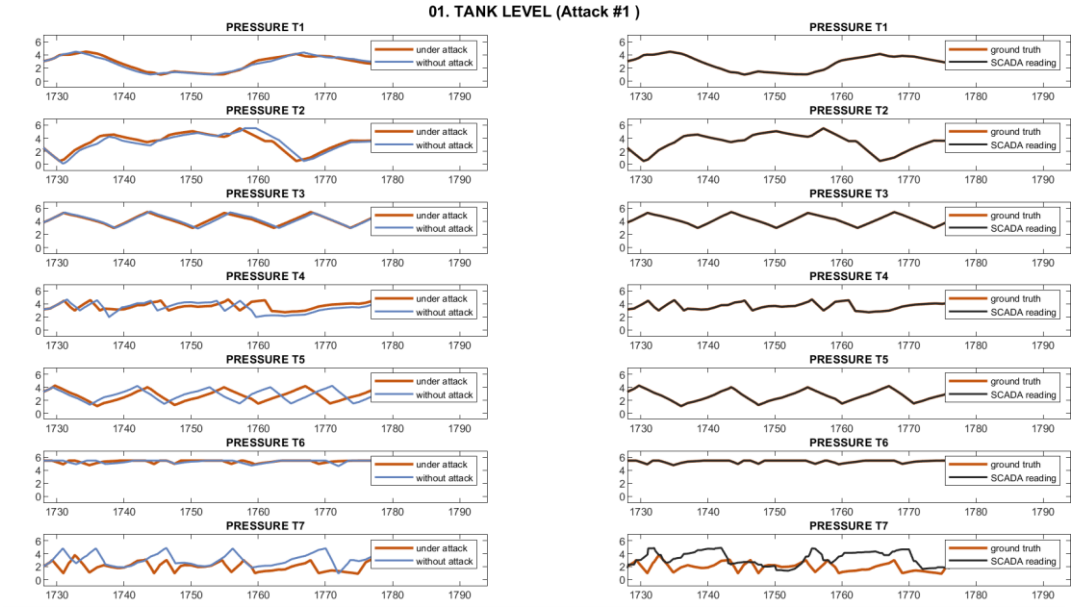
APPENDIX

APPENDIX A

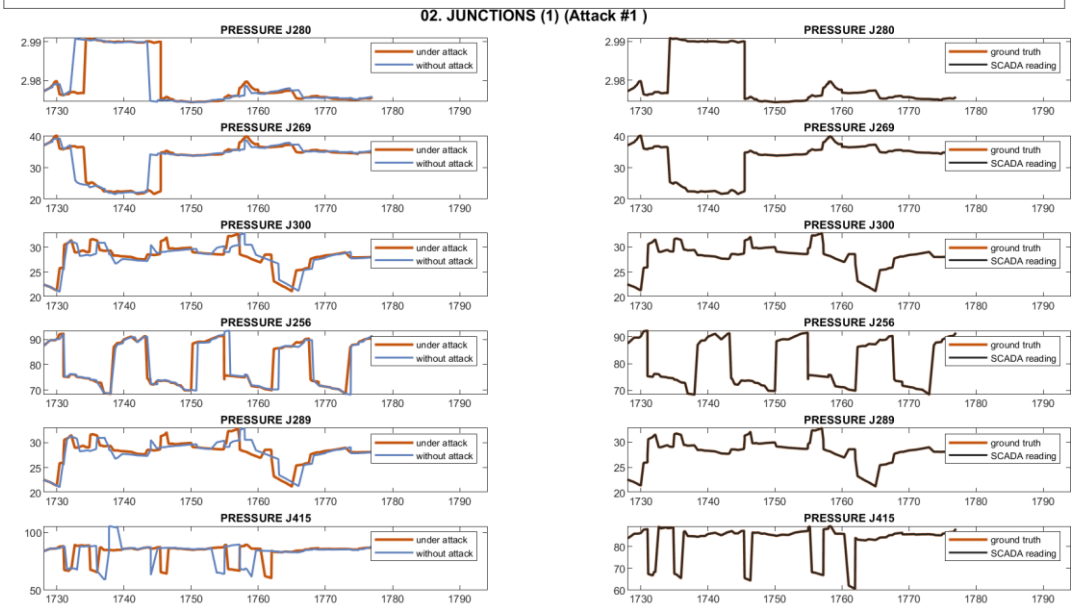
epanetCPA simulations

(only for the beta simulation)

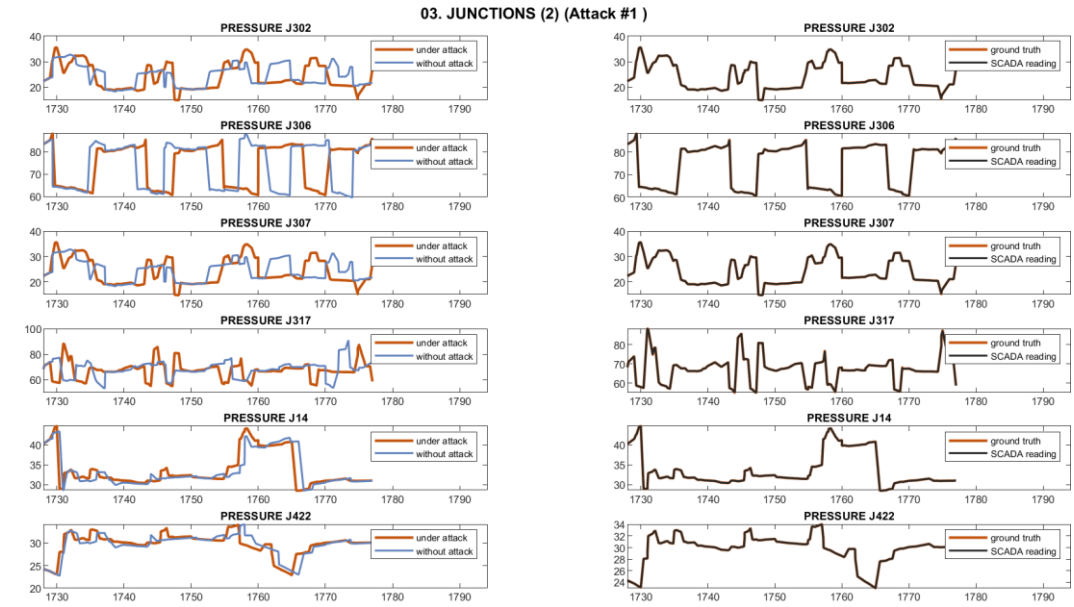
dataset_b06



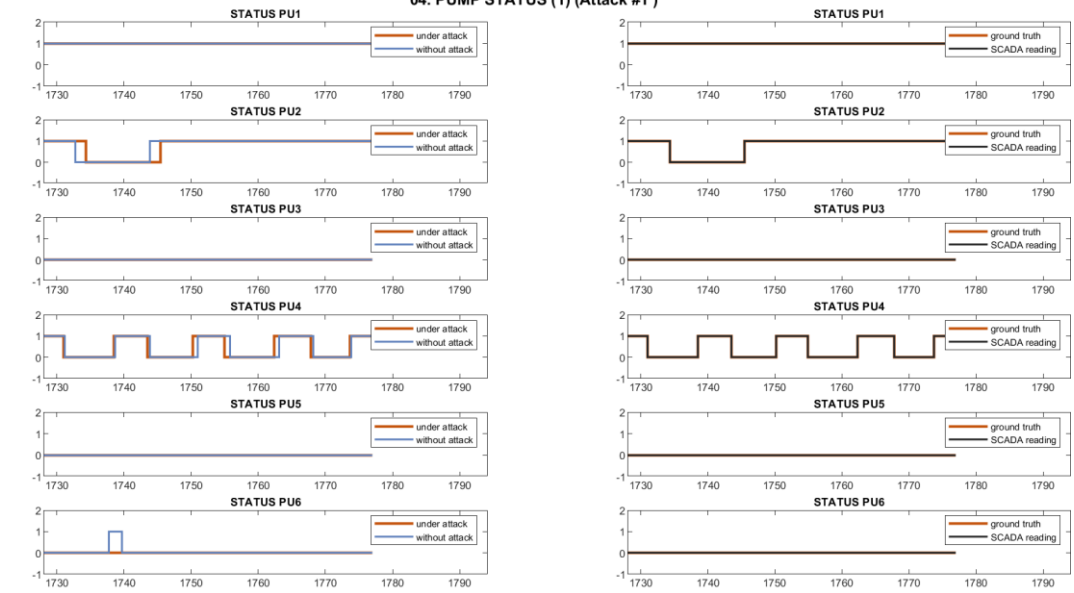
Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.
SCADA concealment: Replay attack (L_T7).



Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.
SCADA concealment: Replay attack (L_T7).

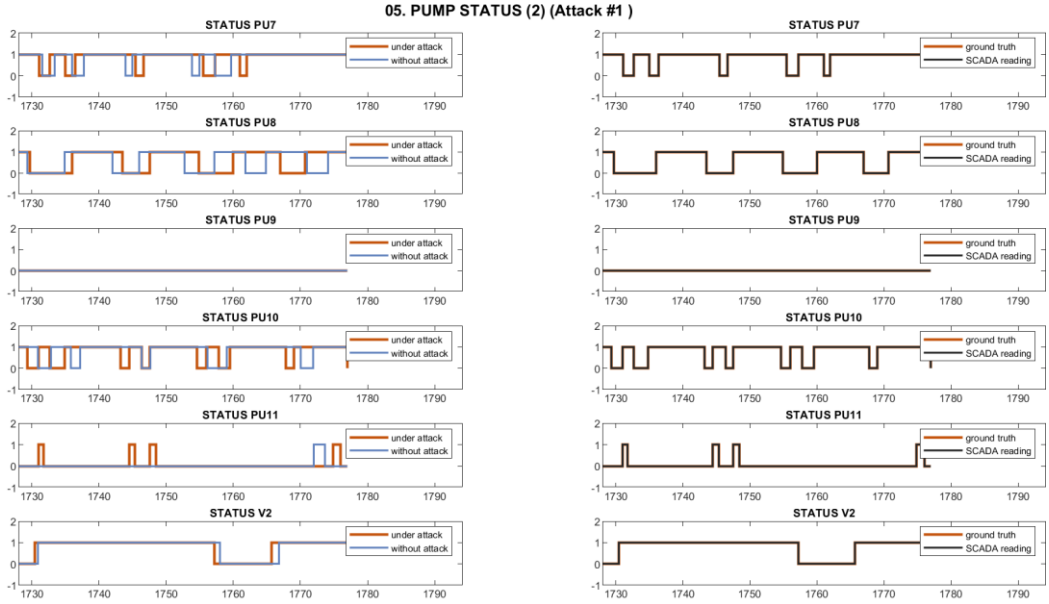


Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.
SCADA concealment: Replay attack (L_T7).



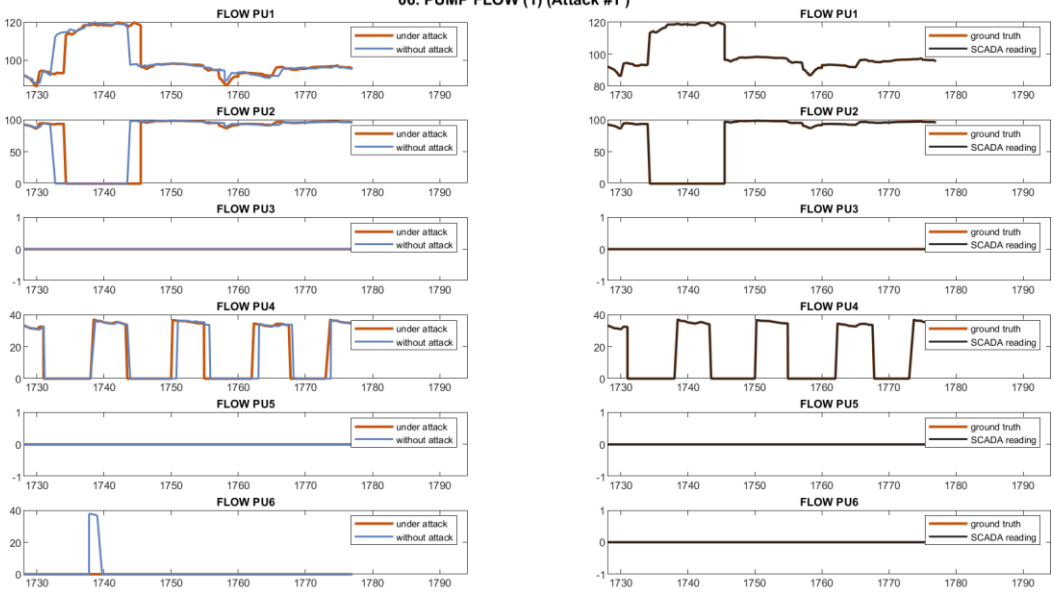
Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.
SCADA concealment: Replay attack (L_T7).

05. PUMP STATUS (2) (Attack #1)



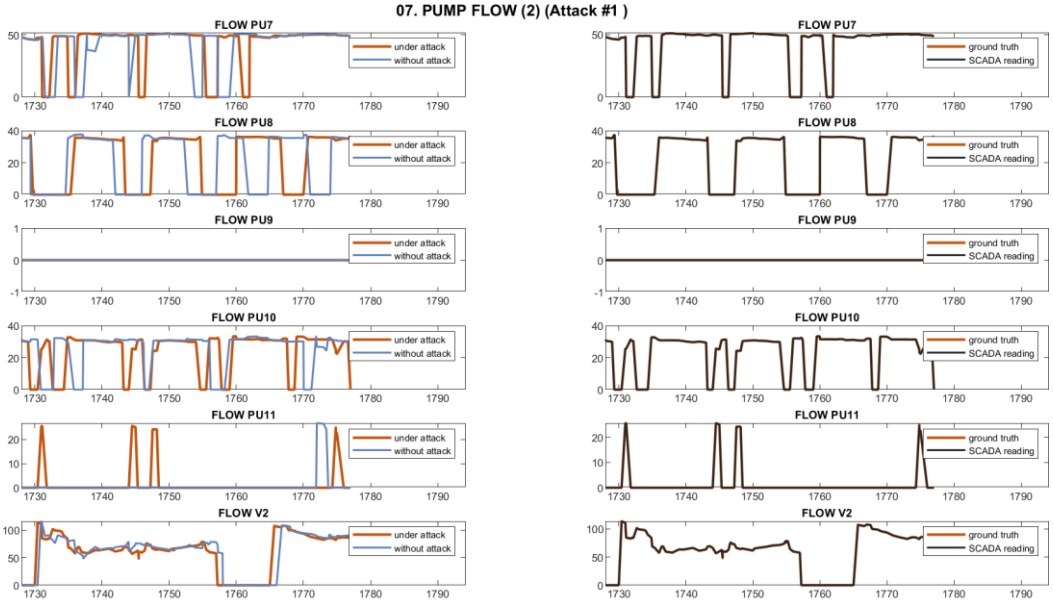
Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.
SCADA concealment: Replay attack (L_T7).

06. PUMP FLOW (1) (Attack #1)



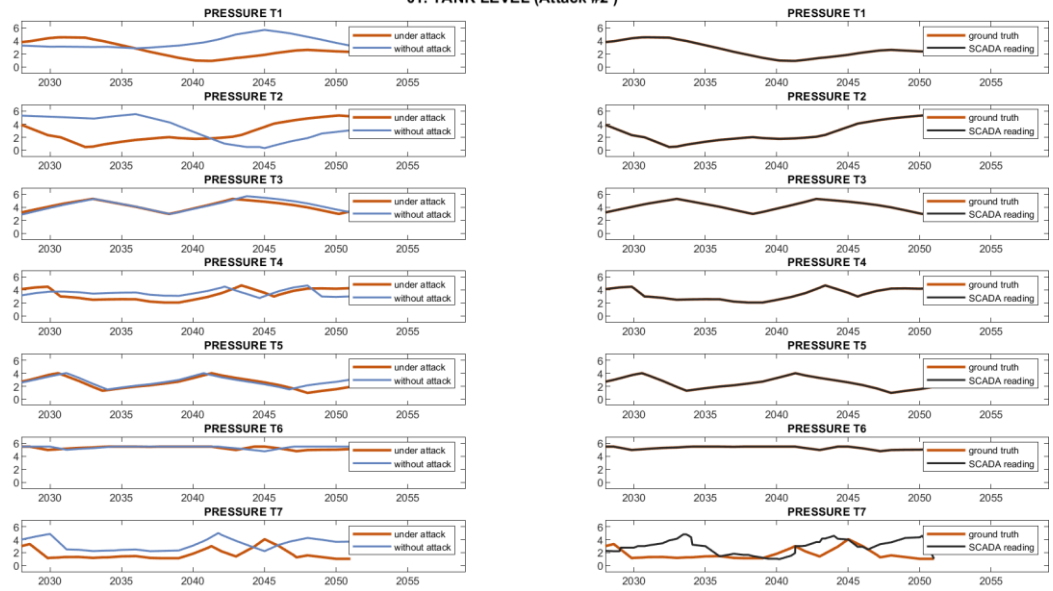
Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.
SCADA concealment: Replay attack (L_T7).

07. PUMP FLOW (2) (Attack #1)



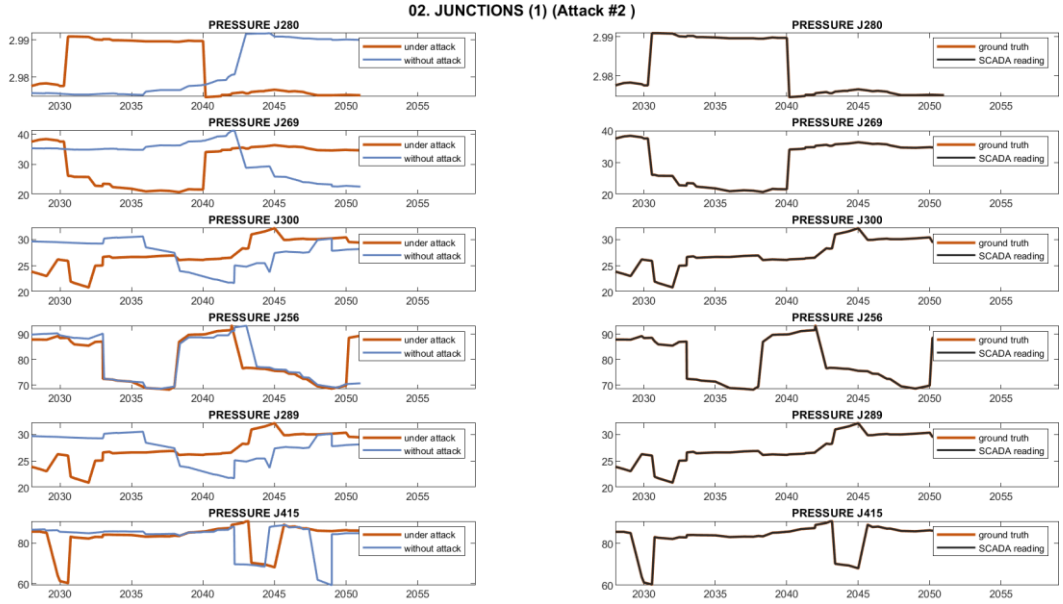
Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by altering SCADA transmission to PLC5. This causes low levels in T7.
SCADA concealment: Replay attack (L_T7).

01. TANK LEVEL (Attack #2)



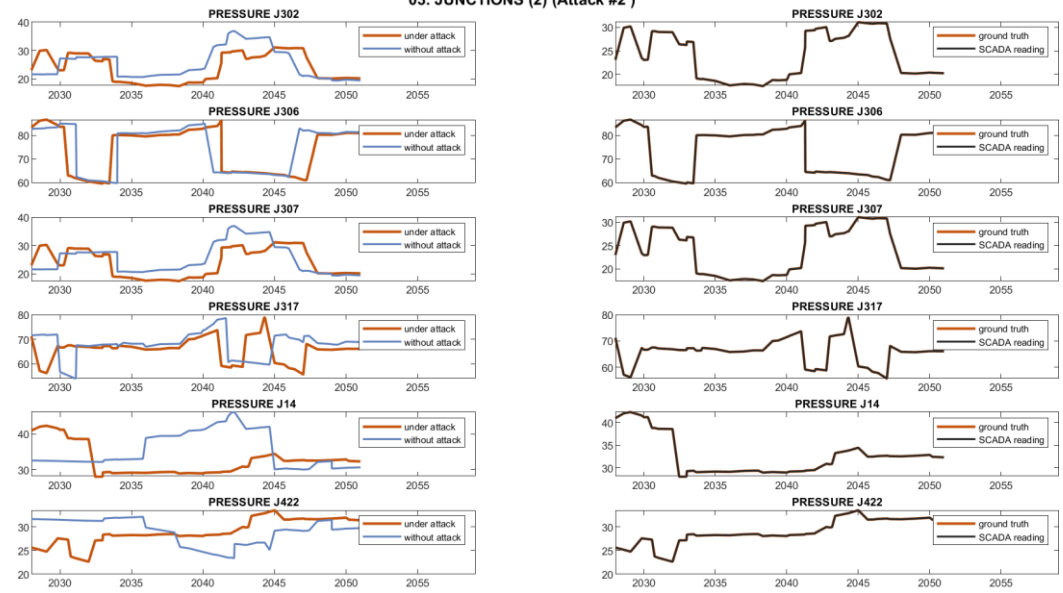
Attack: Like Attack 1.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).

02. JUNCTIONS (1) (Attack #2)



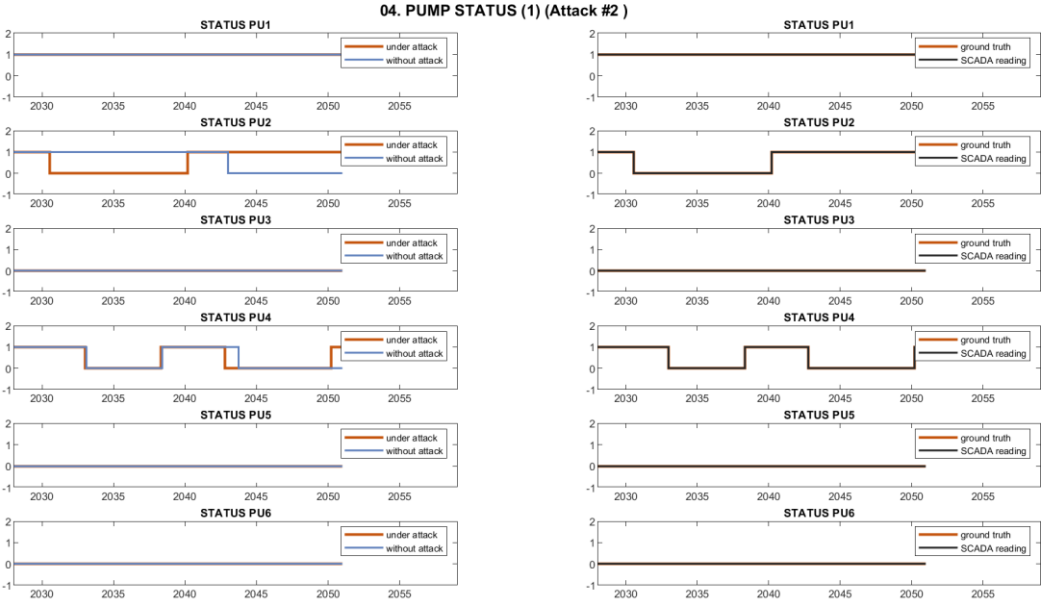
Attack: Like Attack 1.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).

03. JUNCTIONS (2) (Attack #2)



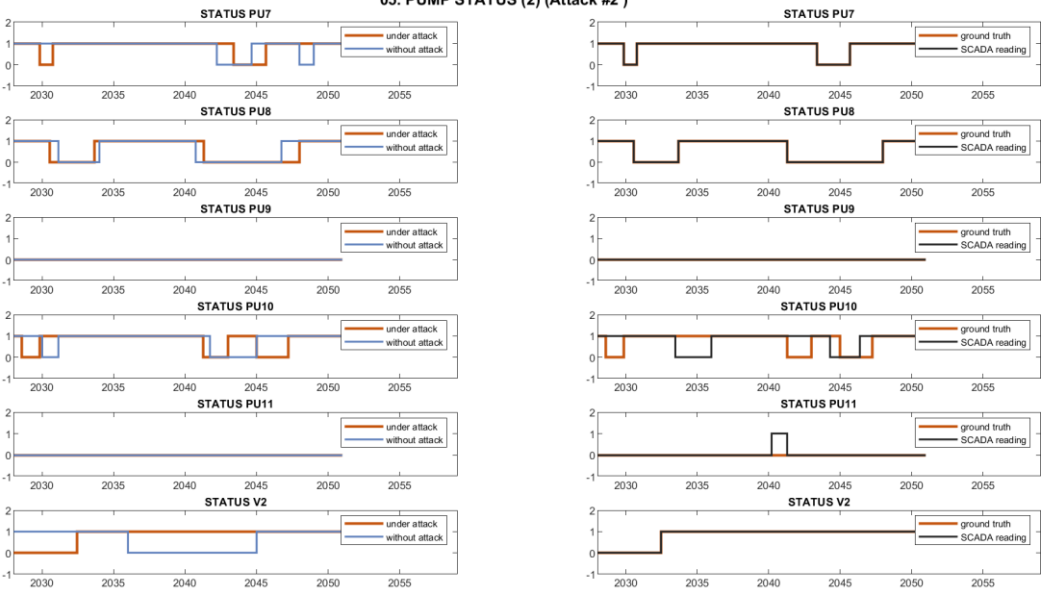
Attack: Like Attack 1.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).

04. PUMP STATUS (1) (Attack #2)



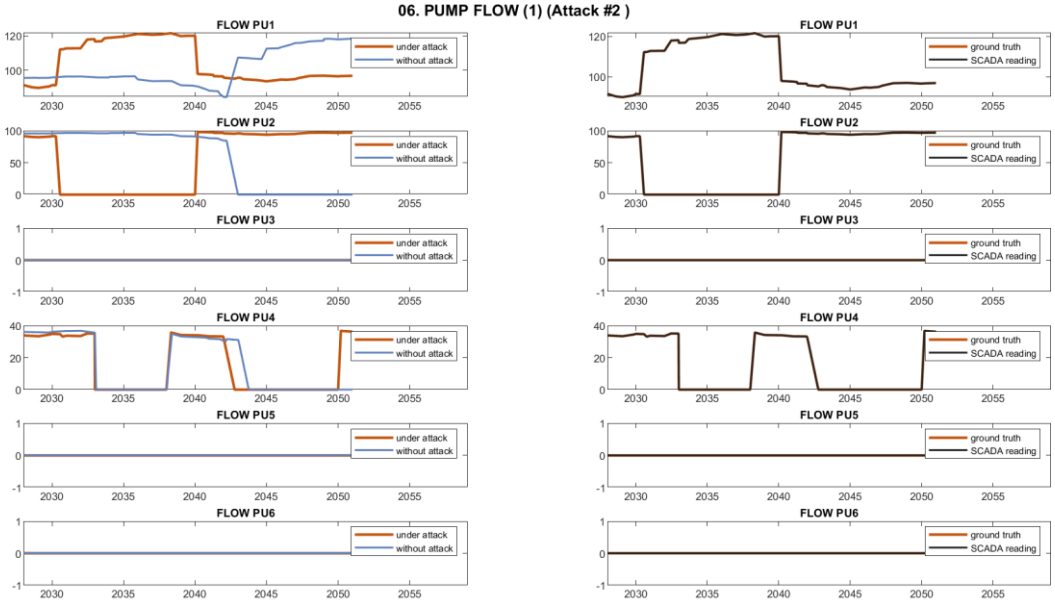
Attack: Like Attack 1.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).

05. PUMP STATUS (2) (Attack #2)



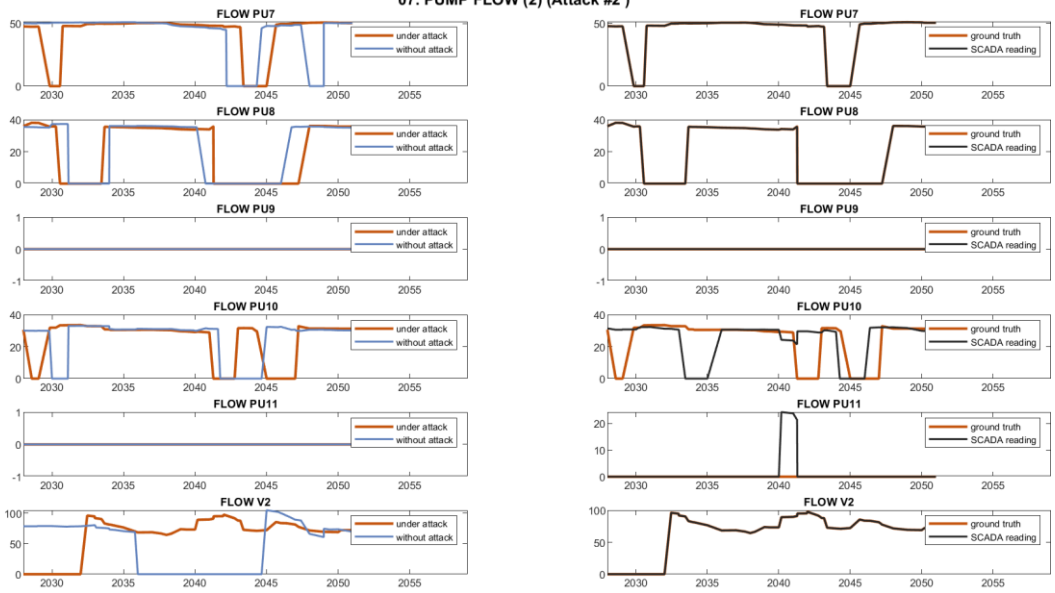
Attack: Like Attack 1.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).

06. PUMP FLOW (1) (Attack #2)

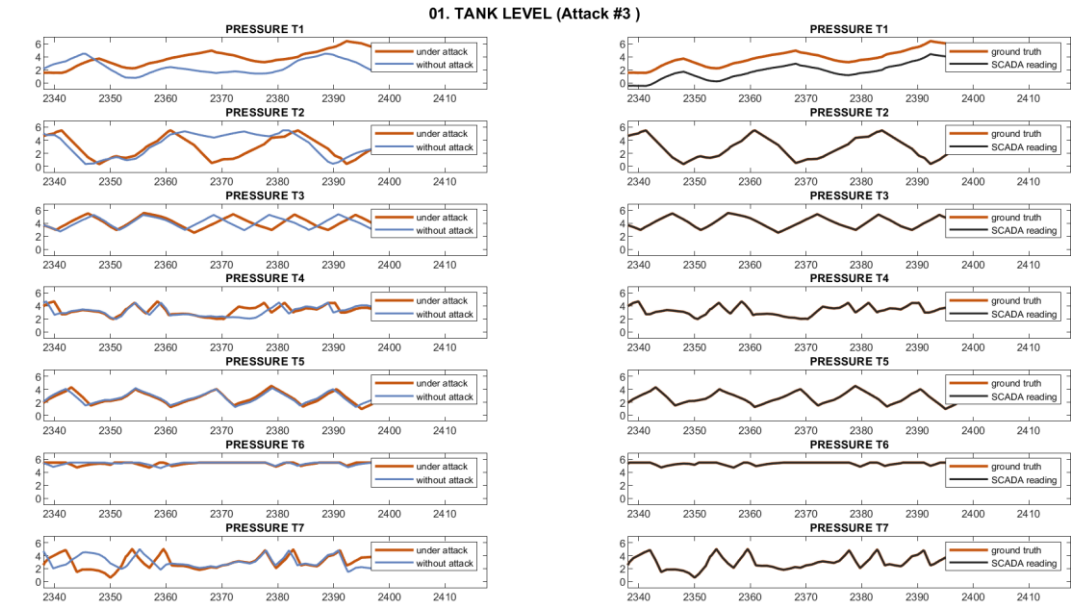


Attack: Like Attack 1.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).

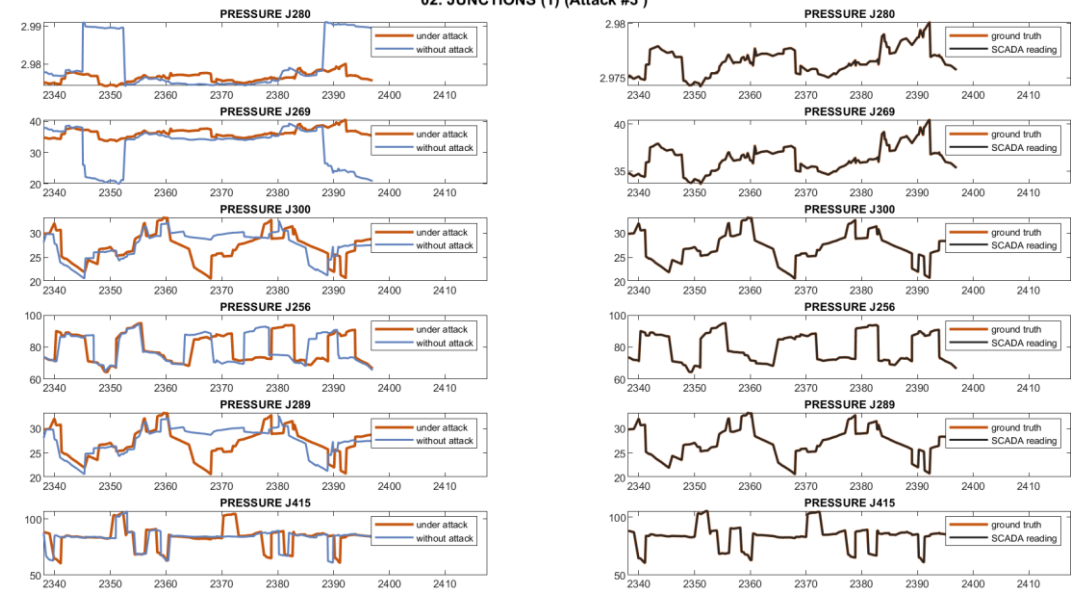
07. PUMP FLOW (2) (Attack #2)



Attack: Like Attack 1.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11).

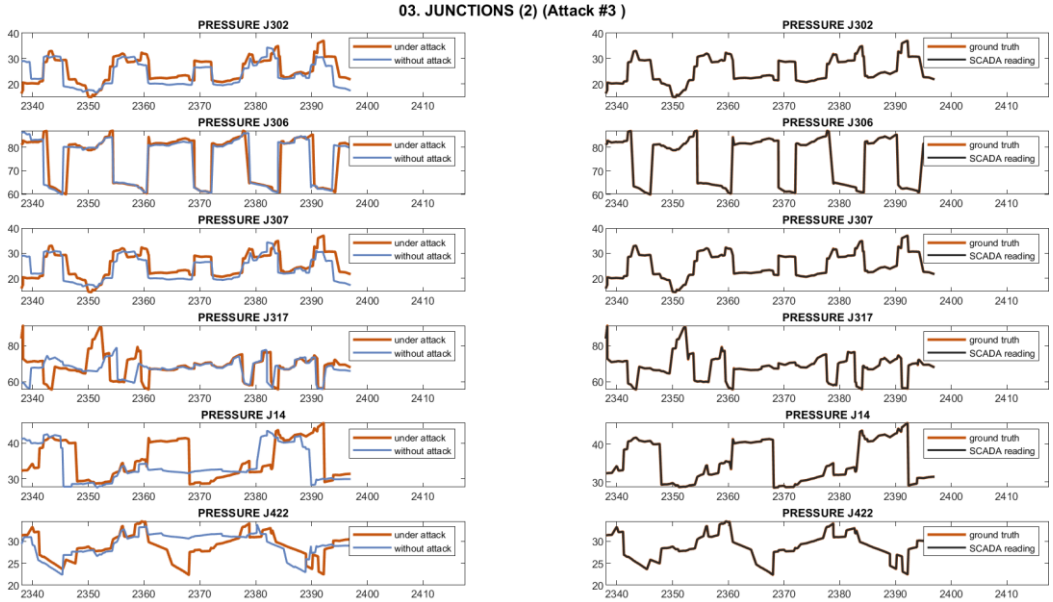


Attack: Attacker alters L_T1 readings arriving to PLC2 with a constant low level. PLC1 receives the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.
SCADA concealment: Polyline to offset L_T1 increase.



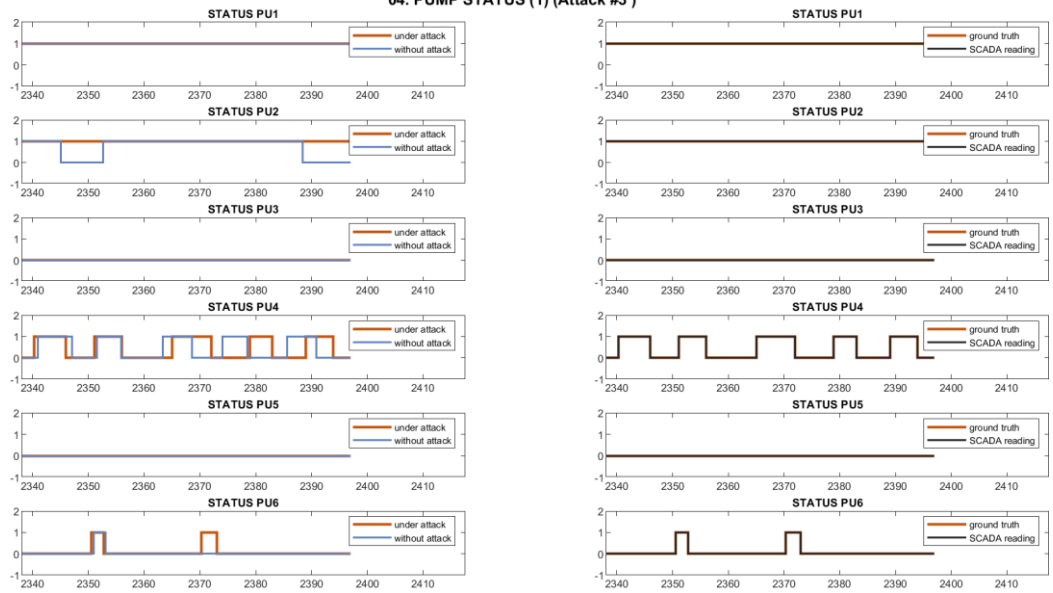
Attack: Attacker alters L_T1 readings arriving to PLC2 with a constant low level. PLC1 receives the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.
SCADA concealment: Polyline to offset L_T1 increase.

03. JUNCTIONS (2) (Attack #3)

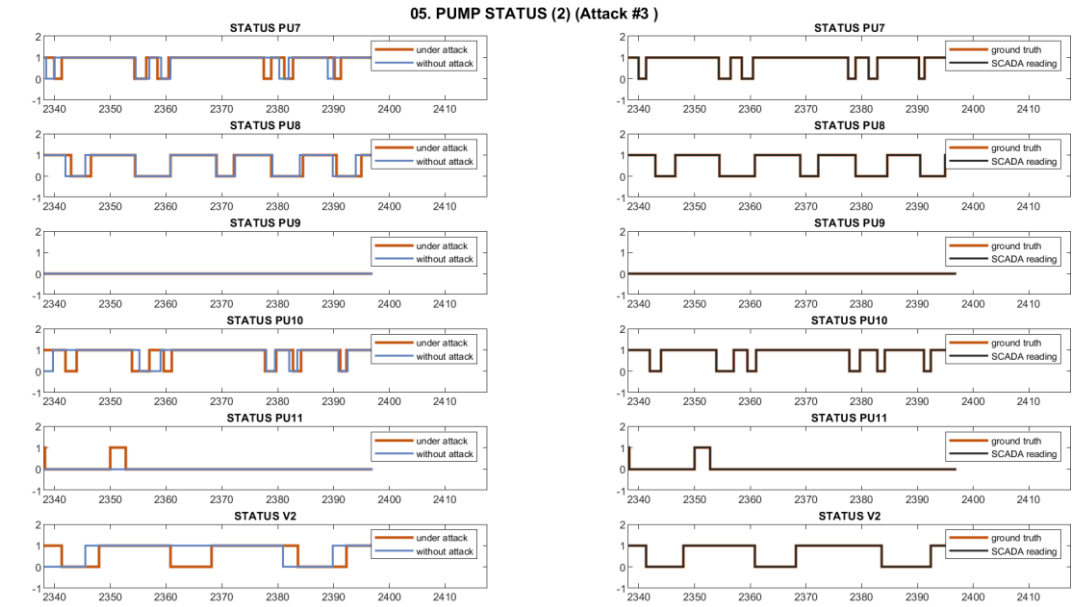


Attack: Attacker alters L_T1 readings arriving to PLC2 with a constant low level. PLC1 receives the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.
SCADA concealment: Polyline to offset L_T1 increase.

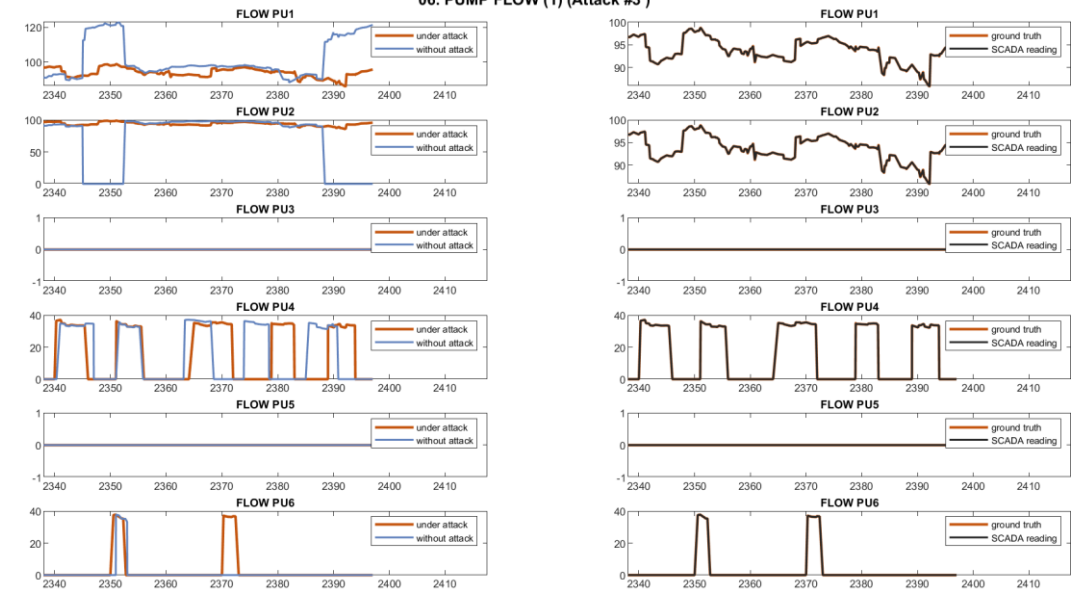
04. PUMP STATUS (1) (Attack #3)



Attack: Attacker alters L_T1 readings arriving to PLC2 with a constant low level. PLC1 receives the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.
SCADA concealment: Polyline to offset L_T1 increase.

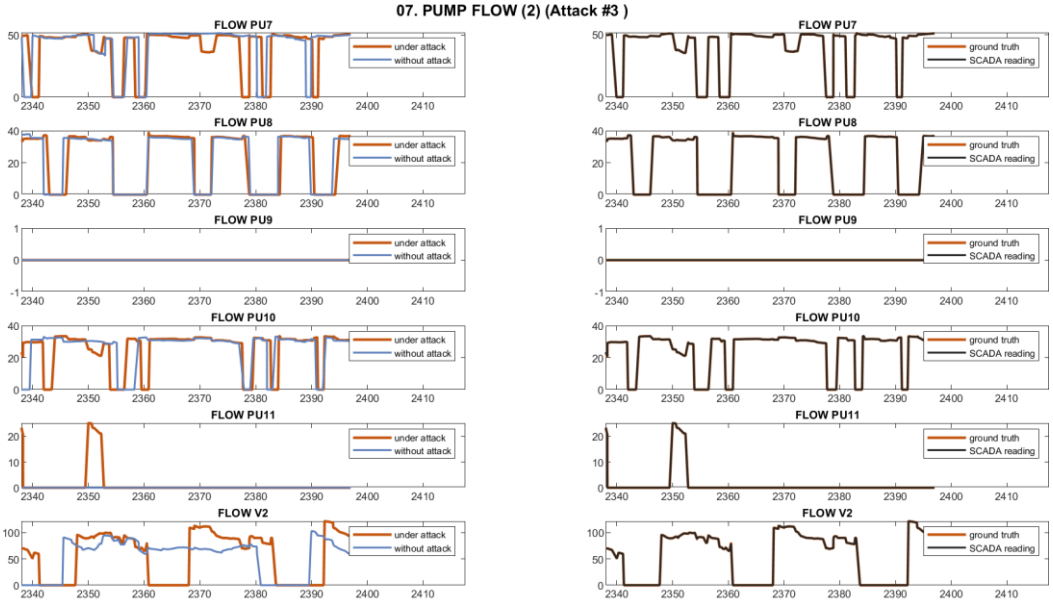


Attack: Attacker alters L_T1 readings arriving to PLC2 with a constant low level. PLC1 receives the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.
SCADA concealment: Polyline to offset L_T1 increase.



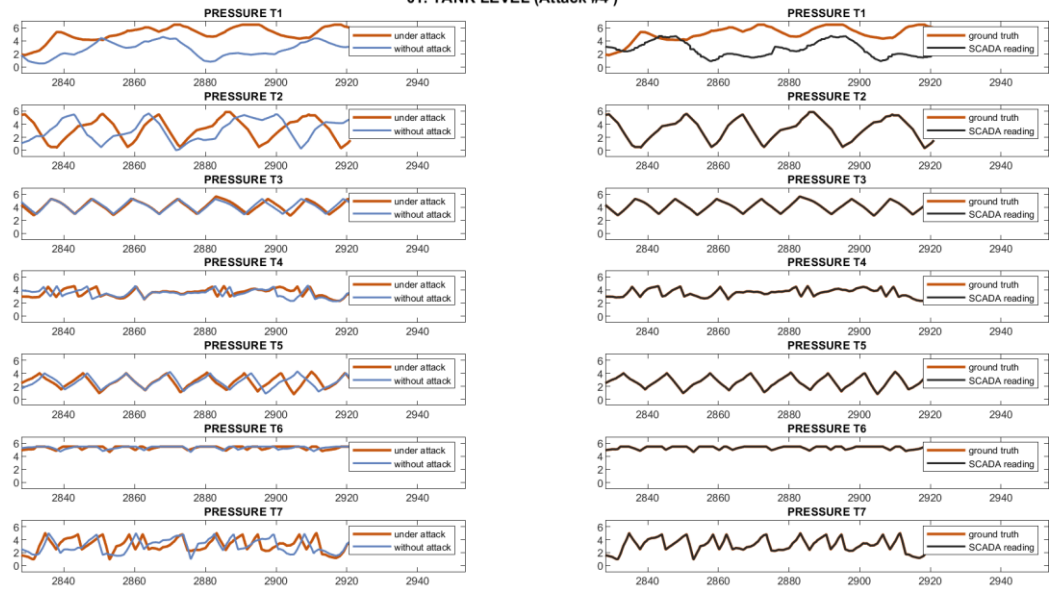
Attack: Attacker alters L_T1 readings arriving to PLC2 with a constant low level. PLC1 receives the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.
SCADA concealment: Polyline to offset L_T1 increase.

07. PUMP FLOW (2) (Attack #3)

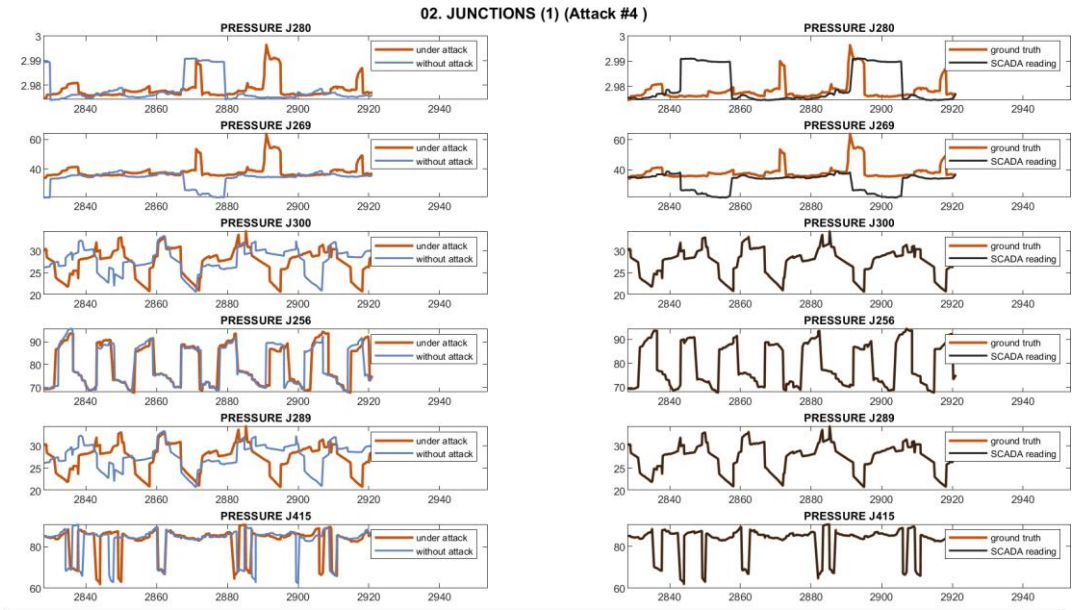


Attack: Attacker alters L_T1 readings arriving to PLC2 with a constant low level. PLC1 receives the manipulated readings from PLC2 and keeps Pumps PU1 and PU2 on, driving T1 to overflow.
SCADA concealment: Polyline to offset L_T1 increase.

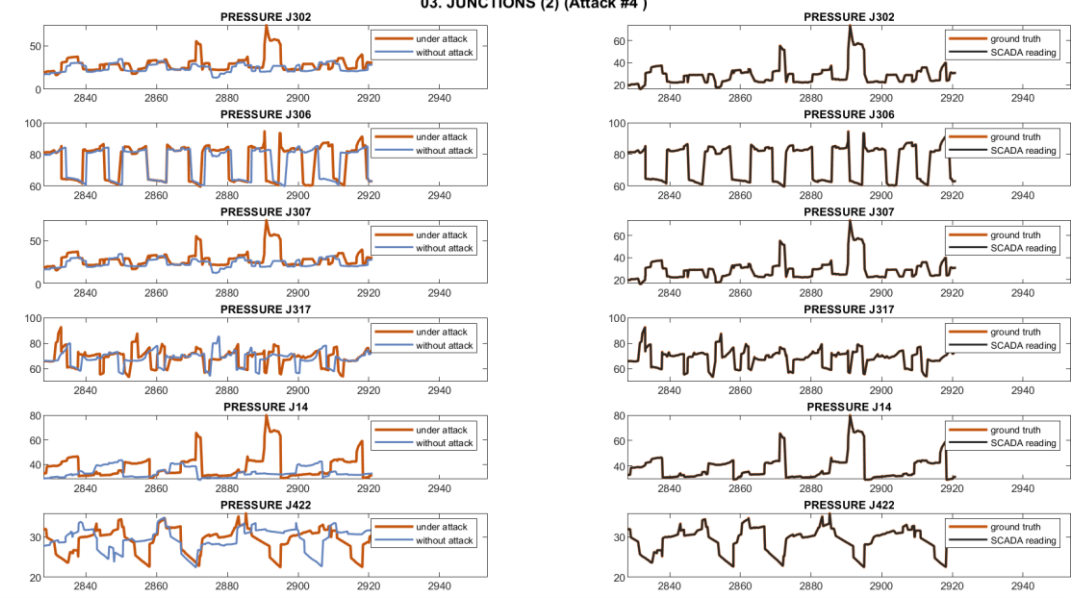
01. TANK LEVEL (Attack #4)



Attack: As in Attack 3.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).

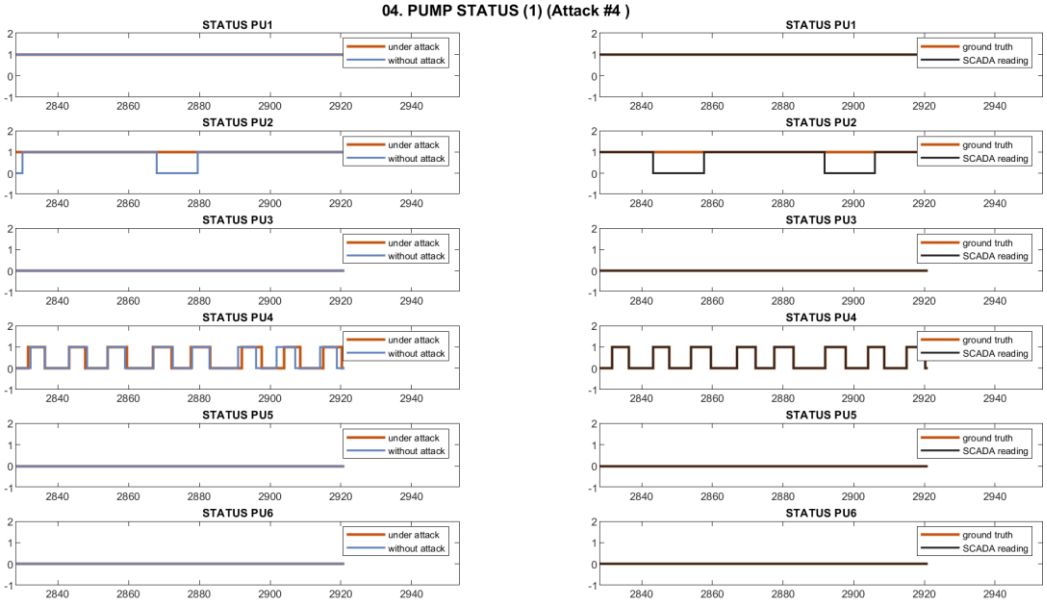


Attack: As in Attack 3.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).



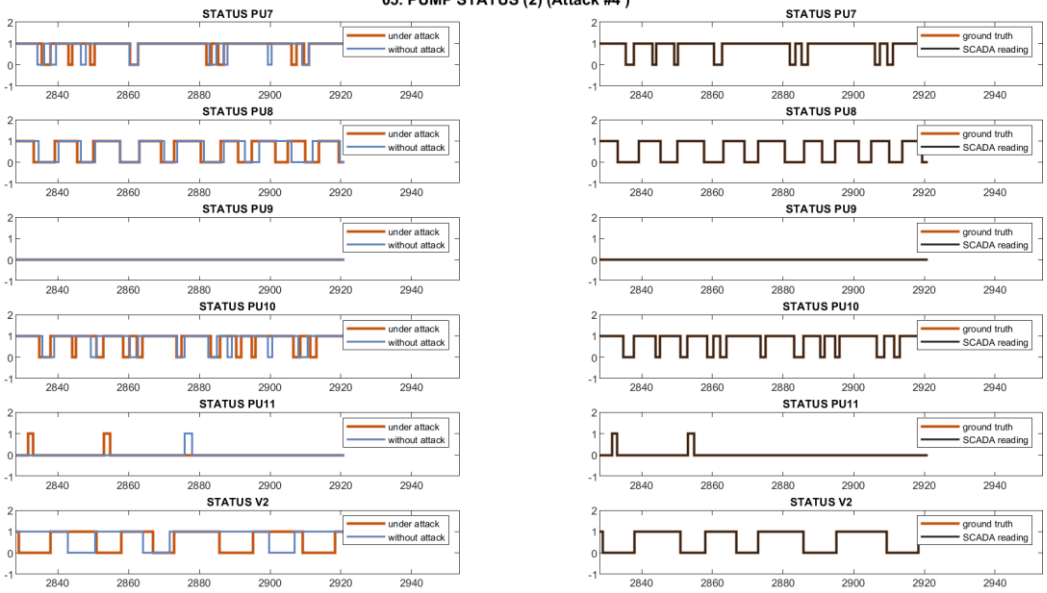
Attack: As in Attack 3.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).

04. PUMP STATUS (1) (Attack #4)



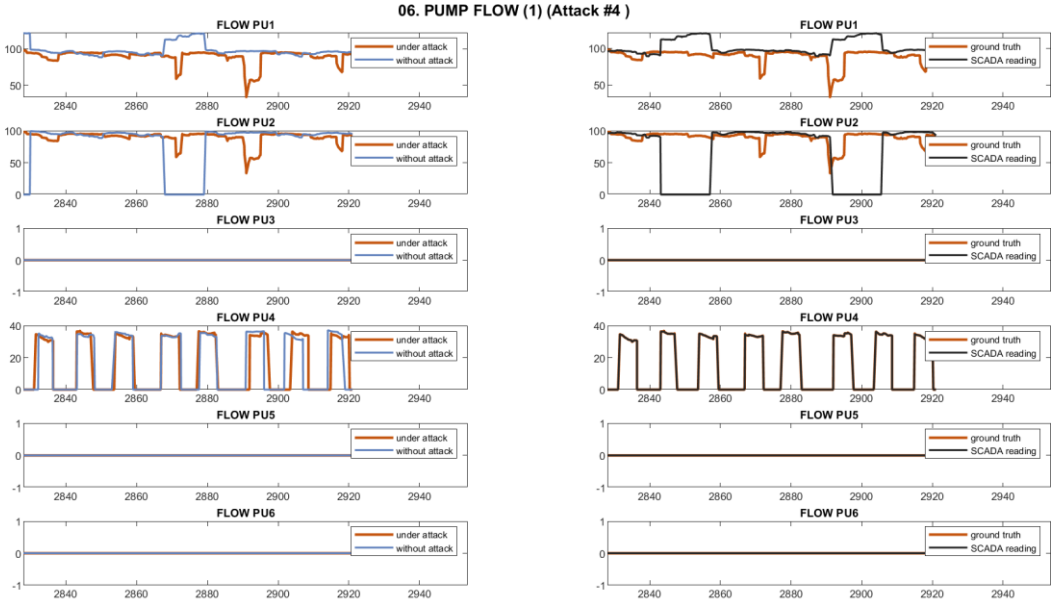
Attack: As in Attack 3.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).

05. PUMP STATUS (2) (Attack #4)



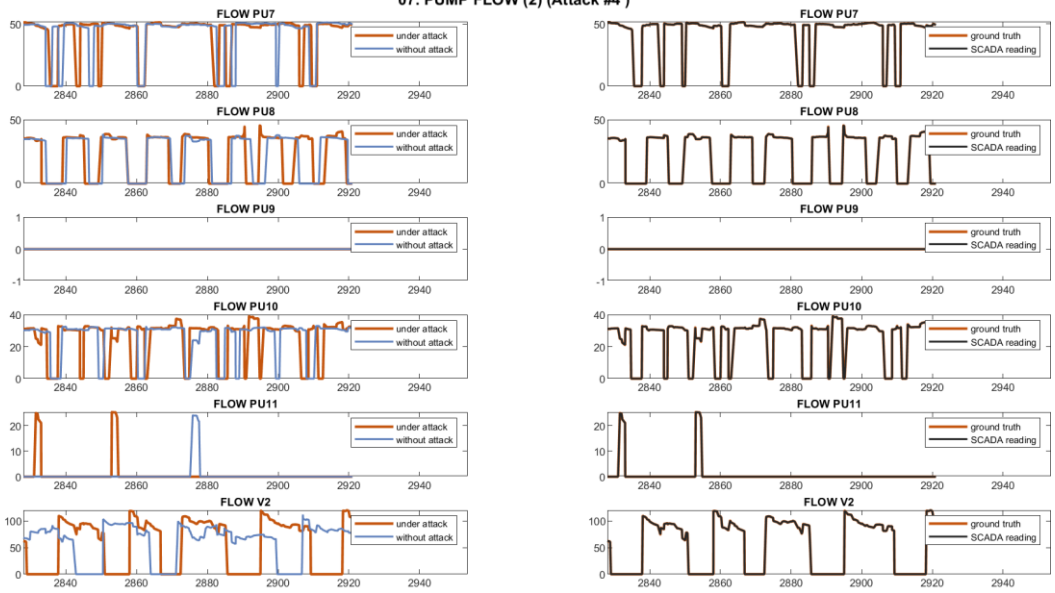
Attack: As in Attack 3.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).

06. PUMP FLOW (1) (Attack #4)



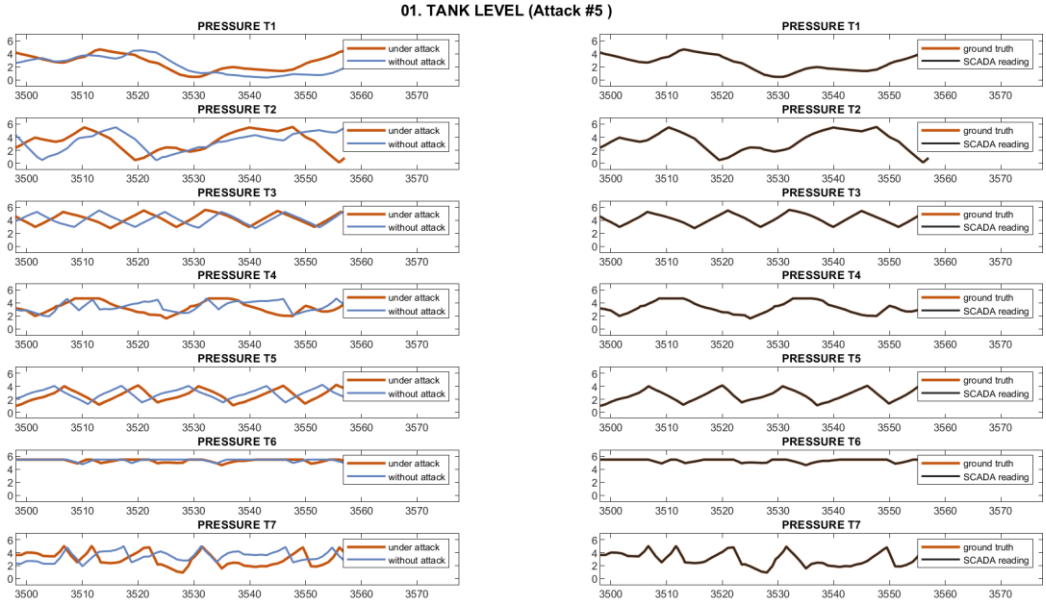
Attack: As in Attack 3.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).

07. PUMP FLOW (2) (Attack #4)



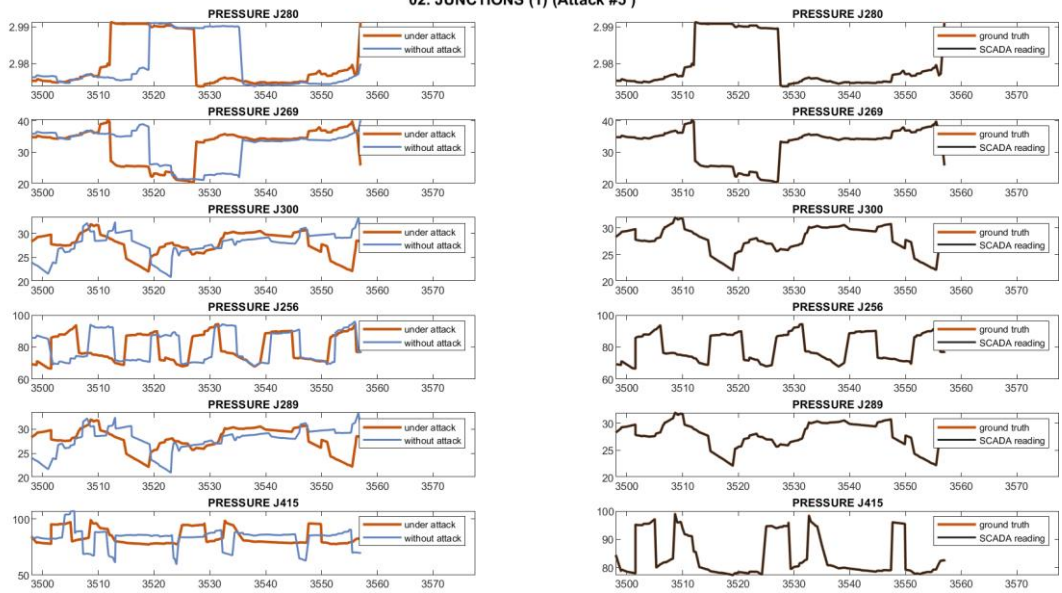
Attack: As in Attack 3.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2, P_J269, P_J280).

01. TANK LEVEL (Attack #5)

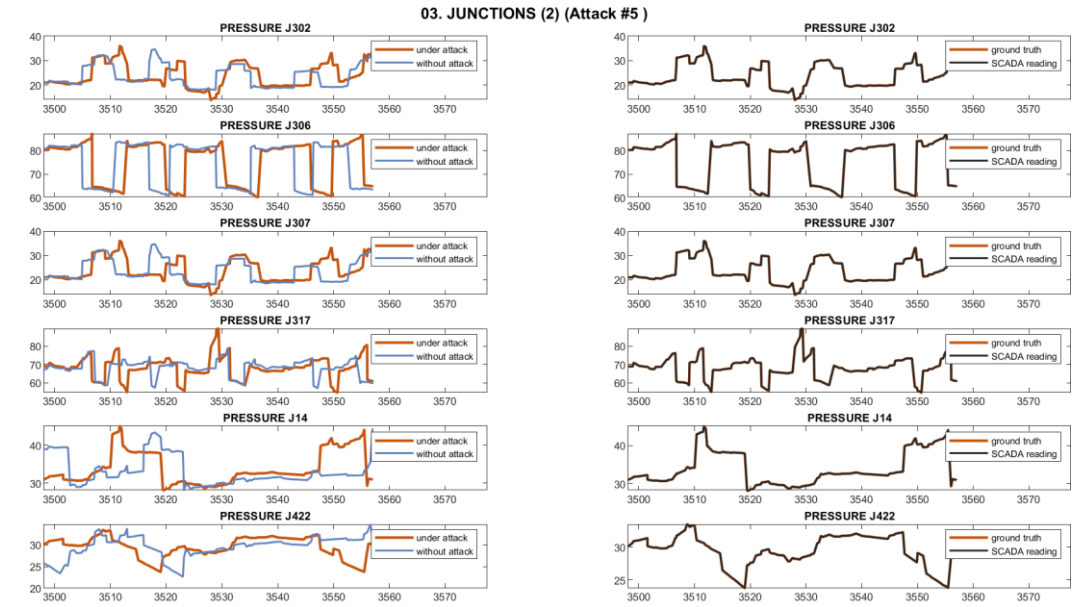


Attack: Working speed of PU7 reduces to 0.9 of nominal speed caused lower water levels in T4
SCADA concealment: -

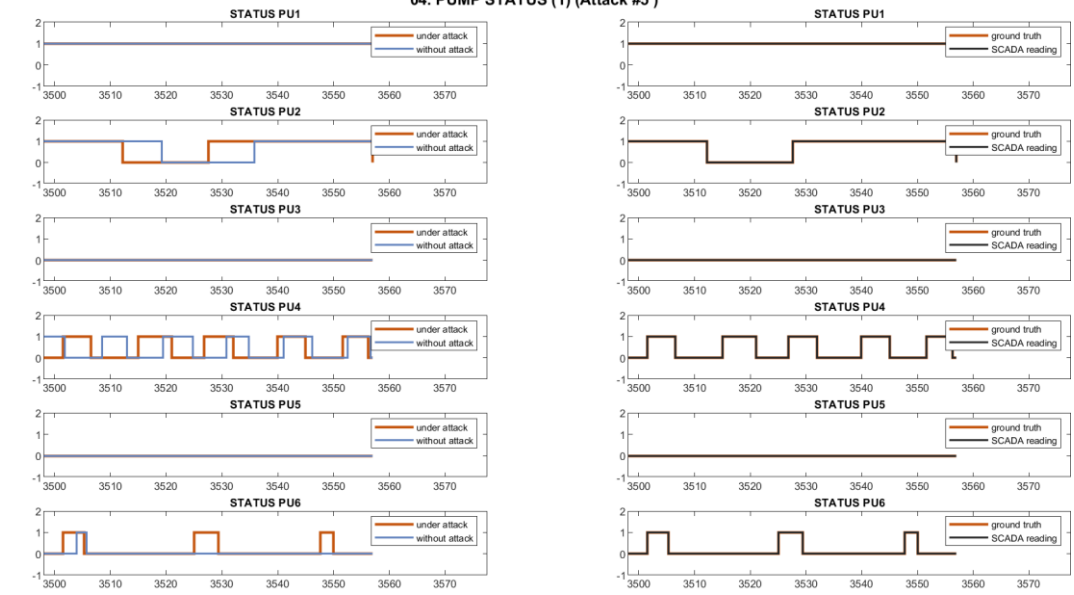
02. JUNCTIONS (1) (Attack #5)



Attack: Working speed of PU7 reduces to 0.9 of nominal speed caused lower water levels in T4
SCADA concealment: -

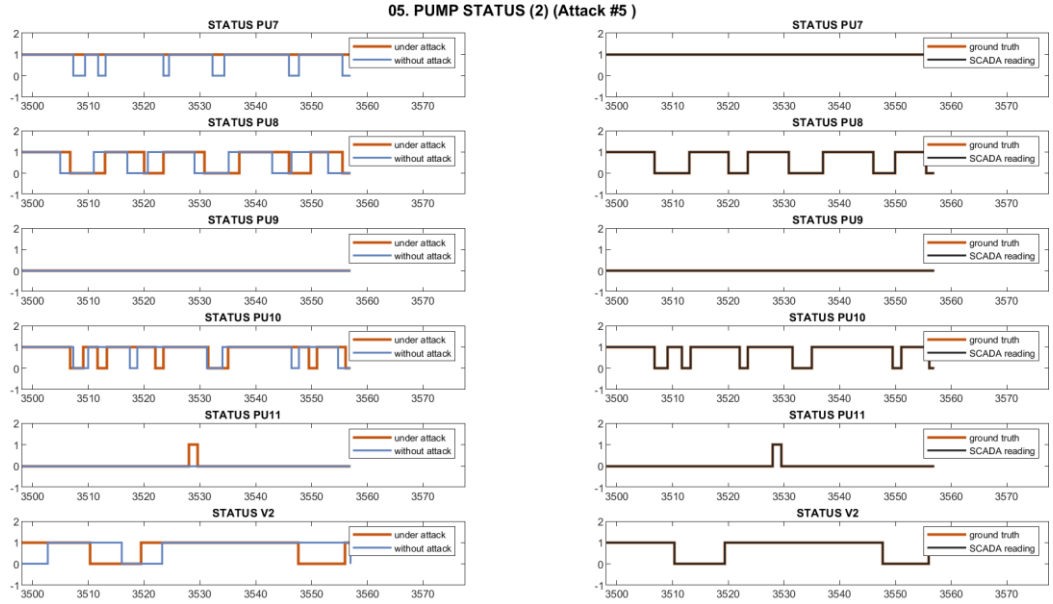


Attack: Working speed of PU7 reduces to 0.9 of nominal speed caused lower water levels in T4
SCADA concealment: -



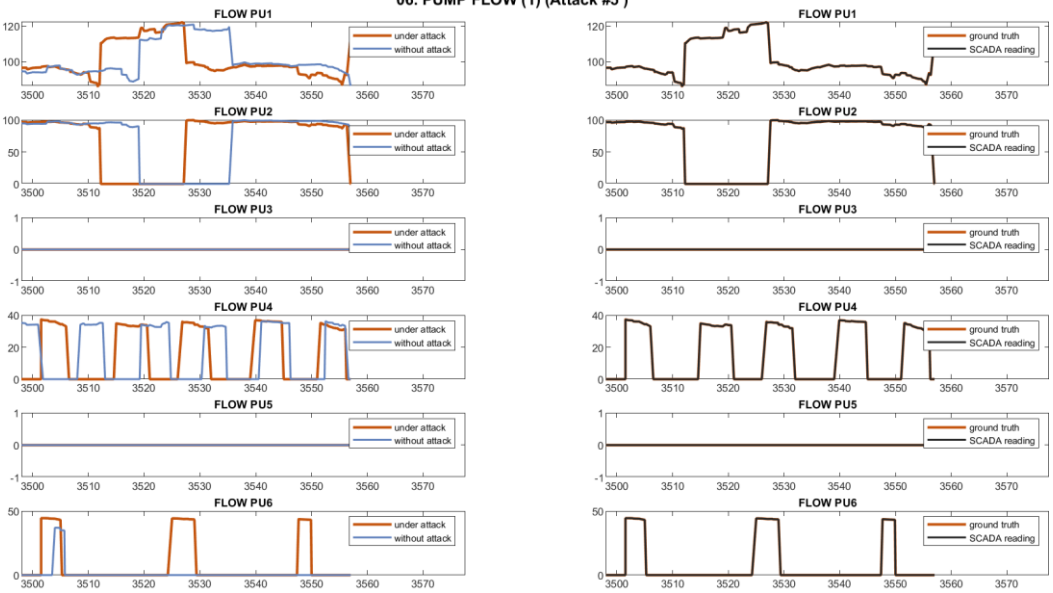
Attack: Working speed of PU7 reduces to 0.9 of nominal speed caused lower water levels in T4
SCADA concealment: -

05. PUMP STATUS (2) (Attack #5)



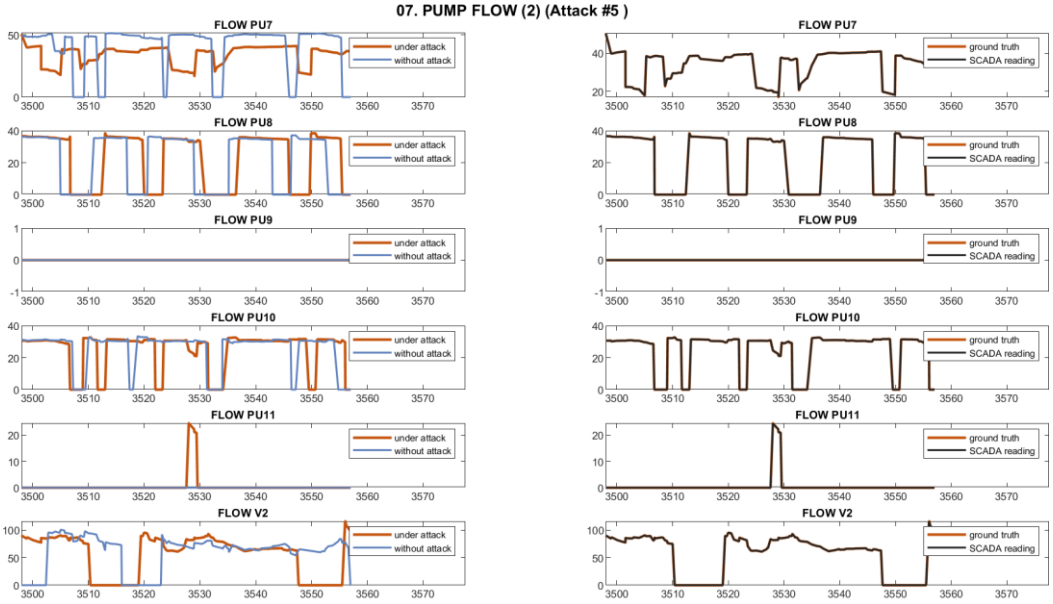
Attack: Working speed of PU7 reduces to 0.9 of nominal speed caused lower water levels in T4
SCADA concealment: -

06. PUMP FLOW (1) (Attack #5)



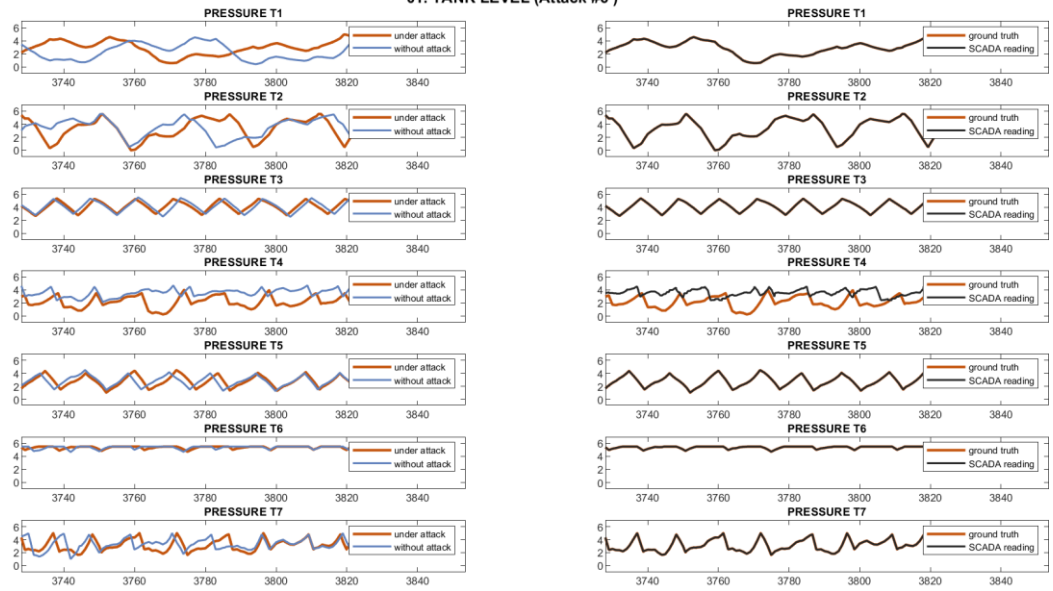
Attack: Working speed of PU7 reduces to 0.9 of nominal speed caused lower water levels in T4
SCADA concealment: -

07. PUMP FLOW (2) (Attack #5)

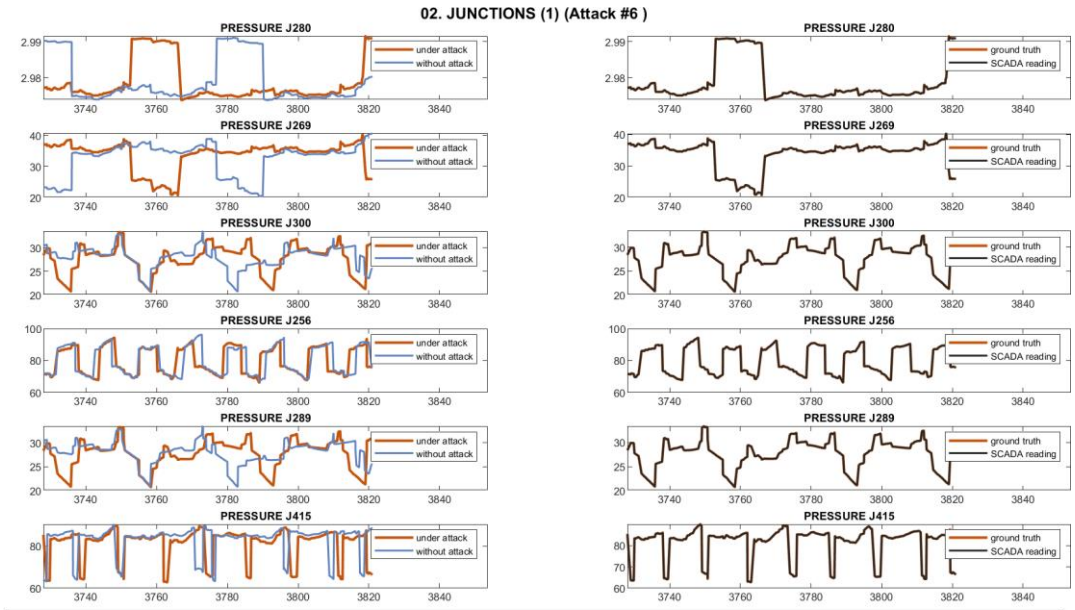


Attack: Working speed of PU7 reduces to 0.9 of nominal speed caused lower water levels in T4
SCADA concealment: -

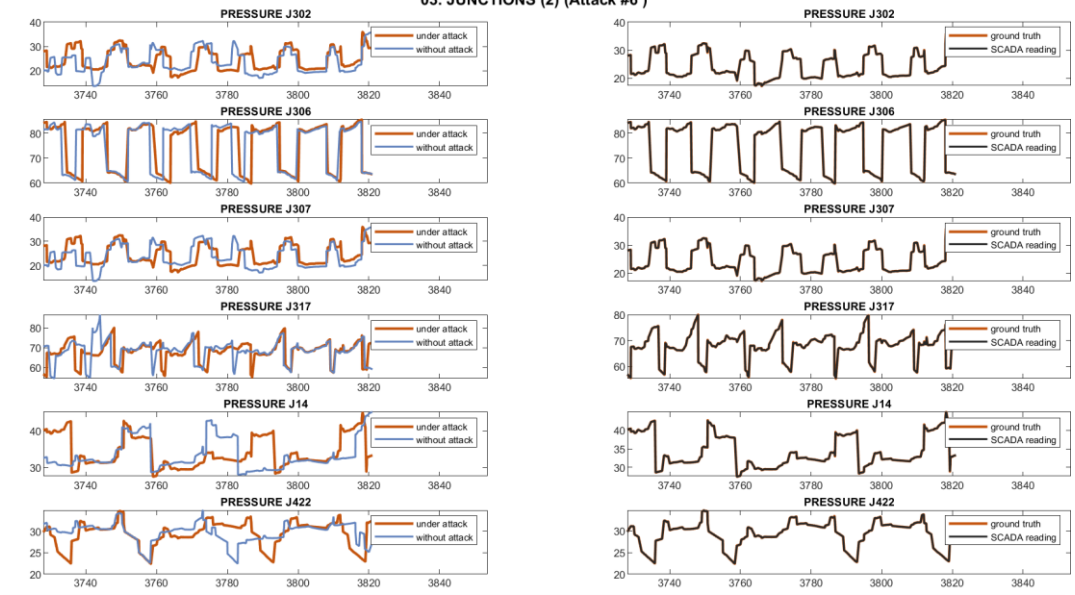
01. TANK LEVEL (Attack #6)



Attack: As in Attack 5, but speed reduced to 0.7.
SCADA concealment: Replay attack (L_T4).

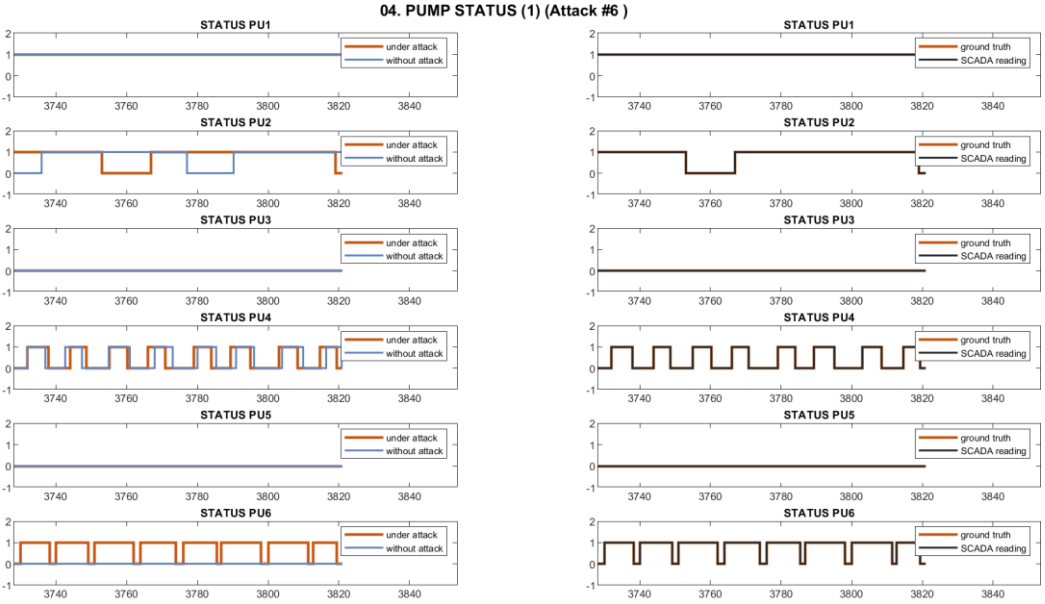


Attack: As in Attack 5, but speed reduced to 0.7.
SCADA concealment: Replay attack (L_T4).



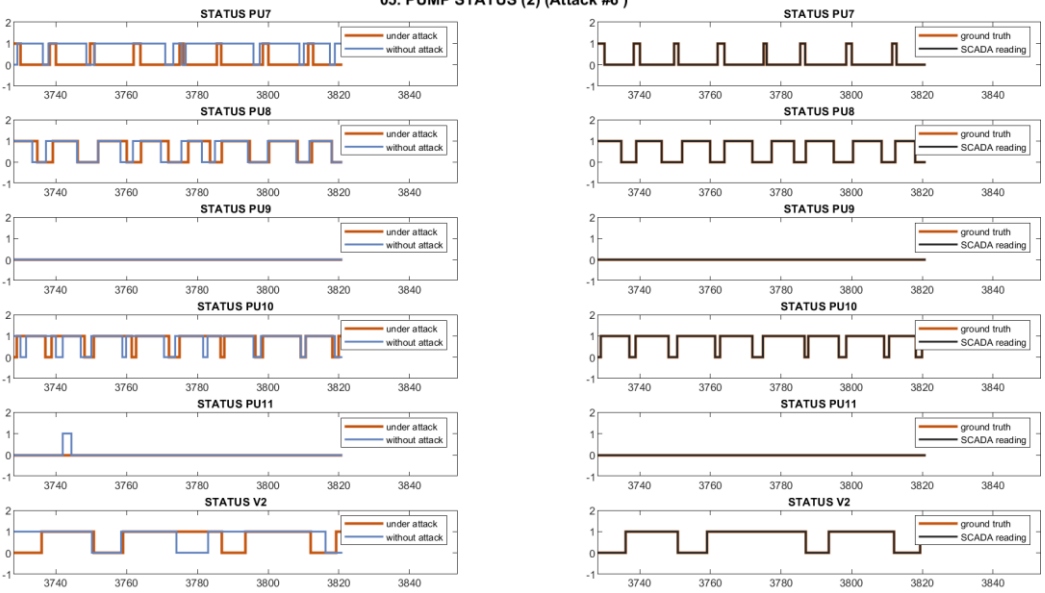
Attack: As in Attack 5, but speed reduced to 0.7.
SCADA concealment: Replay attack (L_T4).

04. PUMP STATUS (1) (Attack #6)



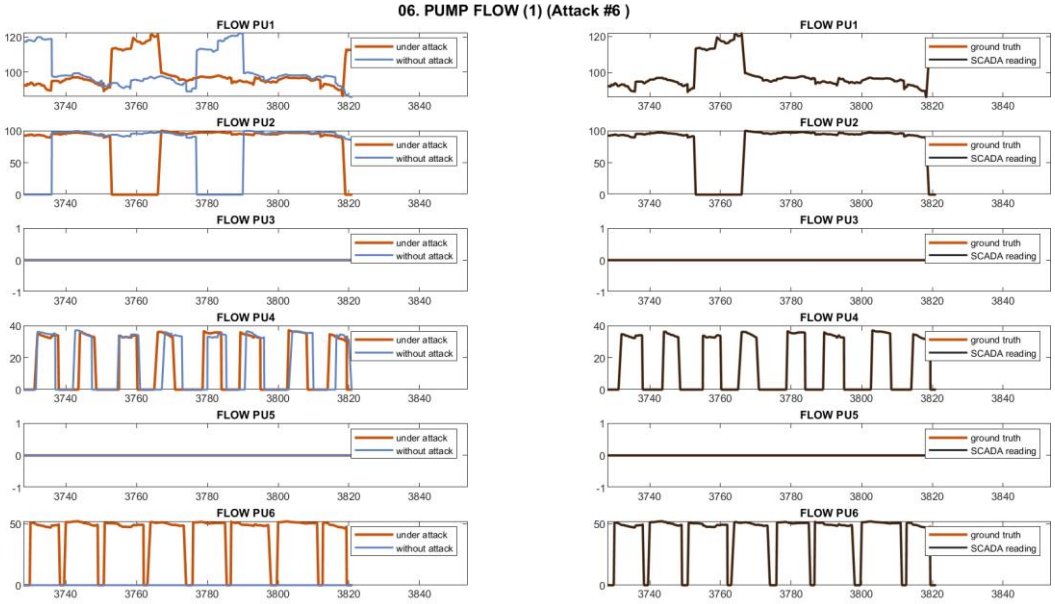
Attack: As in Attack 5, but speed reduced to 0.7.
SCADA concealment: Replay attack (L_T4).

05. PUMP STATUS (2) (Attack #6)



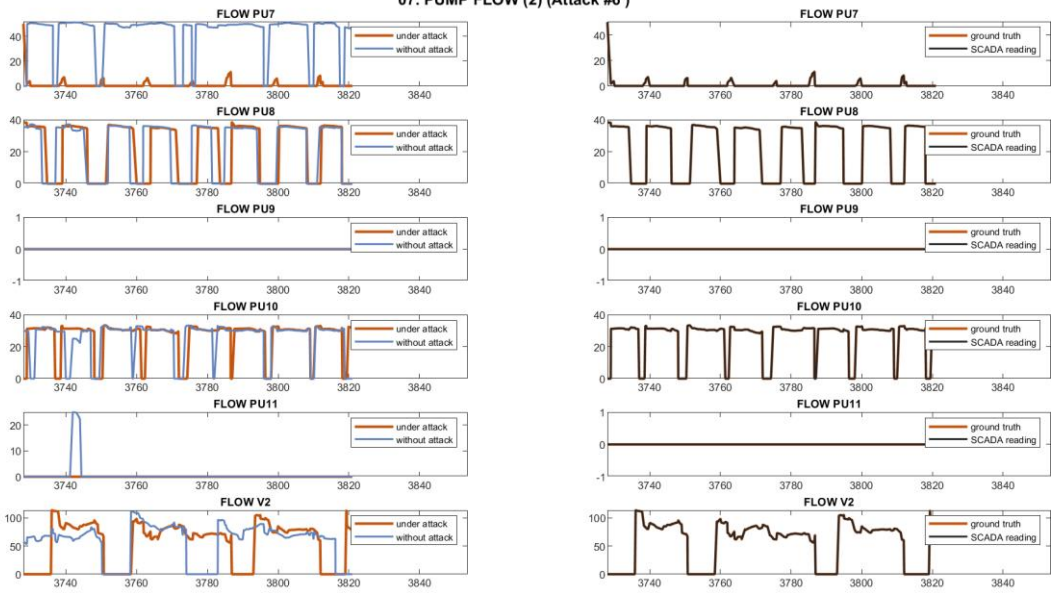
Attack: As in Attack 5, but speed reduced to 0.7.
SCADA concealment: Replay attack (L_T4).

06. PUMP FLOW (1) (Attack #6)



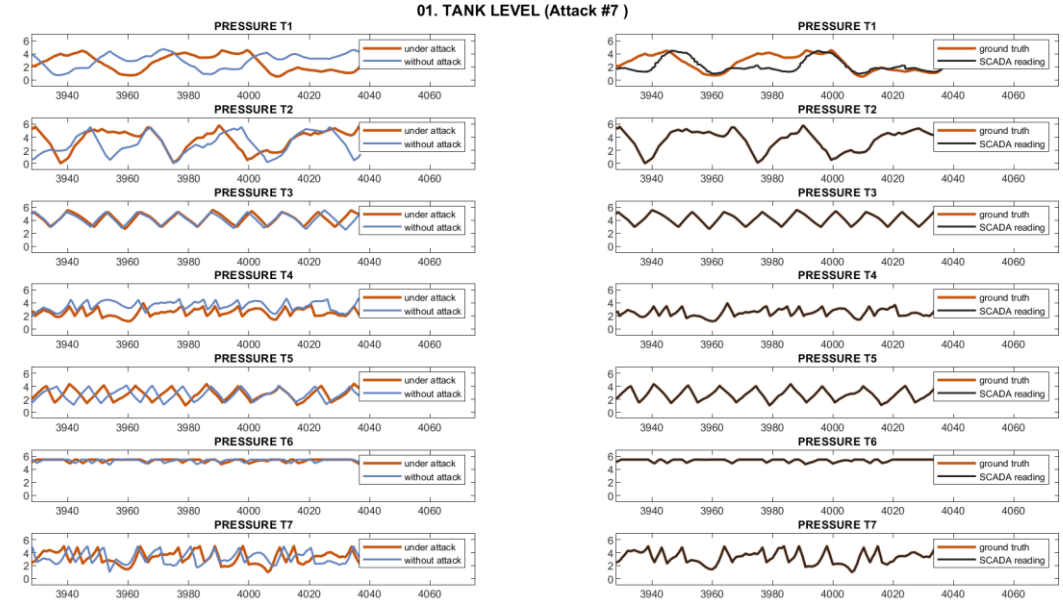
Attack: As in Attack 5, but speed reduced to 0.7.
SCADA concealment: Replay attack (L_T4).

07. PUMP FLOW (2) (Attack #6)



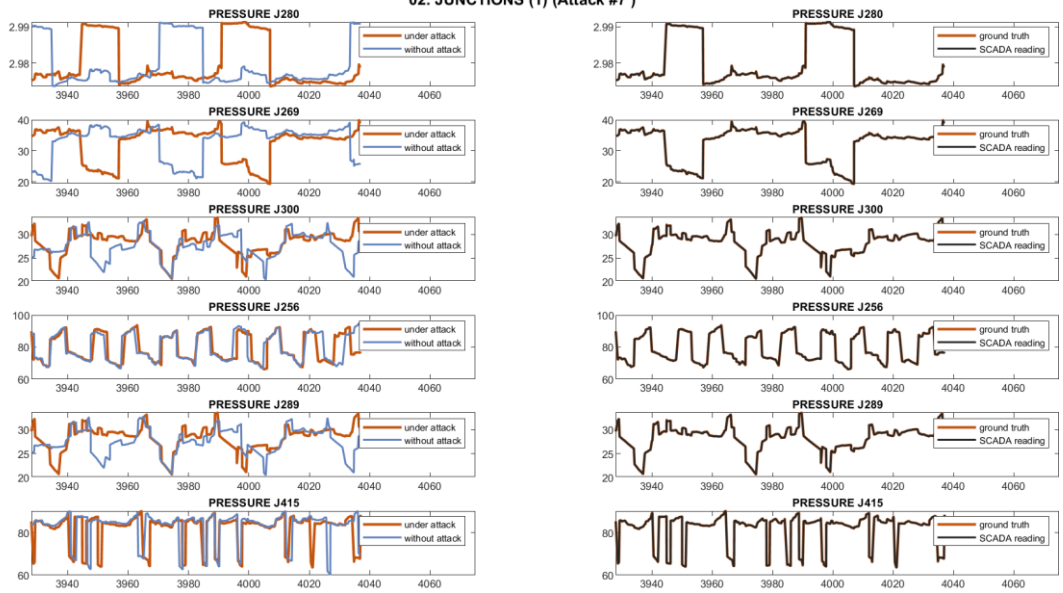
Attack: As in Attack 5, but speed reduced to 0.7.
SCADA concealment: Replay attack (L_T4).

01. TANK LEVEL (Attack #7)

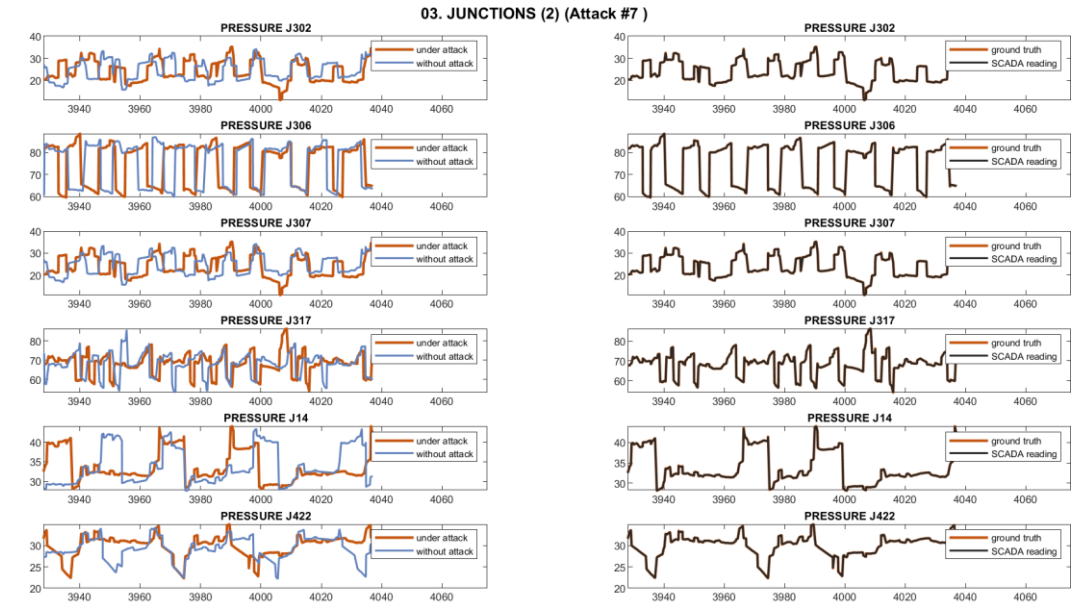


Attack: As in Attack 6.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).

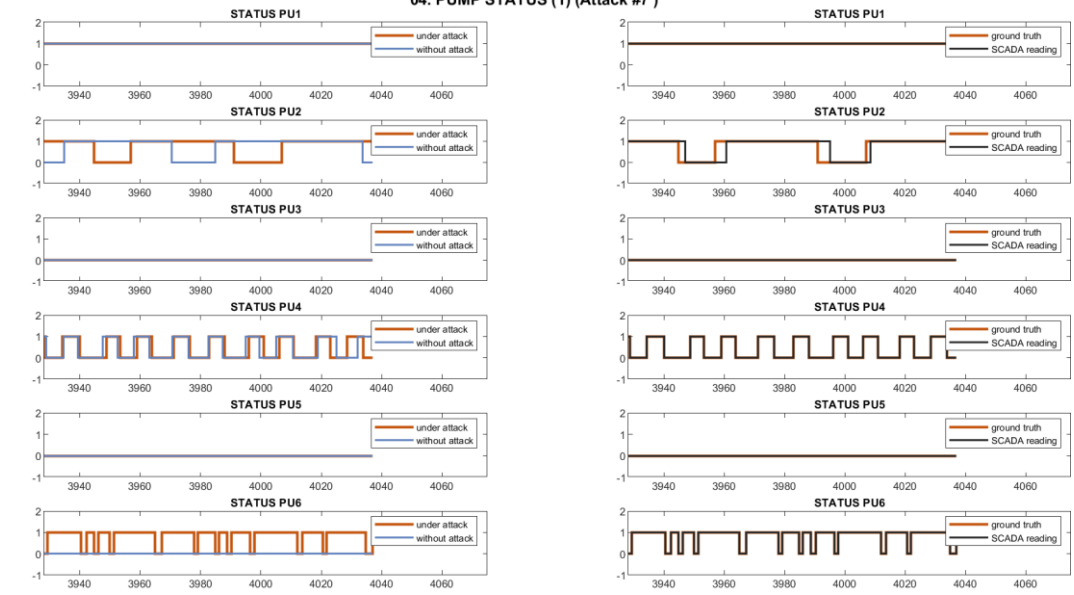
02. JUNCTIONS (1) (Attack #7)



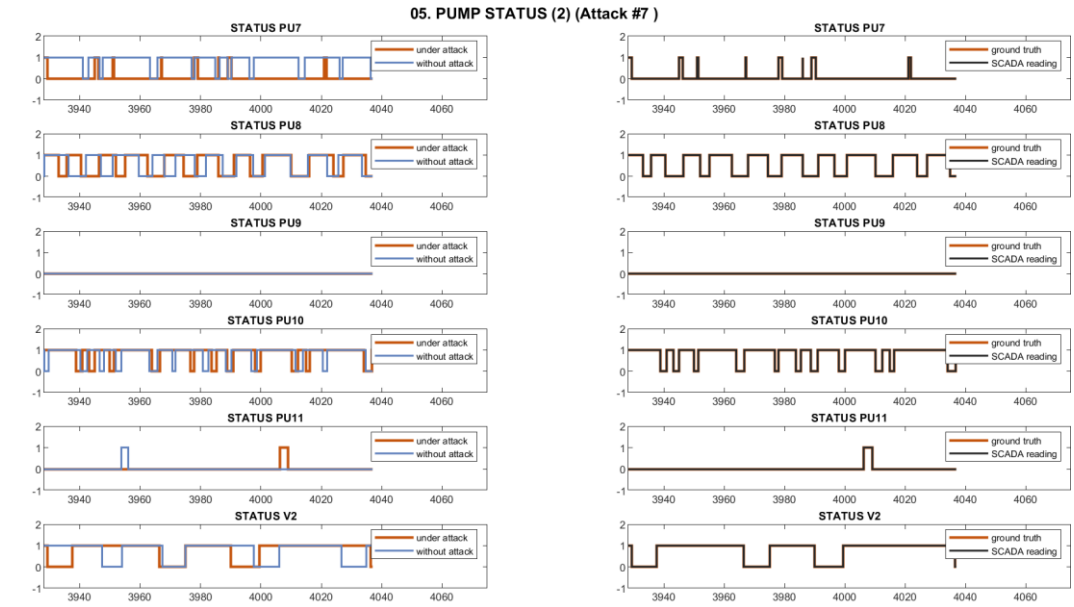
Attack: As in Attack 6.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).



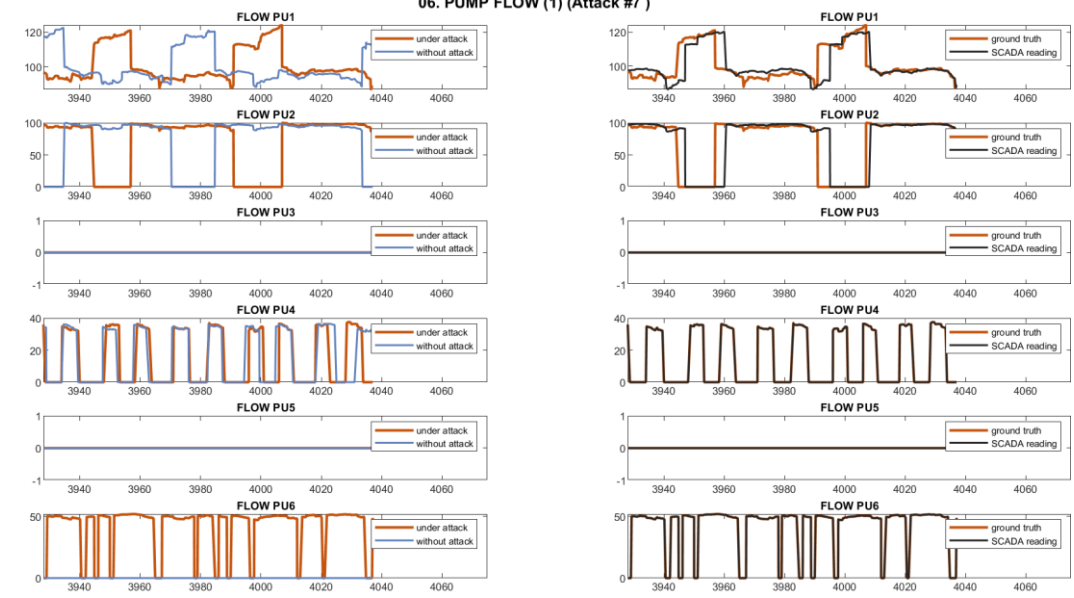
Attack: As in Attack 6.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).



Attack: As in Attack 6.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).

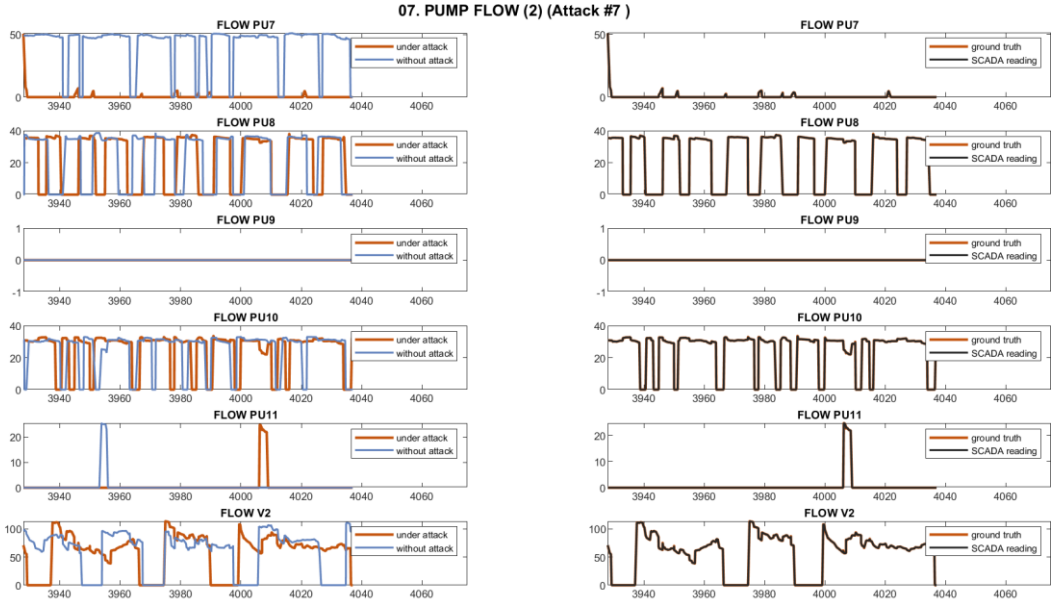


Attack: As in Attack 6.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).



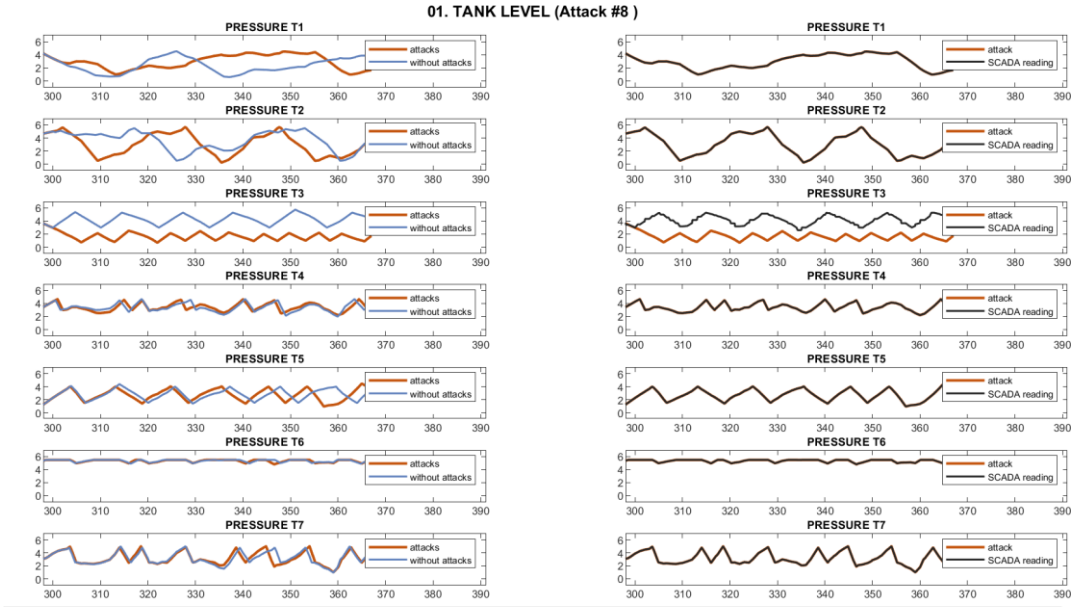
Attack: As in Attack 6.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).

07. PUMP FLOW (2) (Attack #7)

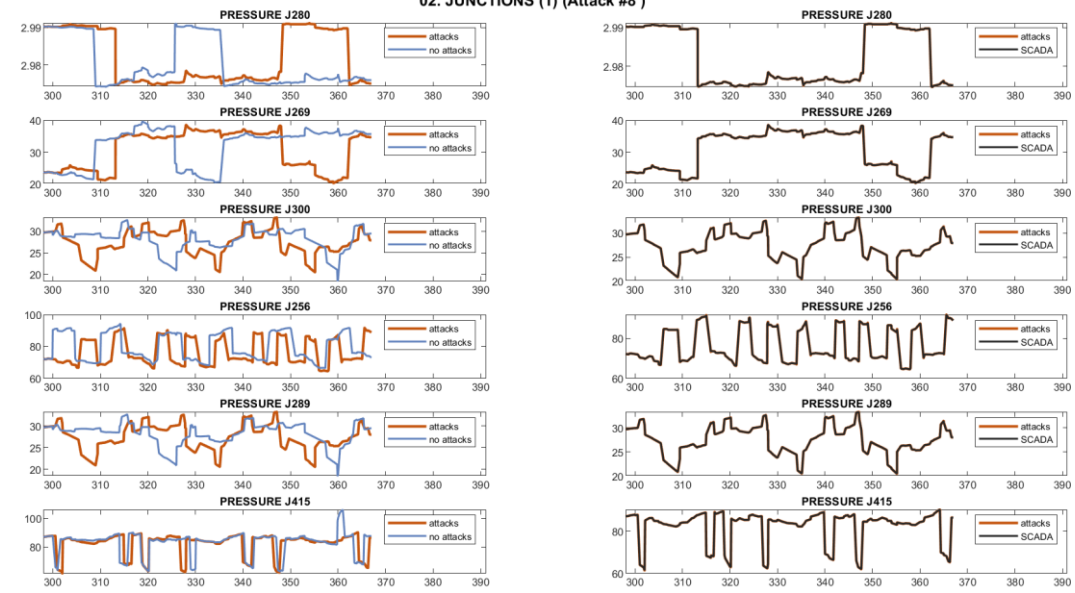


Attack: As in Attack 6.
SCADA concealment: Replay attack (L_T1, F_PU1, F_PU2, S_PU1, S_PU2).

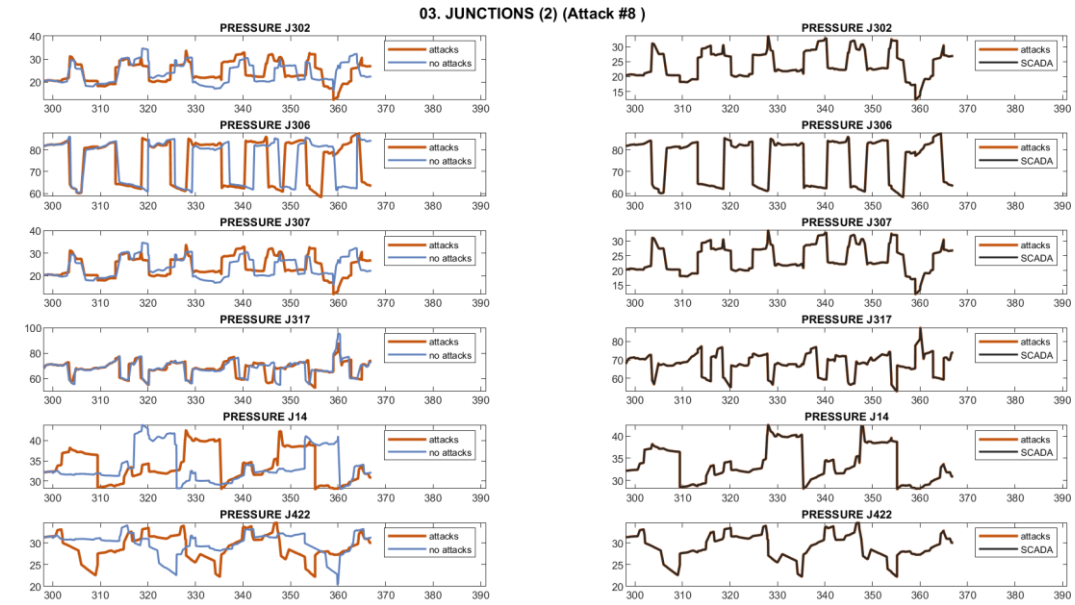
dataset_b03



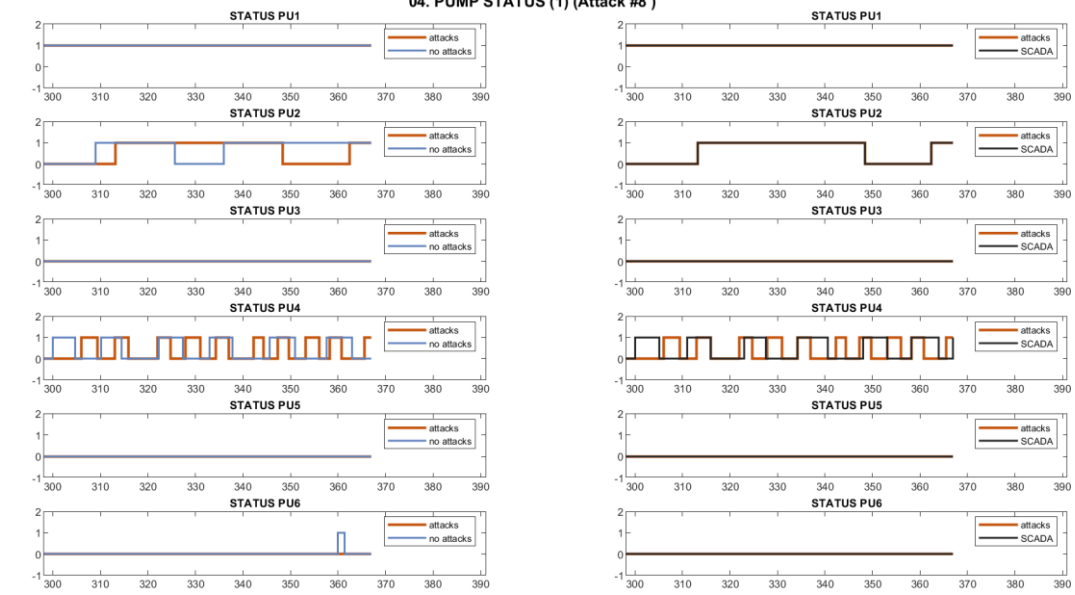
Attack: Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.
SCADA concealment: Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).



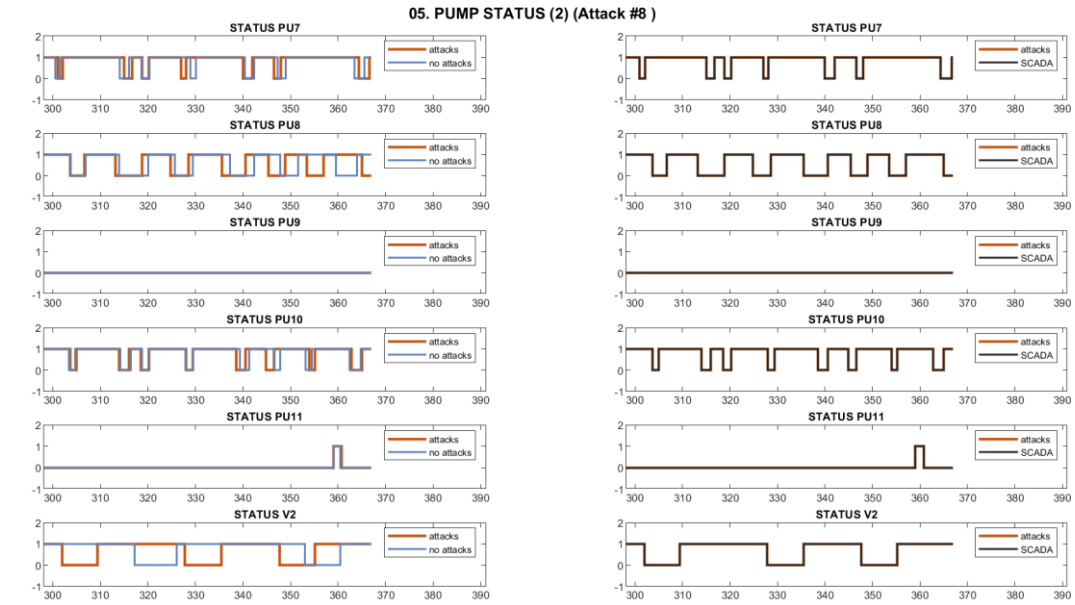
Attack: Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.
SCADA concealment: Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).



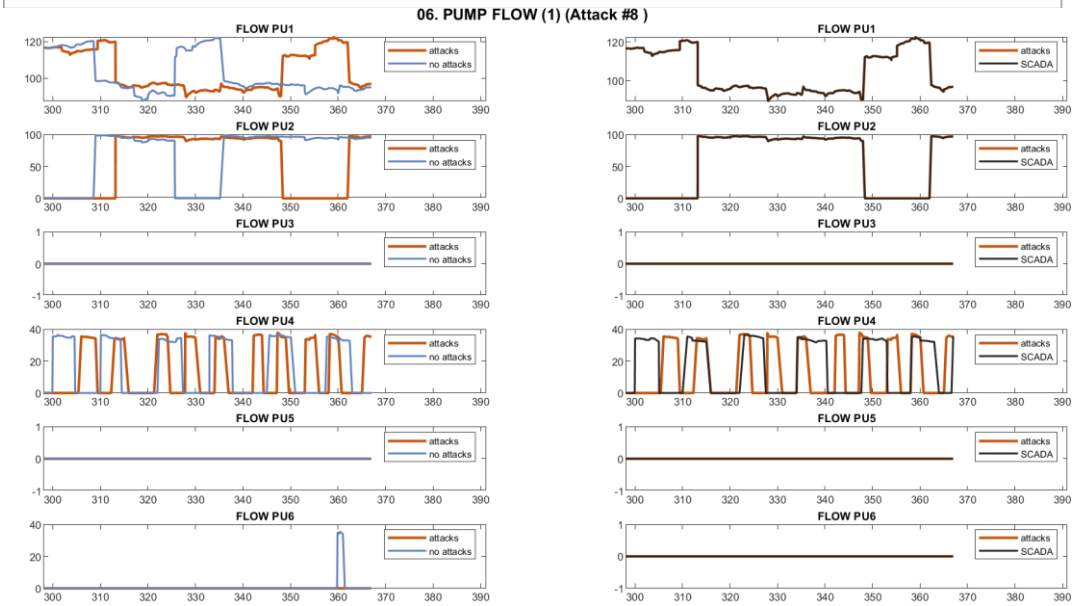
Attack: Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.
SCADA concealment: Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).



Attack: Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.
SCADA concealment: Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).

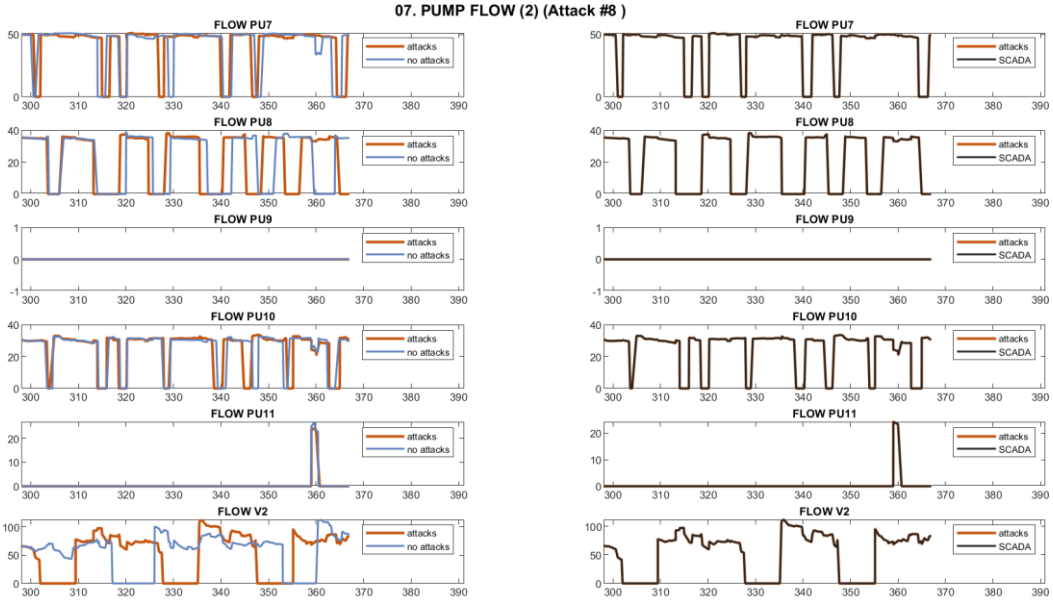


Attack: Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.
SCADA concealment: Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).



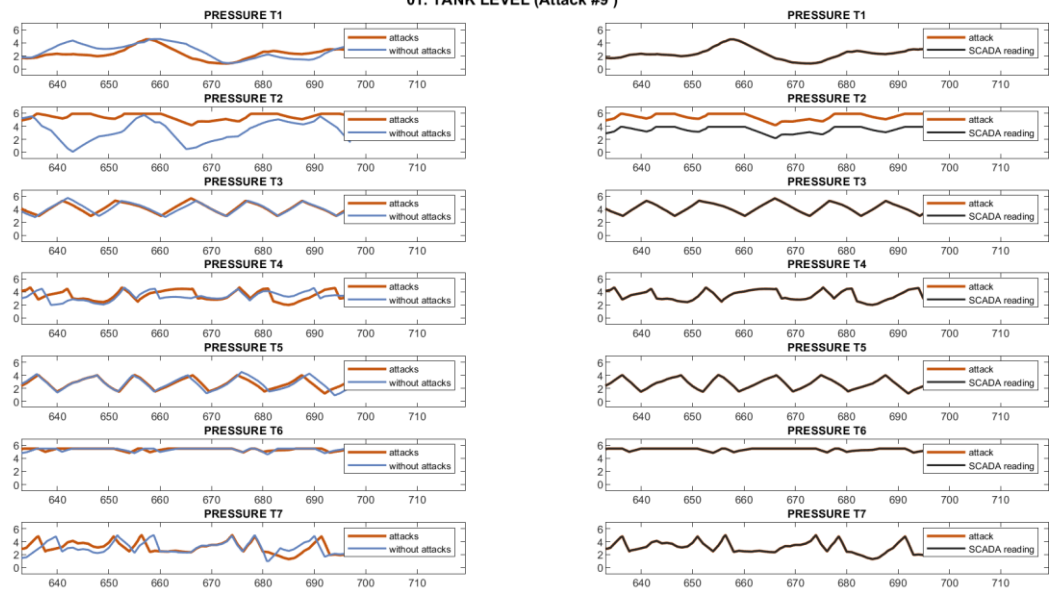
Attack: Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.
SCADA concealment: Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).

07. PUMP FLOW (2) (Attack #8)

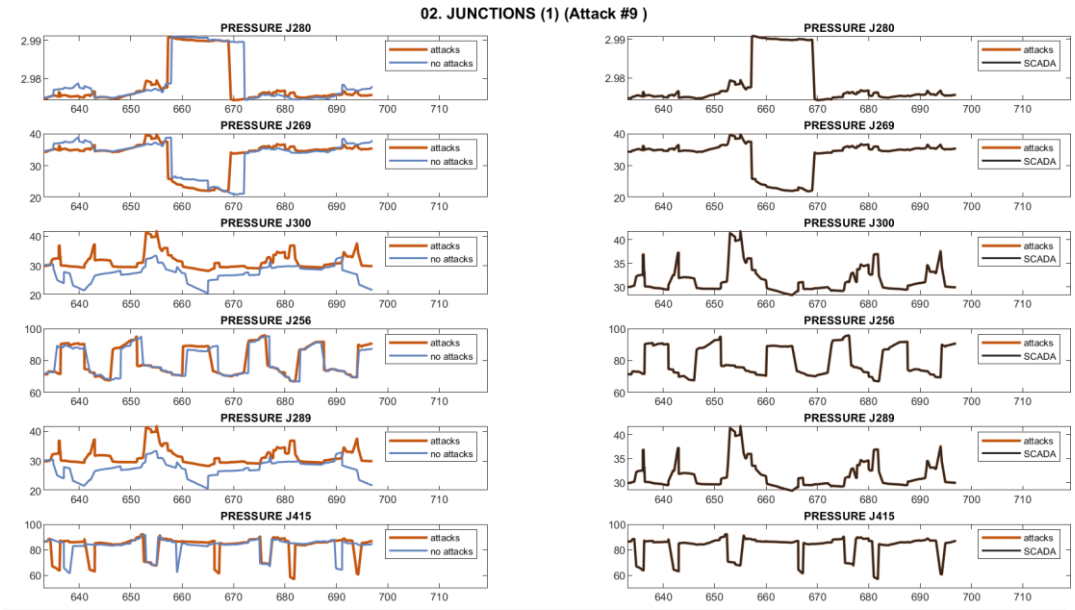


Attack: Attacker changes L_T3 thresholds controlling PU4 and PU5 by gaining control of PLC3. Low levels in T3.
SCADA concealment: Replay attack (L_T3, F_PU4, F_PU5, S_PU4, S_PU5).

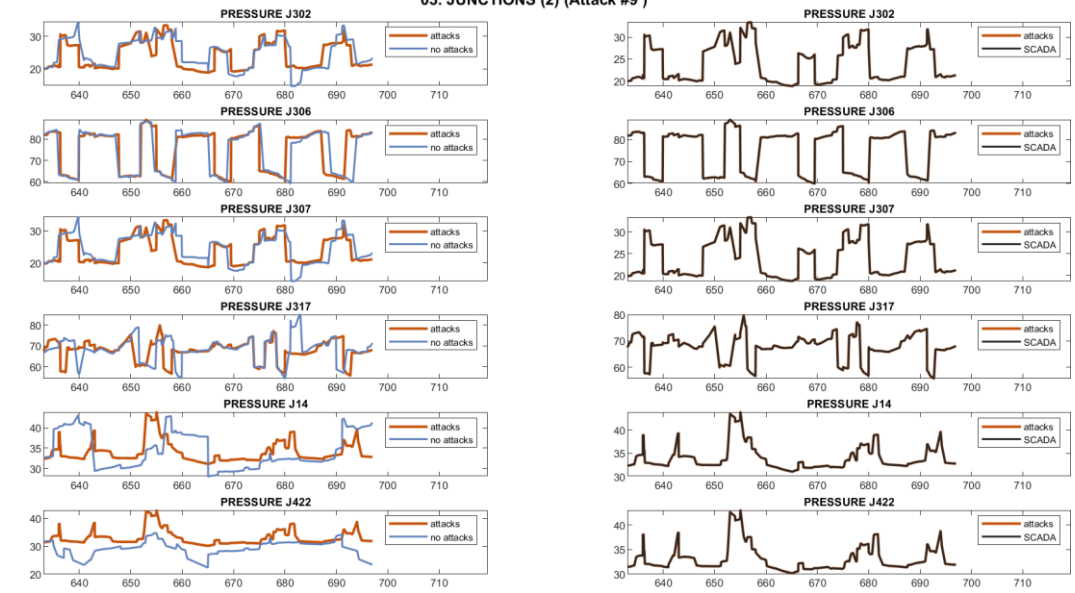
01. TANK LEVEL (Attack #9)



Attack: Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.
SCADA concealment: Polyline to offset L_T2 increase.

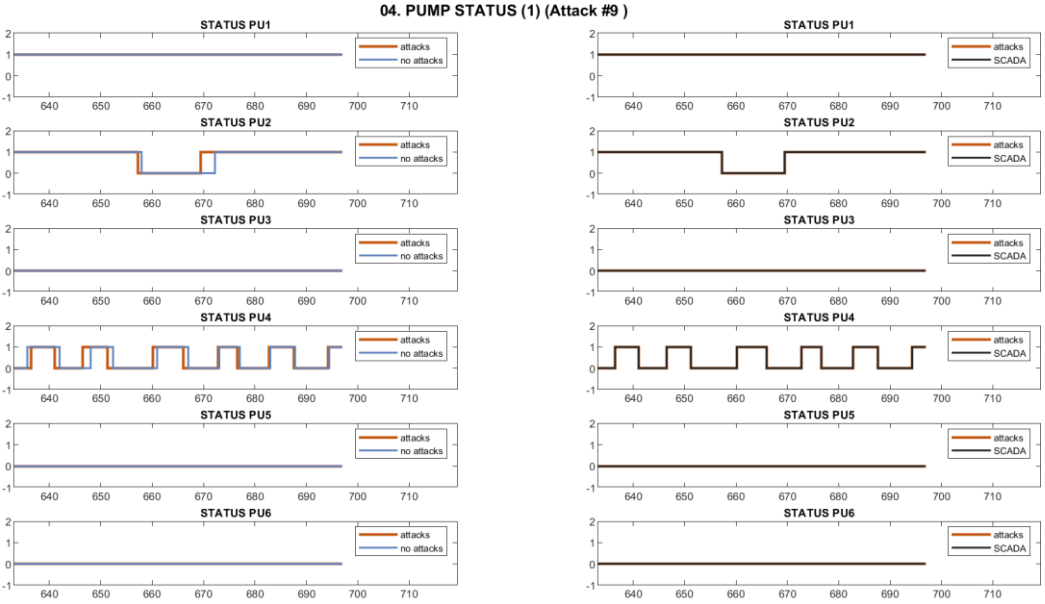


Attack: Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.
SCADA concealment: Polyline to offset L_T2 increase.



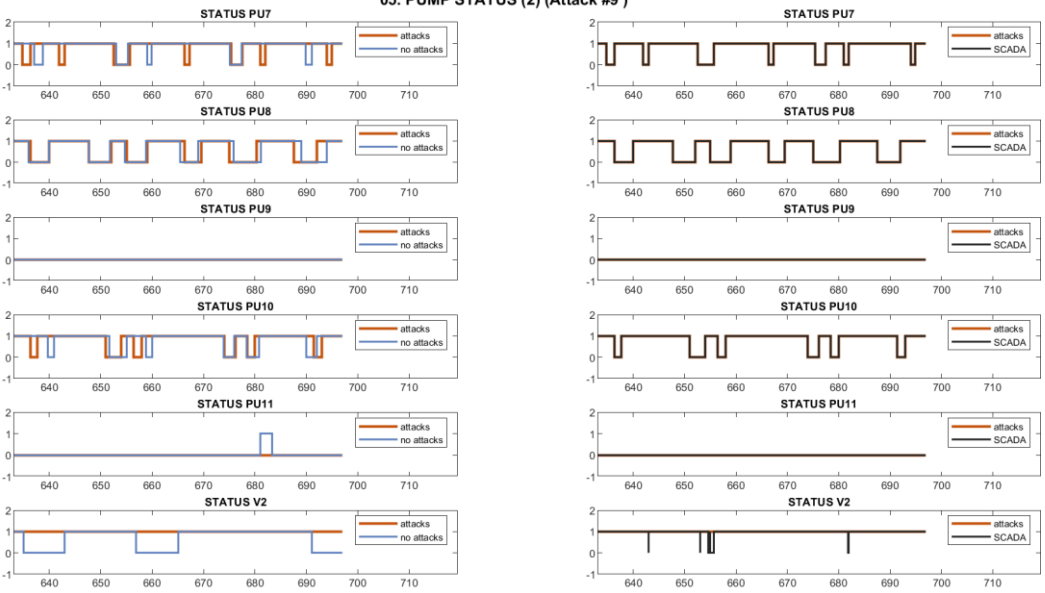
Attack: Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.
SCADA concealment: Polyline to offset L_T2 increase.

04. PUMP STATUS (1) (Attack #9)



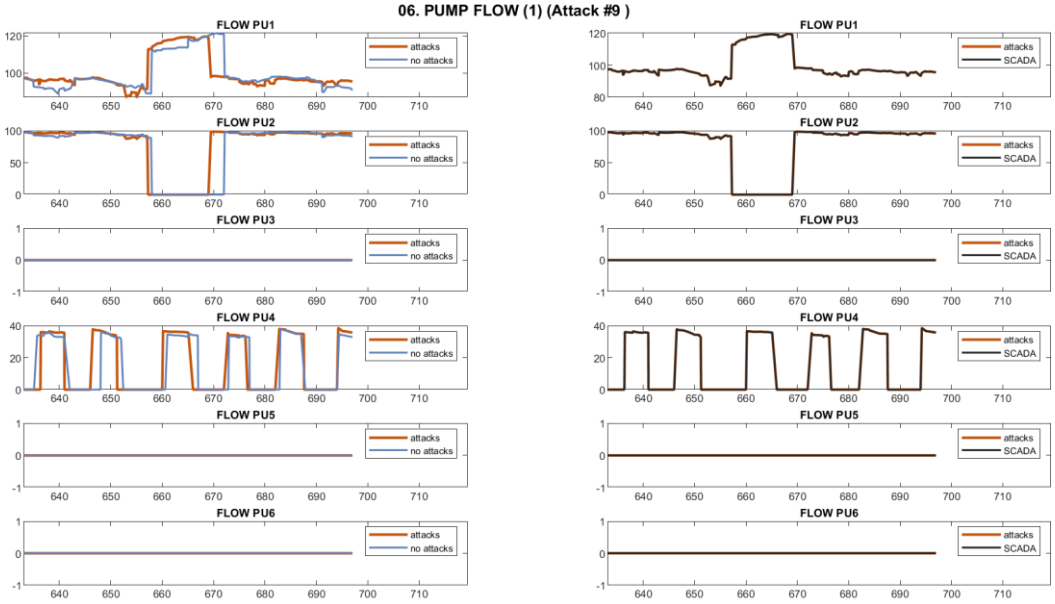
Attack: Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.
SCADA concealment: Polyline to offset L_T2 increase.

05. PUMP STATUS (2) (Attack #9)



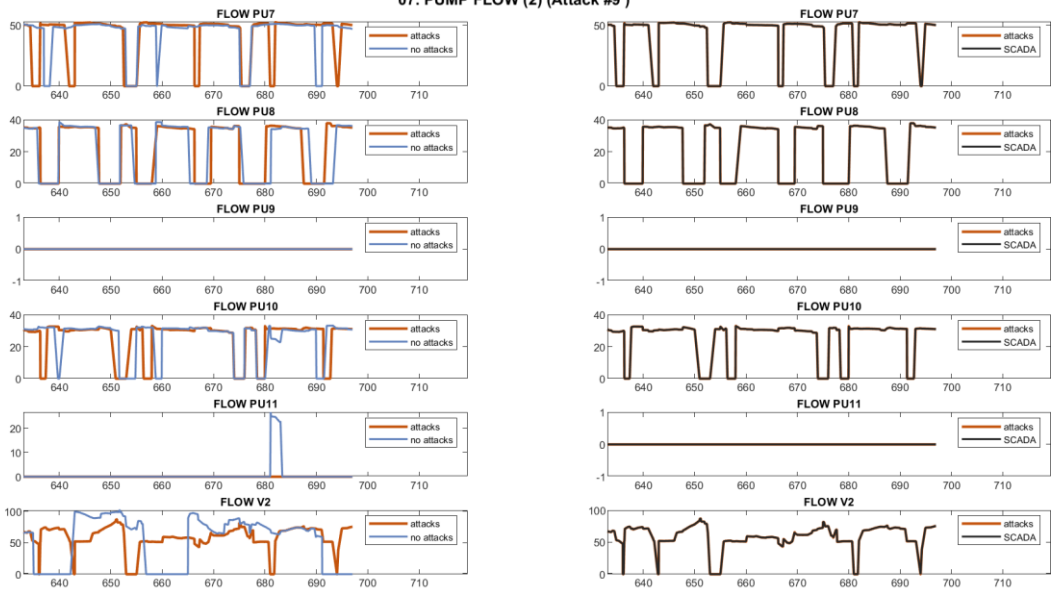
Attack: Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.
SCADA concealment: Polyline to offset L_T2 increase.

06. PUMP FLOW (1) (Attack #9)



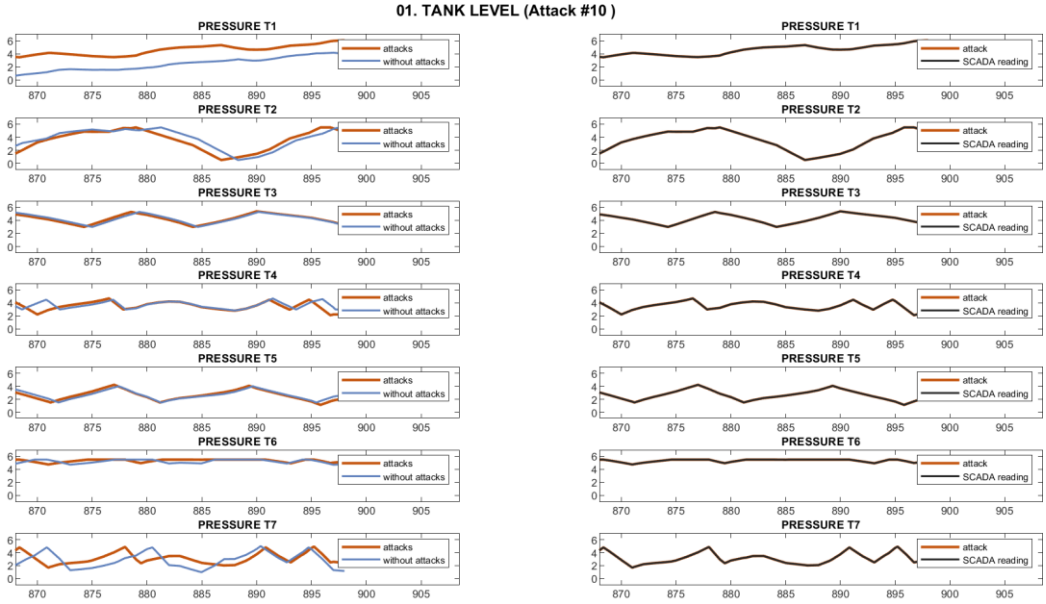
Attack: Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.
SCADA concealment: Polyline to offset L_T2 increase.

07. PUMP FLOW (2) (Attack #9)



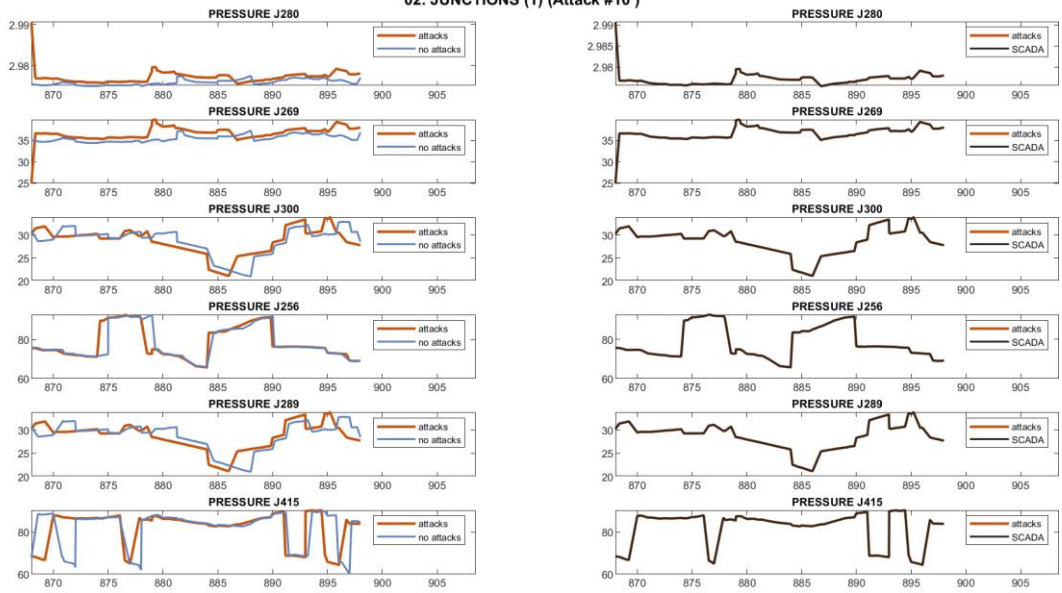
Attack: Attacker alters L_T2 readings arriving to PLC3, which reads a constant low level and forces Valve V2 open, leading T2 to overflow.
SCADA concealment: Polyline to offset L_T2 increase.

01. TANK LEVEL (Attack #10)

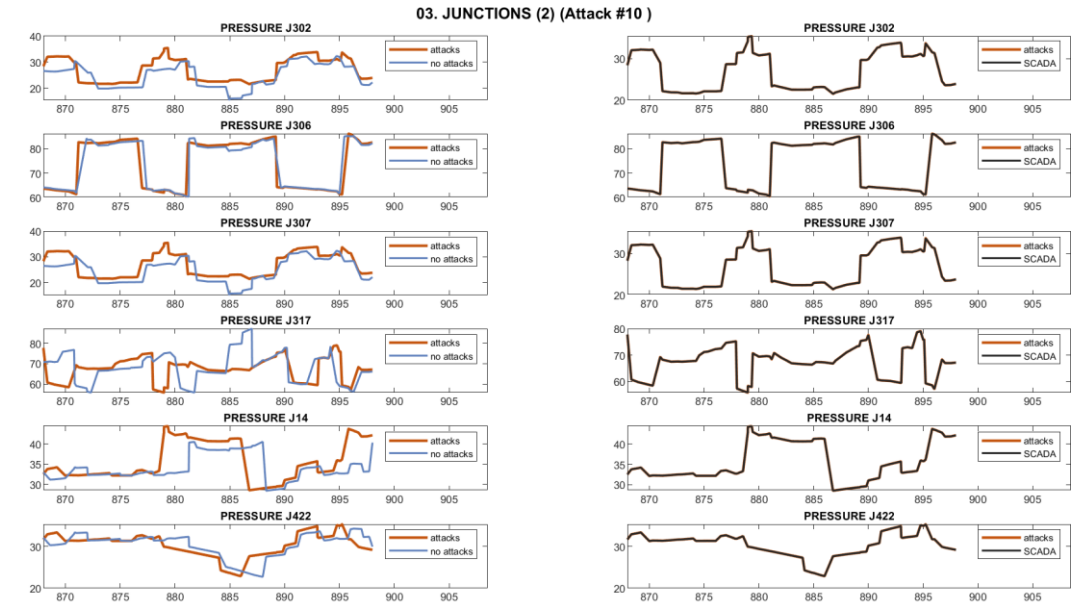


Attack: Malicious activation of Pump PU3.
SCADA concealment: -

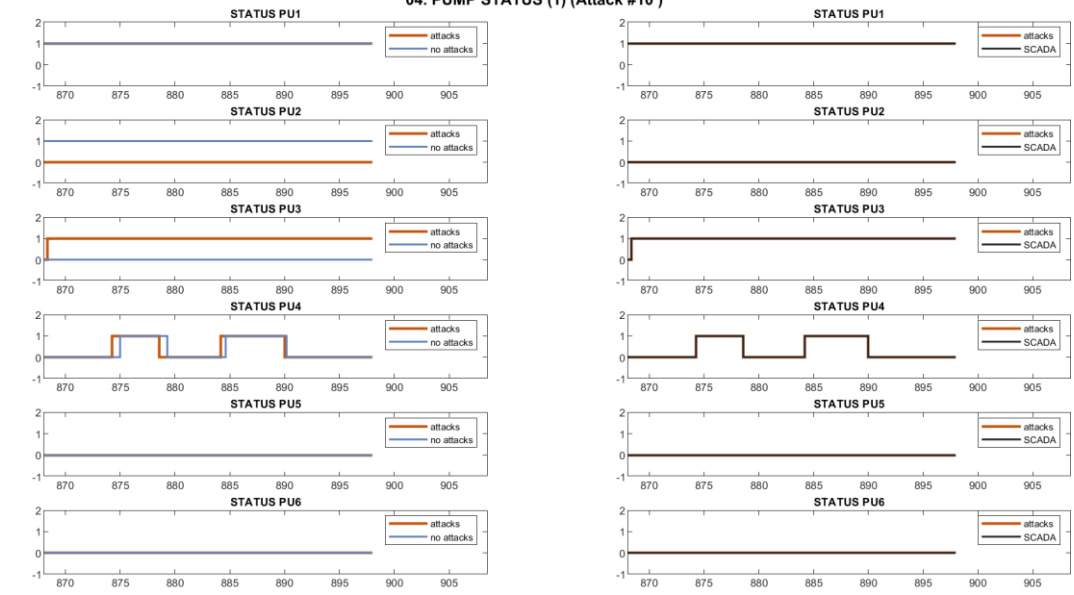
02. JUNCTIONS (1) (Attack #10)



Attack: Malicious activation of Pump PU3.
SCADA concealment: -

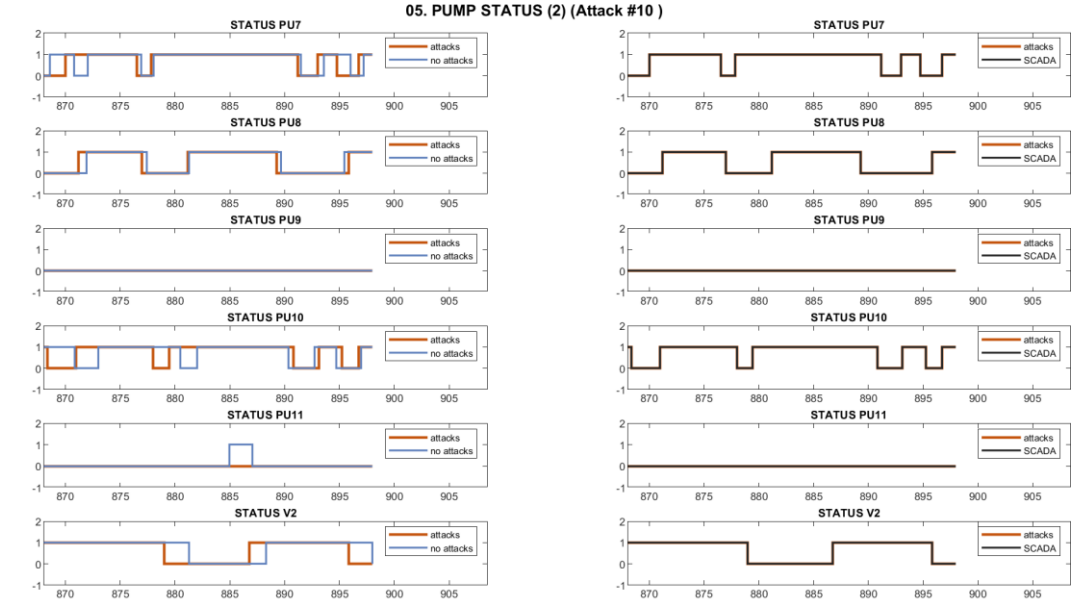


Attack: Malicious activation of Pump PU3.
SCADA concealment: -



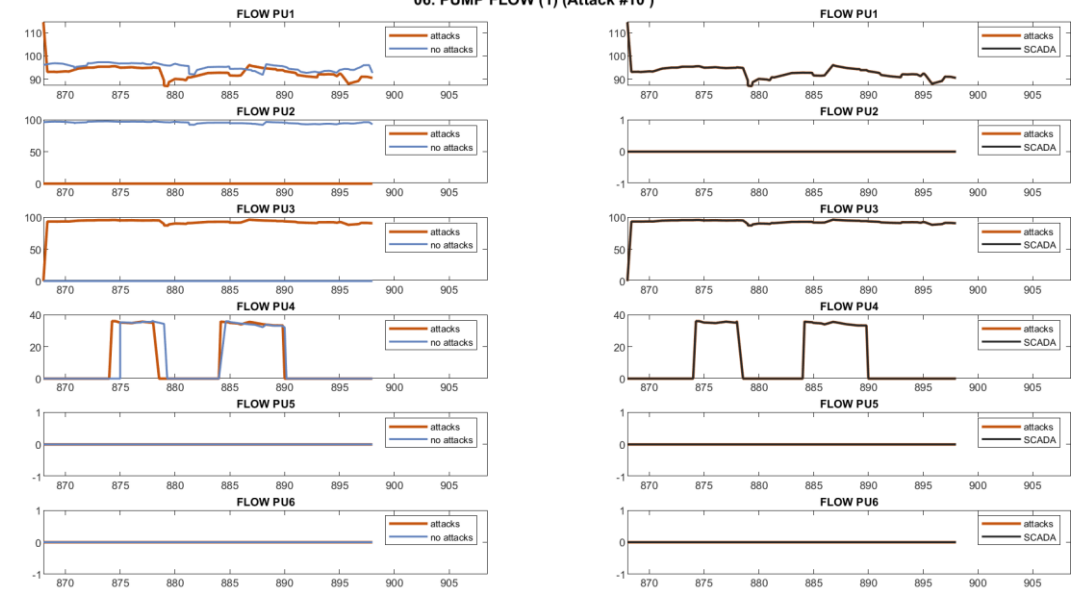
Attack: Malicious activation of Pump PU3.
SCADA concealment: -

05. PUMP STATUS (2) (Attack #10)



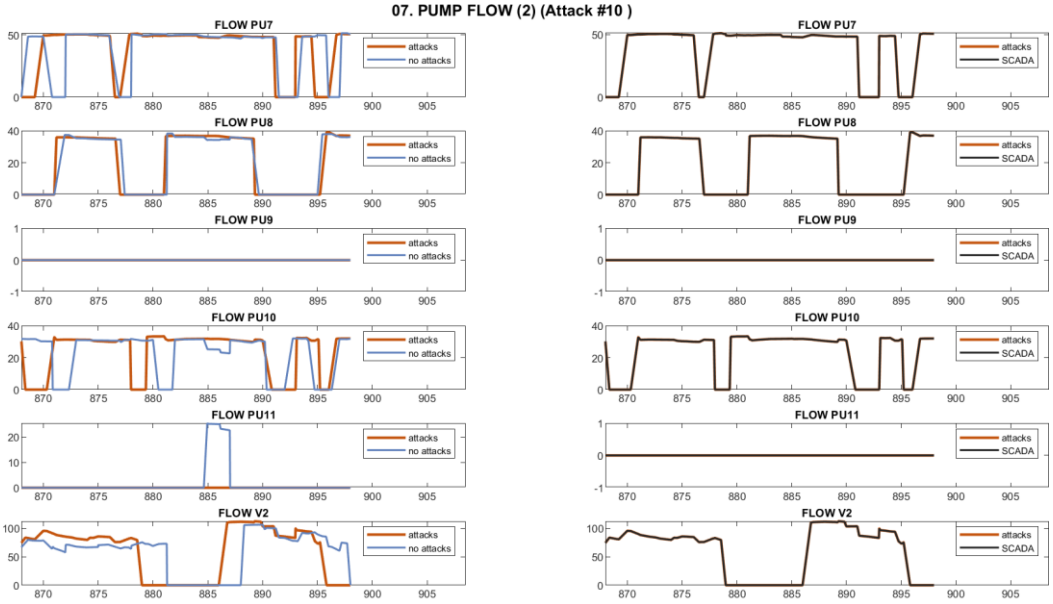
Attack: Malicious activation of Pump PU3.
SCADA concealment: -

06. PUMP FLOW (1) (Attack #10)



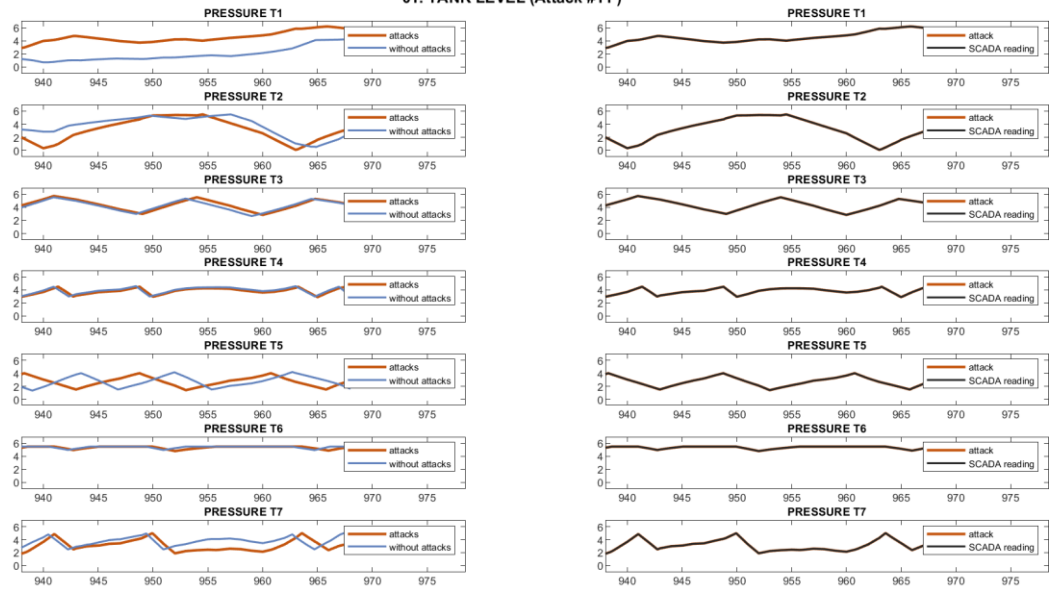
Attack: Malicious activation of Pump PU3.
SCADA concealment: -

07. PUMP FLOW (2) (Attack #10)



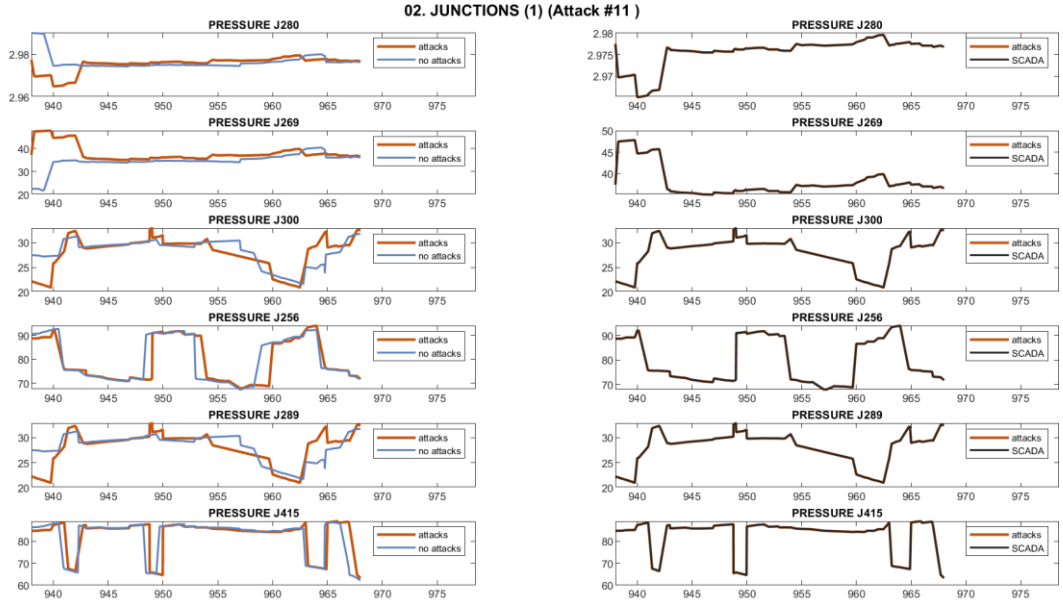
Attack: Malicious activation of Pump PU3.
SCADA concealment: -

01. TANK LEVEL (Attack #11)



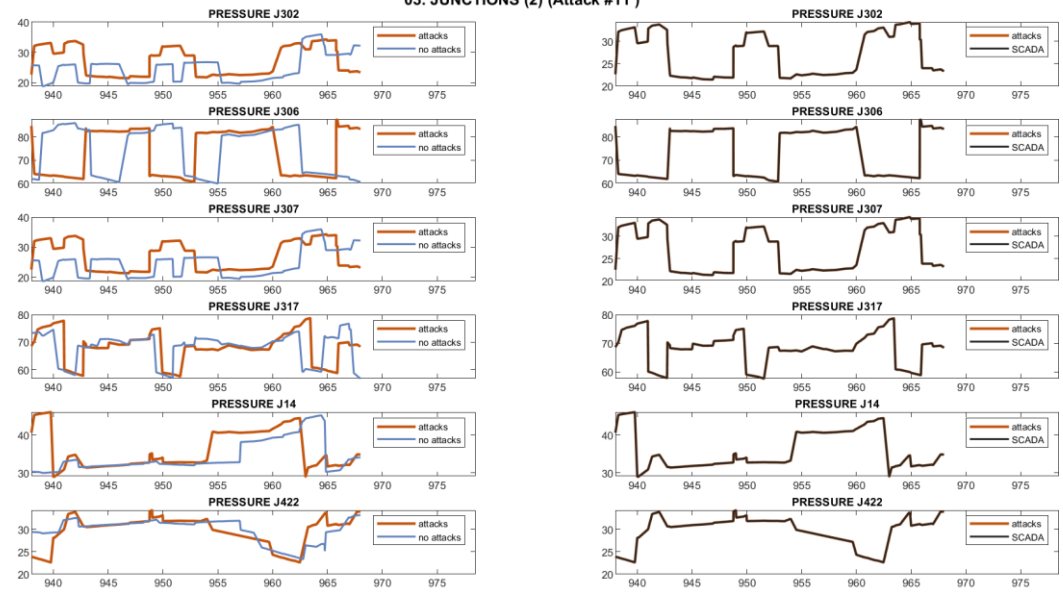
Attack: As in Attack 10.
SCADA concealment: -

02. JUNCTIONS (1) (Attack #11)



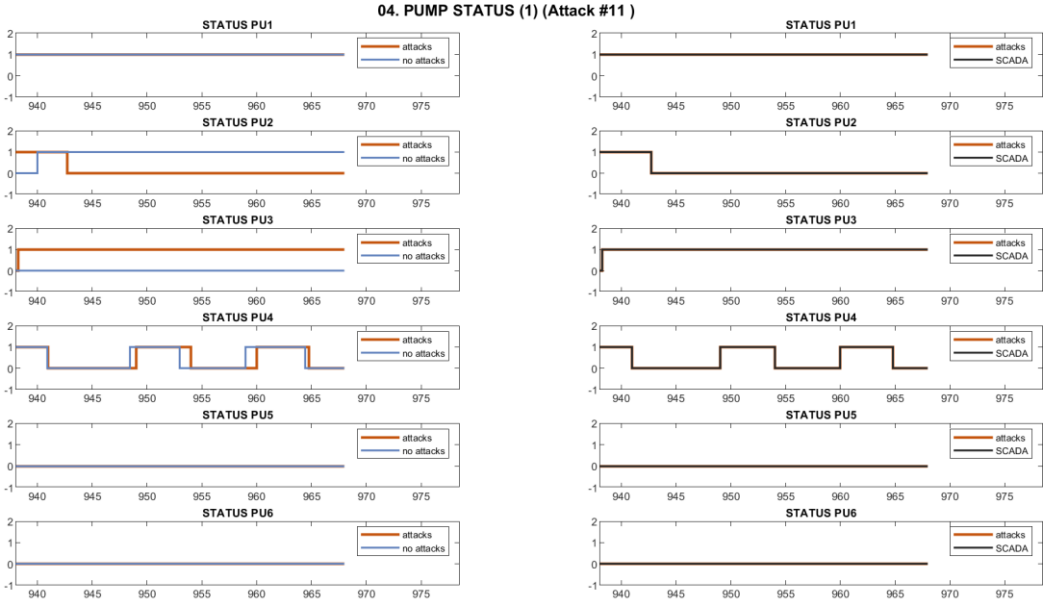
Attack: As in Attack 10.
SCADA concealment: -

03. JUNCTIONS (2) (Attack #11)



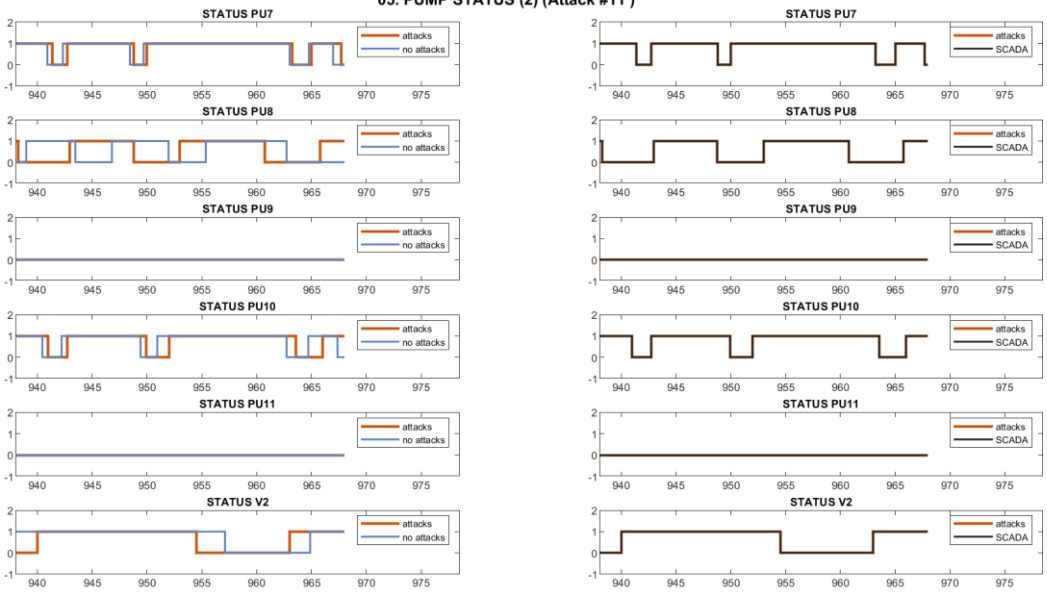
Attack: As in Attack 10.
SCADA concealment: -

04. PUMP STATUS (1) (Attack #11)



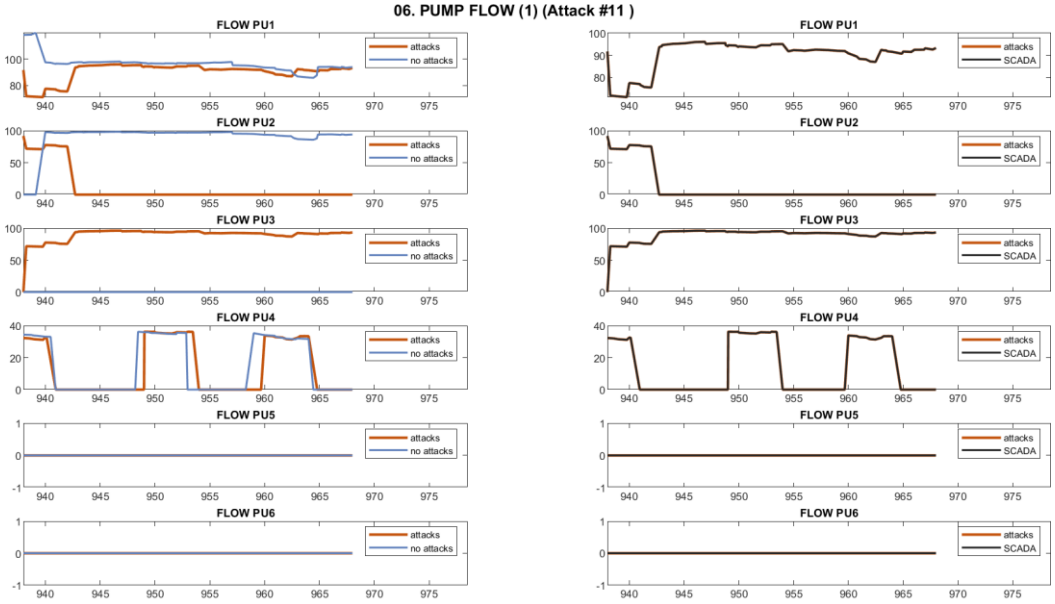
Attack: As in Attack 10.
SCADA concealment: -

05. PUMP STATUS (2) (Attack #11)



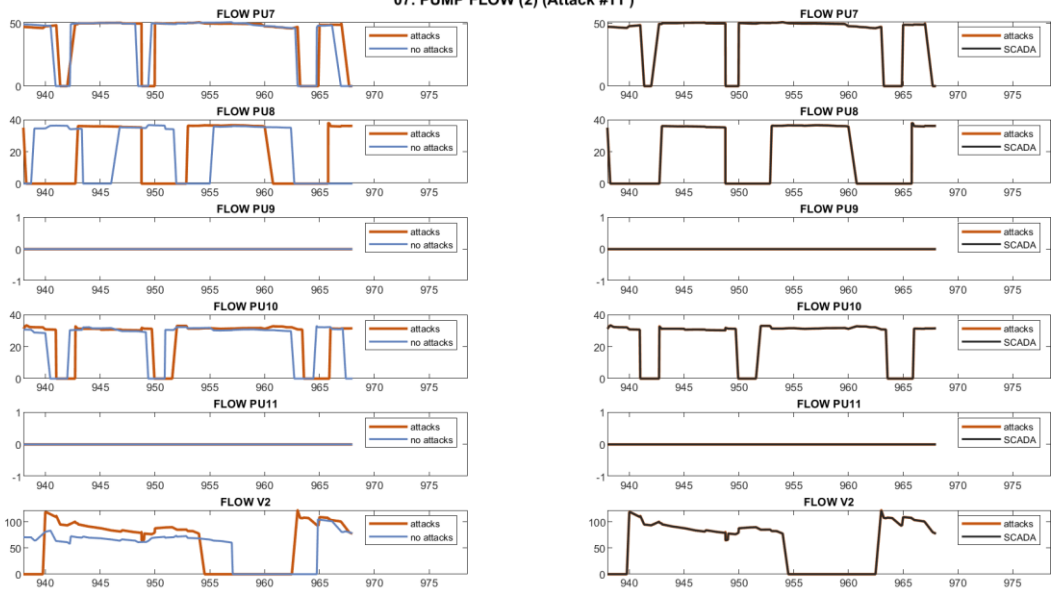
Attack: As in Attack 10.
SCADA concealment: -

06. PUMP FLOW (1) (Attack #11)



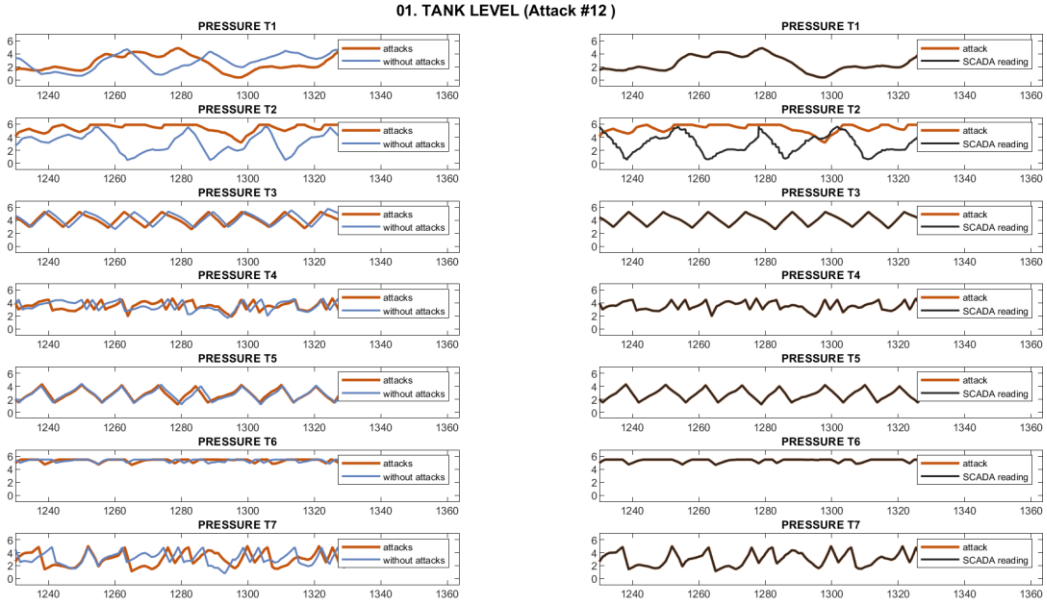
Attack: As in Attack 10.
SCADA concealment: -

07. PUMP FLOW (2) (Attack #11)



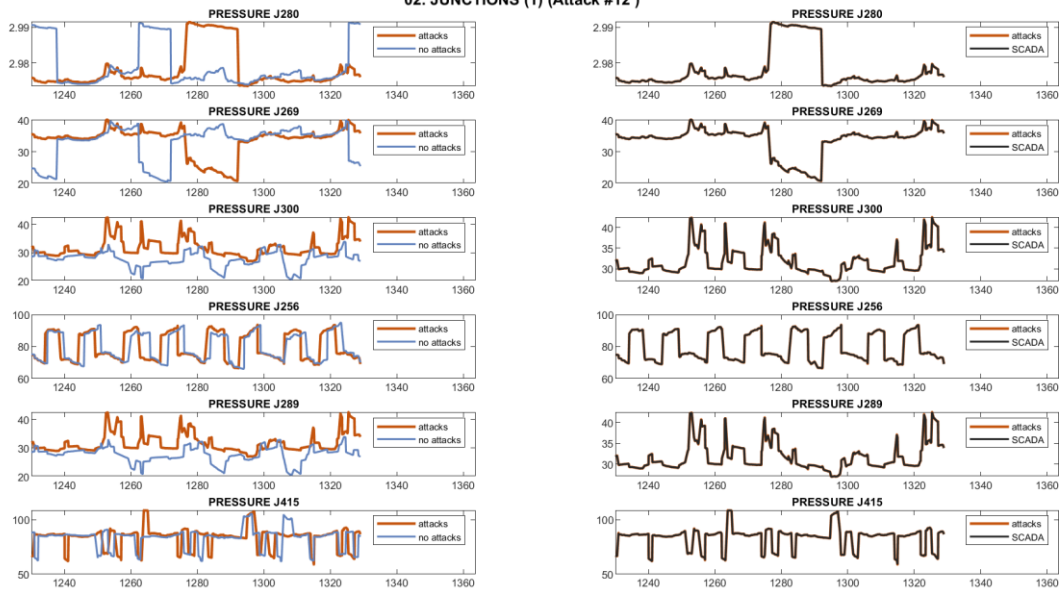
Attack: As in Attack 10.
SCADA concealment: -

01. TANK LEVEL (Attack #12)

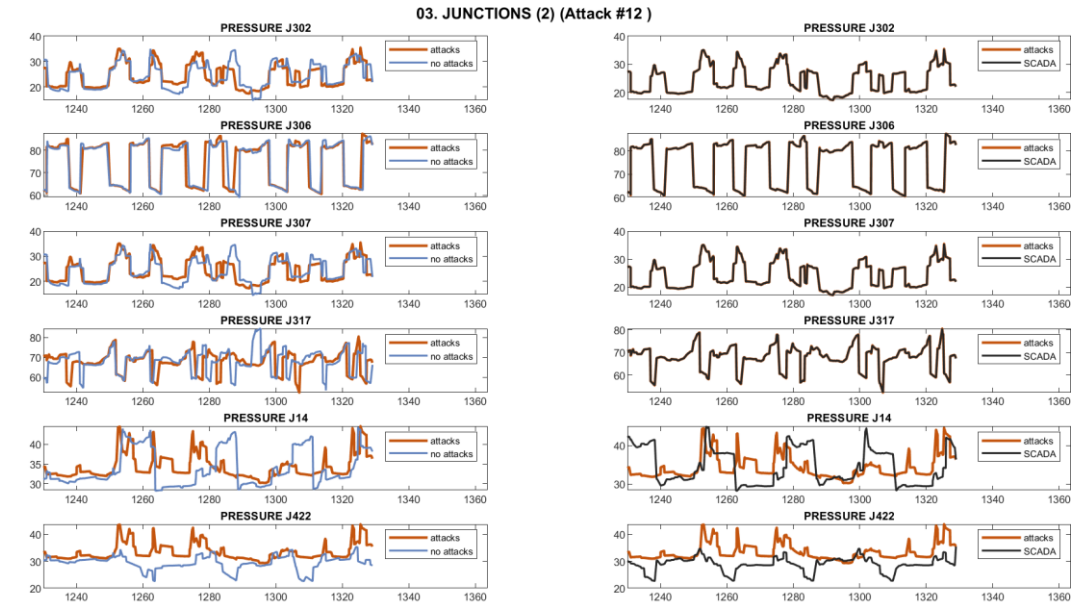


Attack: As in Attack 9.
SCADA concealment: Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)

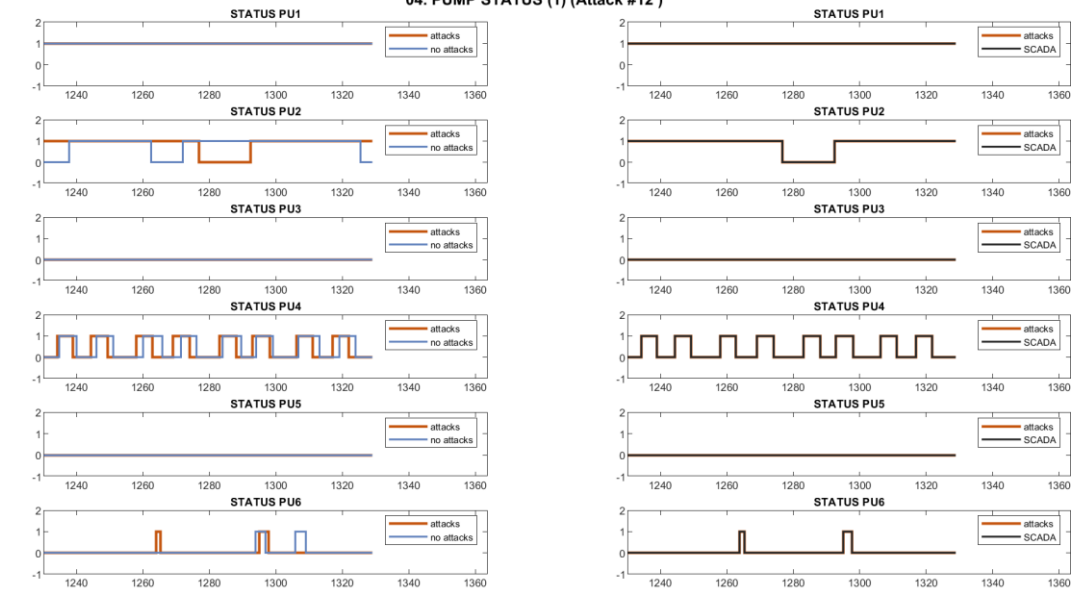
02. JUNCTIONS (1) (Attack #12)



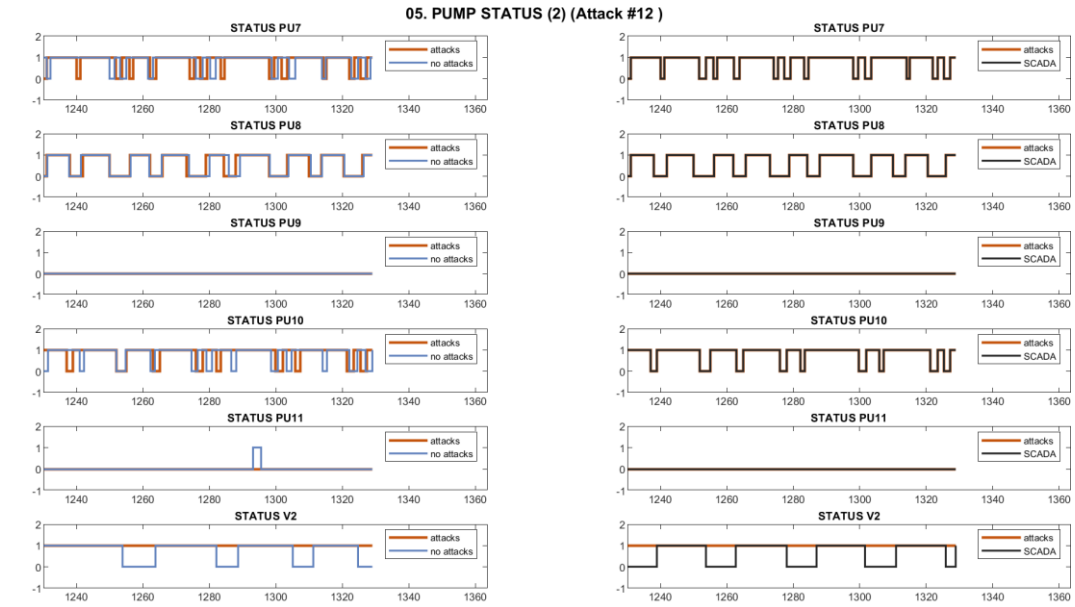
Attack: As in Attack 9.
SCADA concealment: Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)



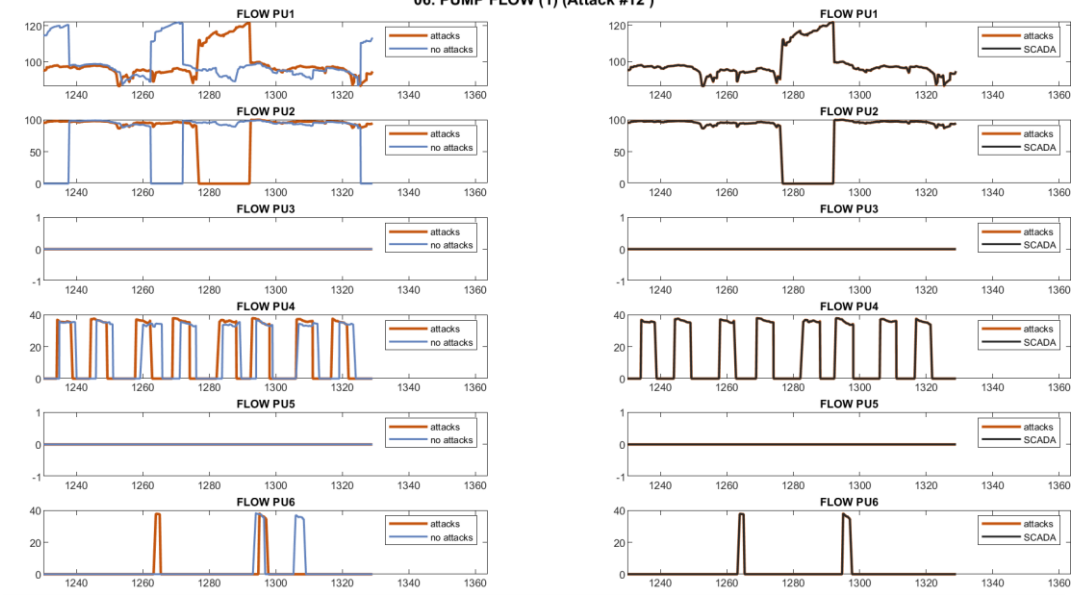
Attack: As in Attack 9.
SCADA concealment: Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)



Attack: As in Attack 9.
SCADA concealment: Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)

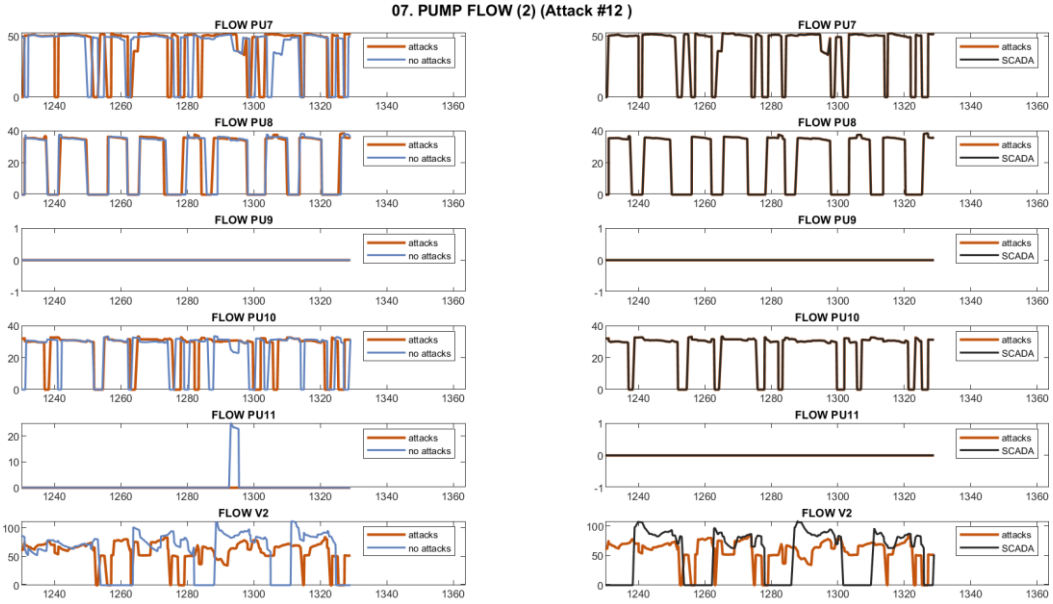


Attack: As in Attack 9.
SCADA concealment: Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)



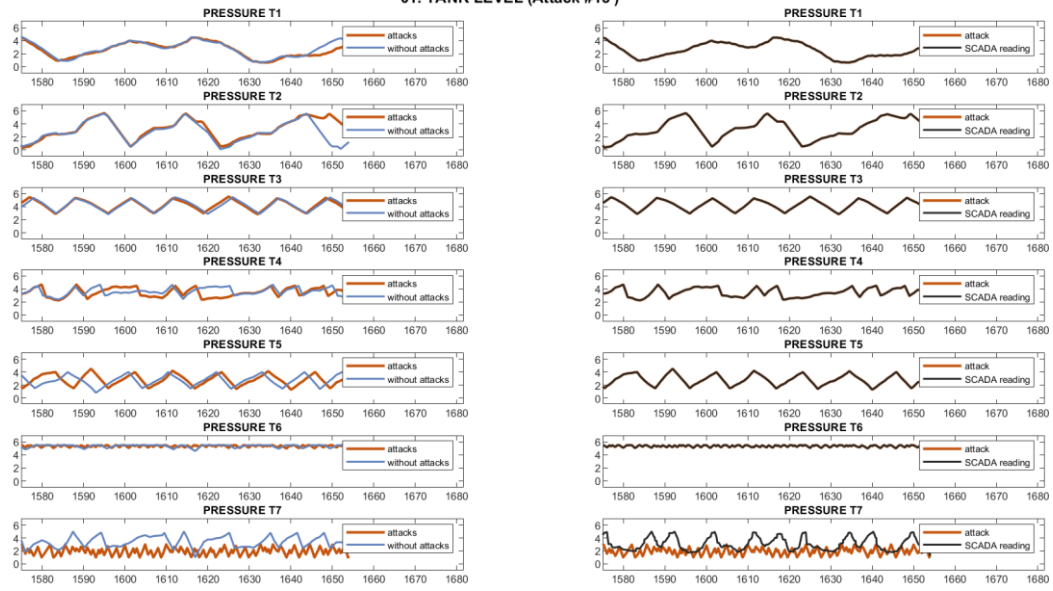
Attack: As in Attack 9.
SCADA concealment: Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)

07. PUMP FLOW (2) (Attack #12)

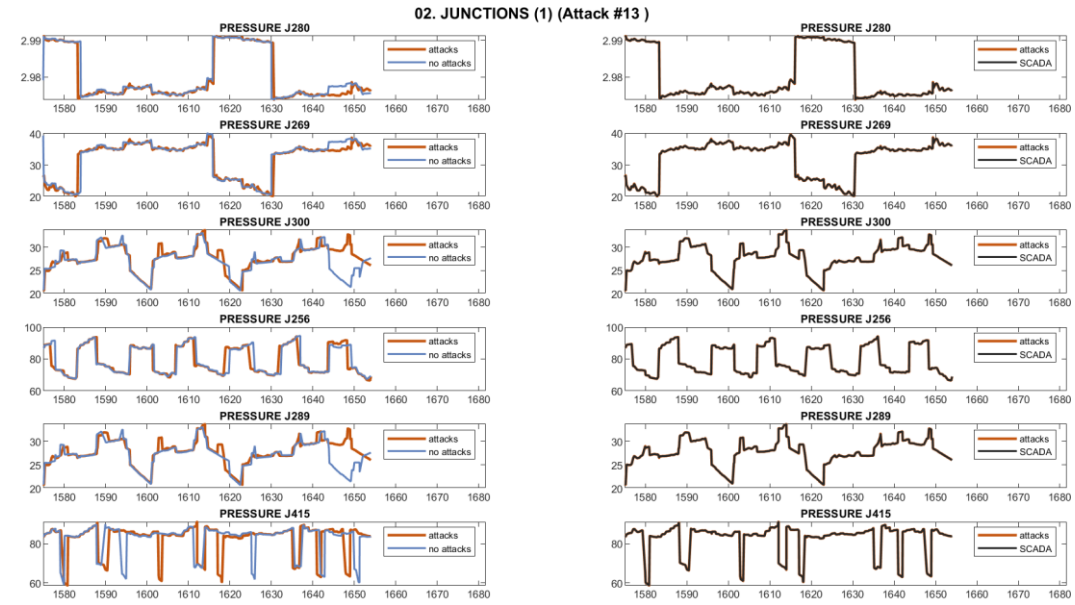


Attack: As in Attack 9.
SCADA concealment: Replay attack (L_T2, F_V2, S_V2, P_J422, P_J14)

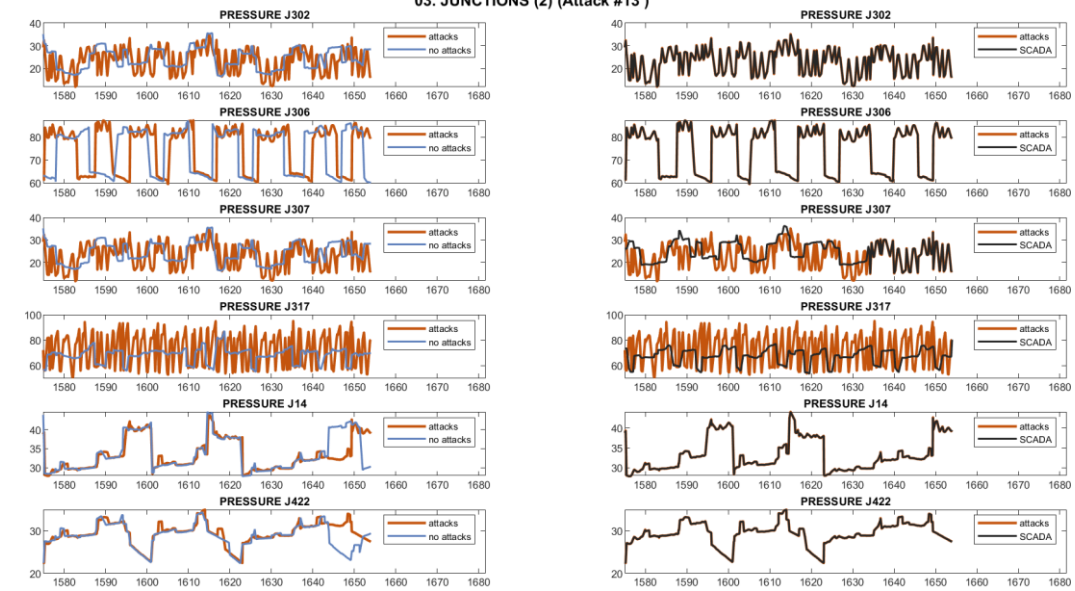
01. TANK LEVEL (Attack #13)



Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.

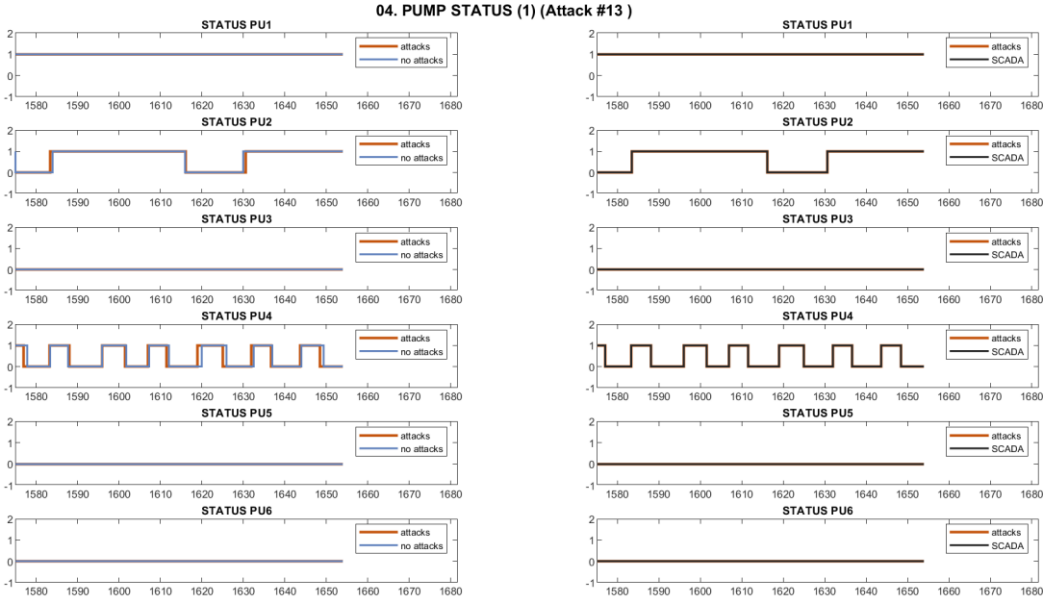


Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.



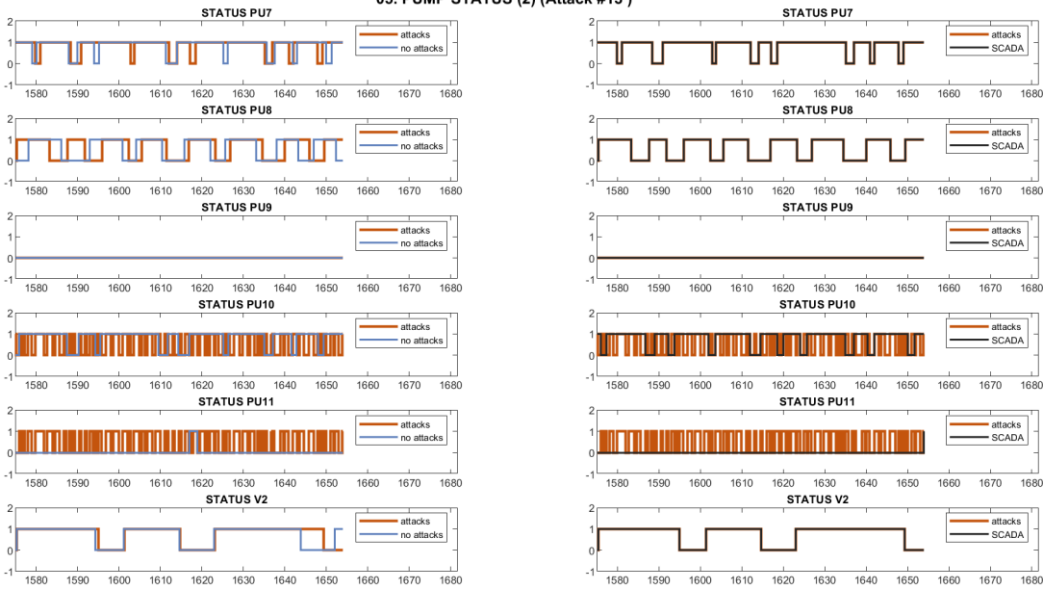
Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.

04. PUMP STATUS (1) (Attack #13)

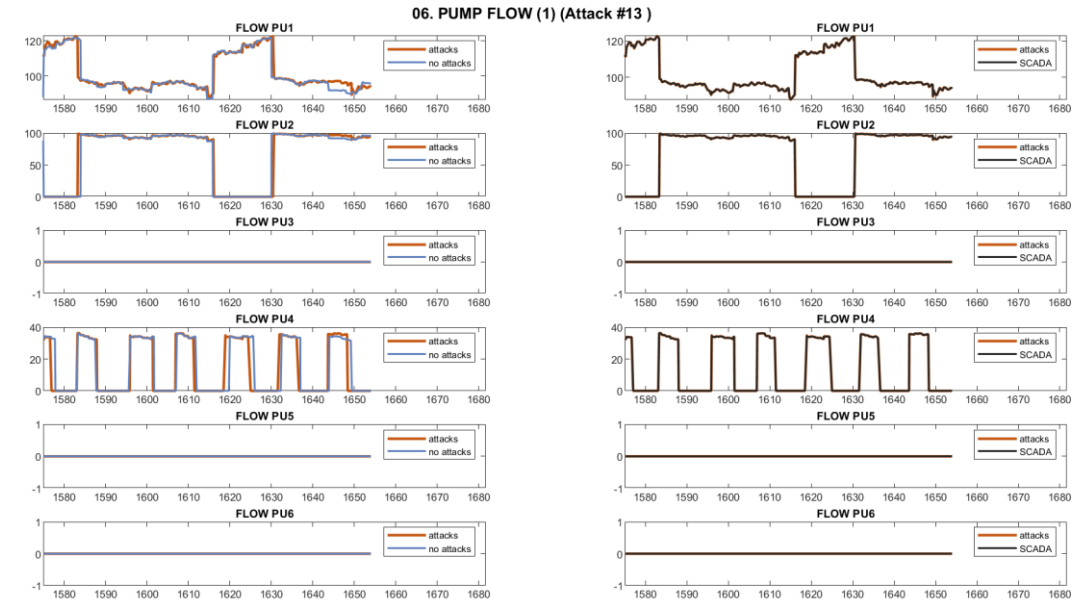


Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.

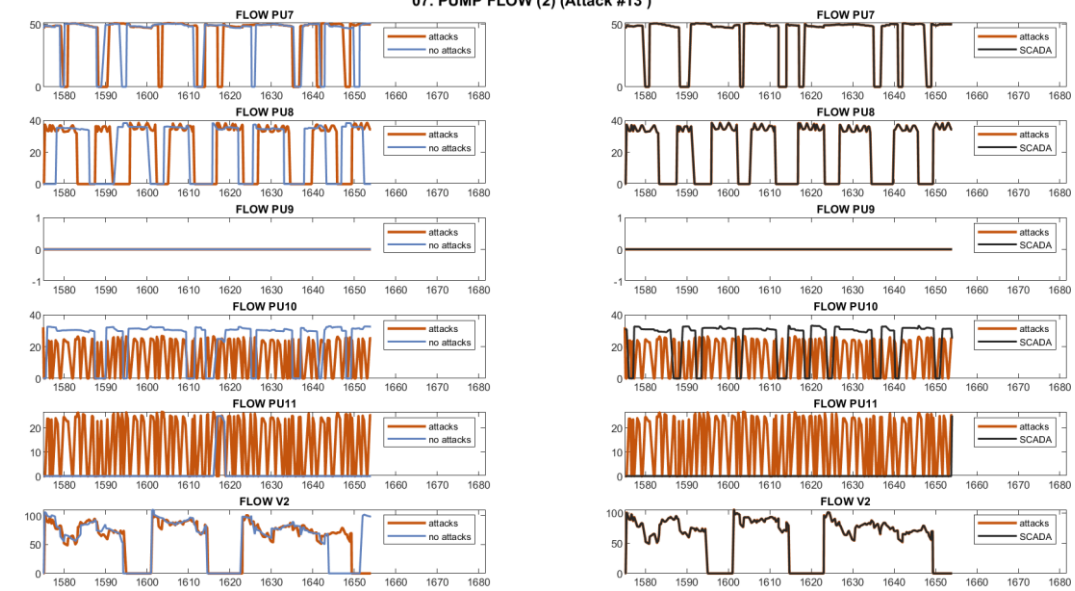
05. PUMP STATUS (2) (Attack #13)



Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.

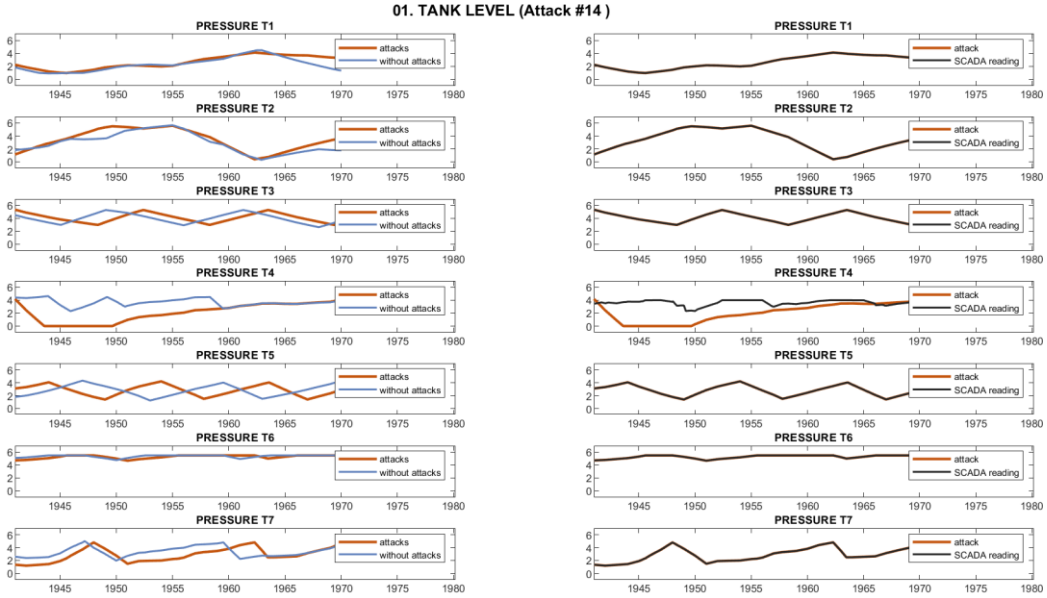


Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.



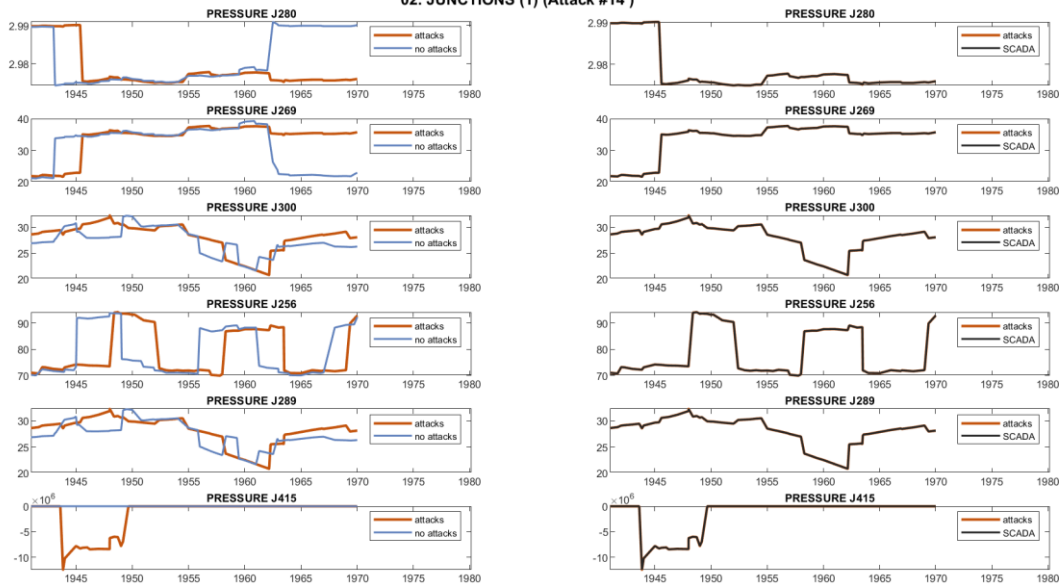
Attack: Attacker changes L_T7 thresholds controlling PU10 and PU11 by gaining control of PLC5, causing the pumps to switch continuously.
SCADA concealment: Replay attack (L_T7, F_PU10, F_PU11, S_PU10, S_PU11, P_J317, P_J307). Inlet pressure (P_J307) concealment terminates before that of other variables.

01. TANK LEVEL (Attack #14)

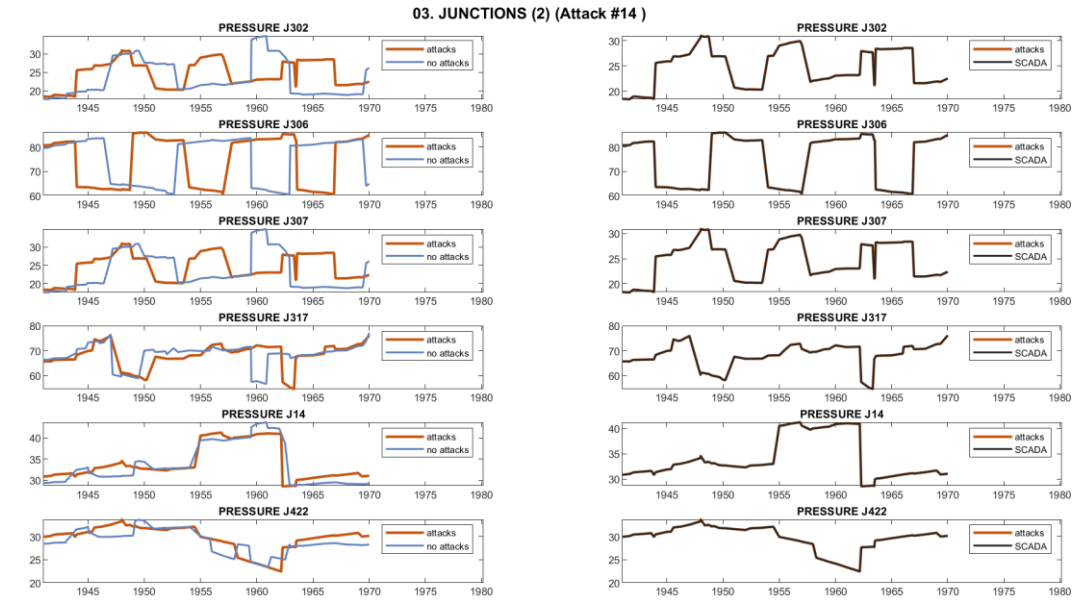


Attack: Alteration of T4 signal arriving at PLC6.
SCADA concealment: -

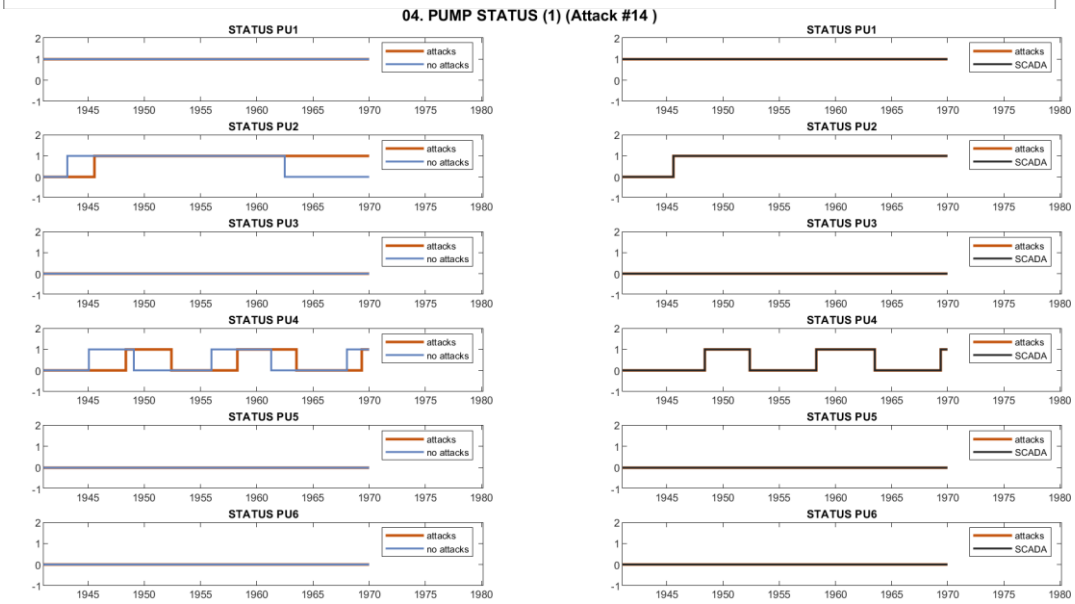
02. JUNCTIONS (1) (Attack #14)



Attack: Alteration of T4 signal arriving at PLC6.
SCADA concealment: -

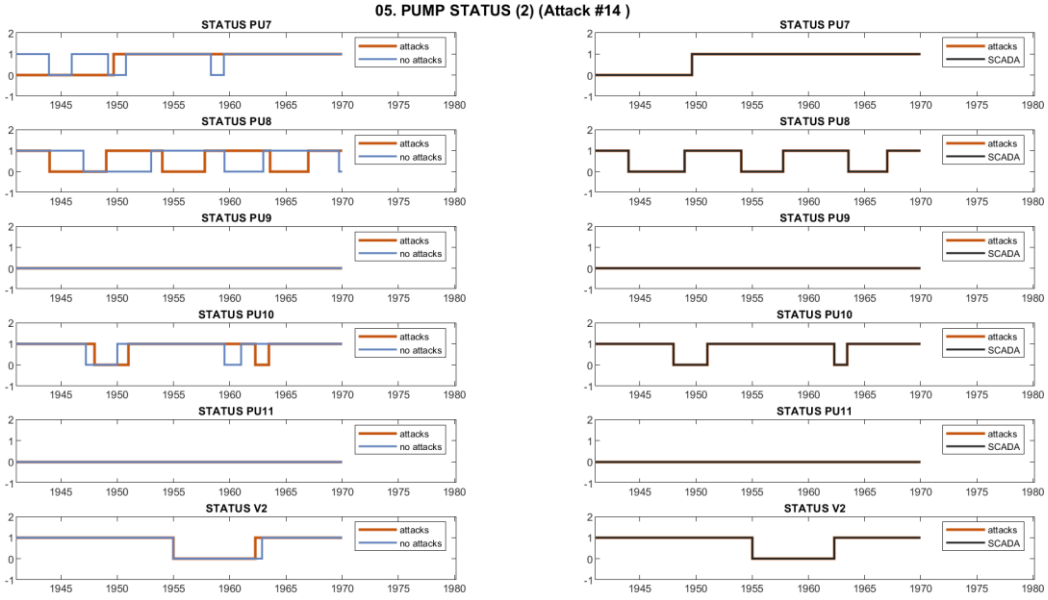


Attack: Alteration of T4 signal arriving at PLC6.
SCADA concealment: -



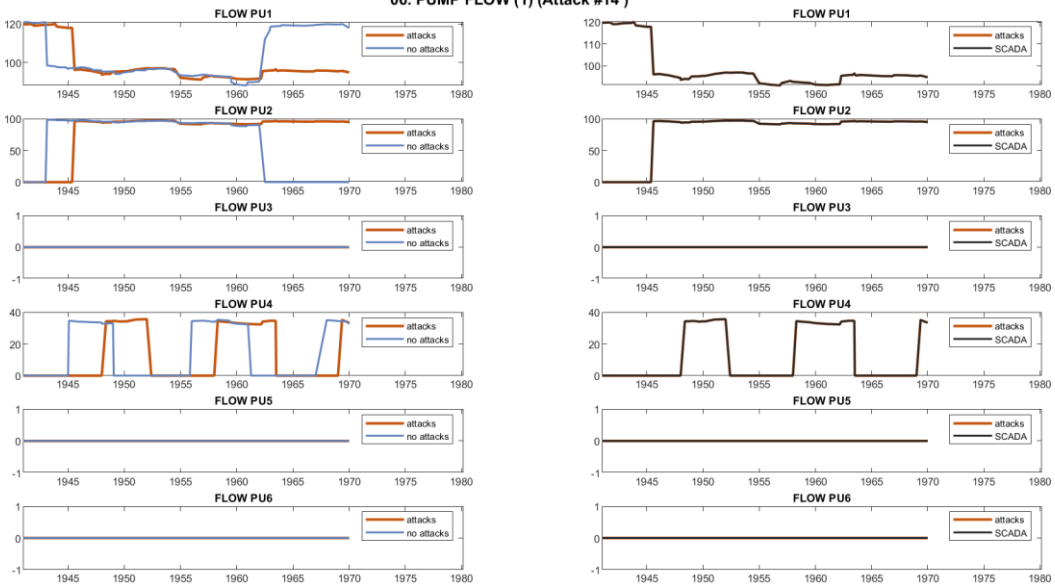
Attack: Alteration of T4 signal arriving at PLC6.
SCADA concealment: -

05. PUMP STATUS (2) (Attack #14)

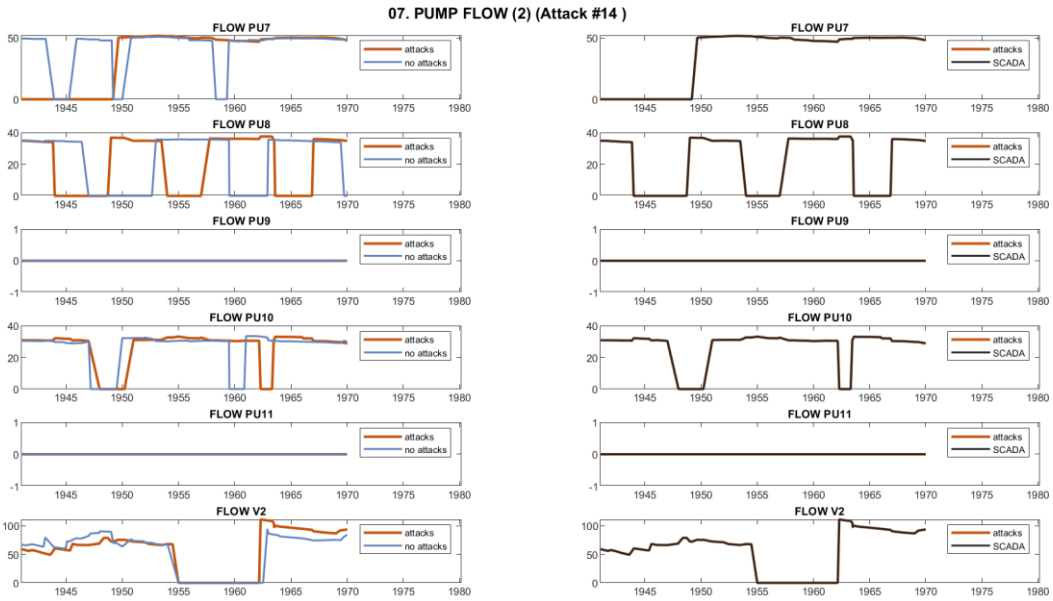


Attack: Alteration of T4 signal arriving at PLC6.
SCADA concealment: -

06. PUMP FLOW (1) (Attack #14)

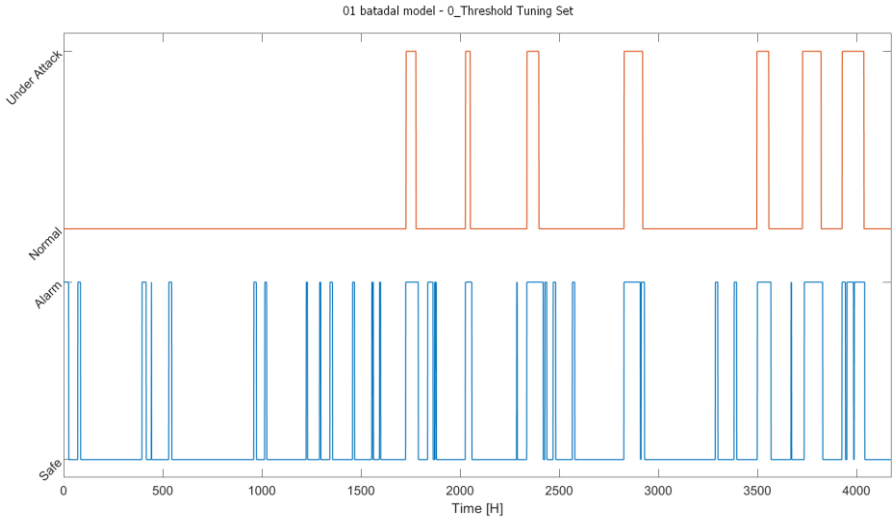


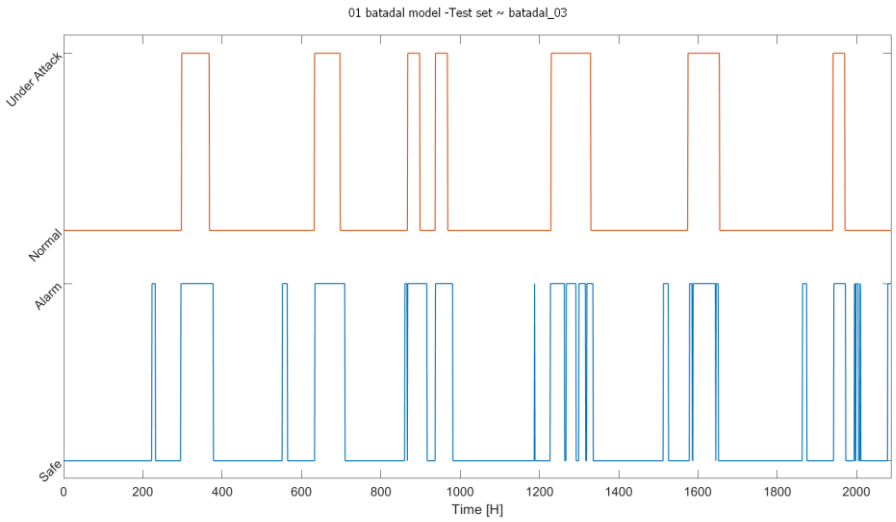
Attack: Alteration of T4 signal arriving at PLC6.
SCADA concealment: -

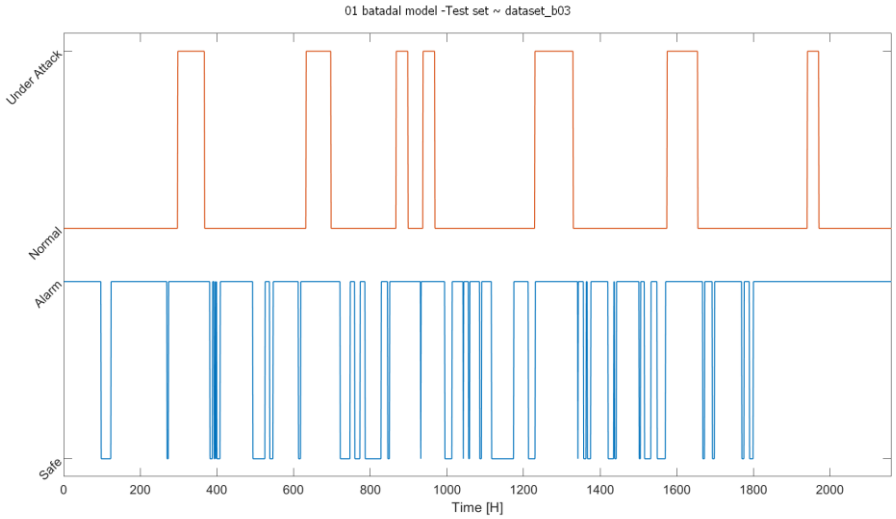


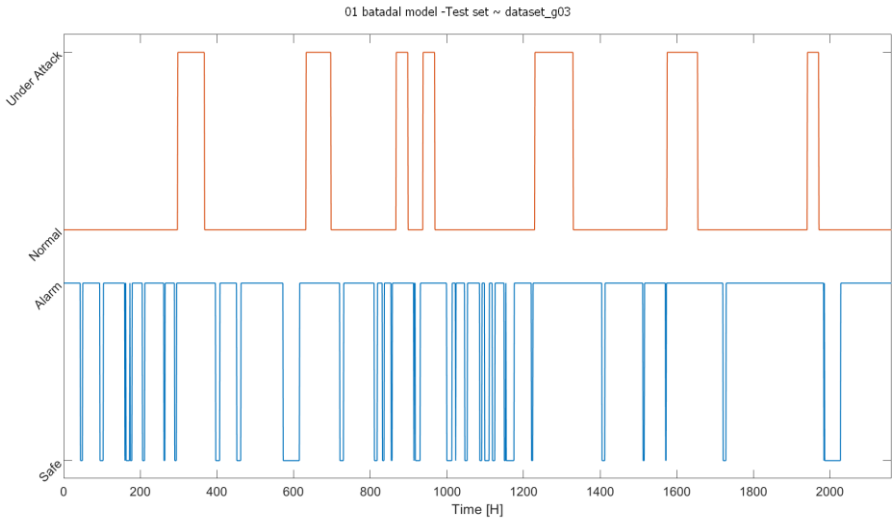
Attack: Alteration of T4 signal arriving at PLC6.
SCADA concealment: -

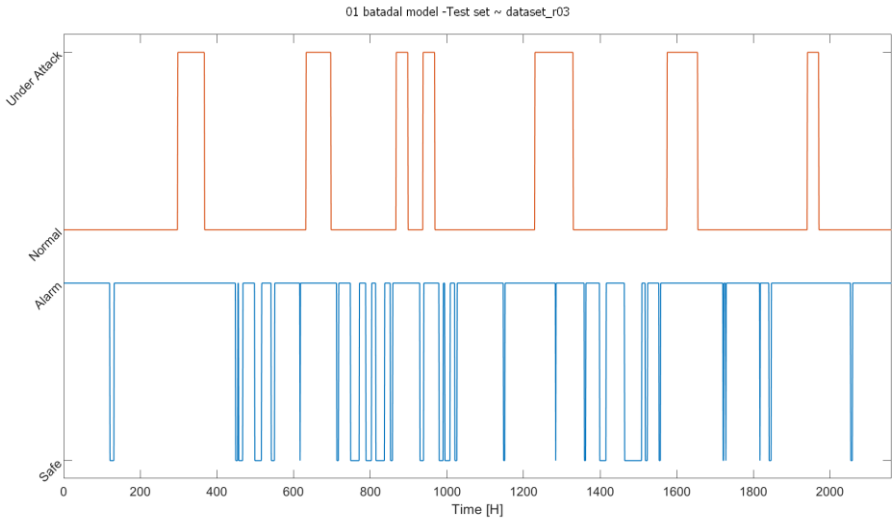
SVDD detection trajectories

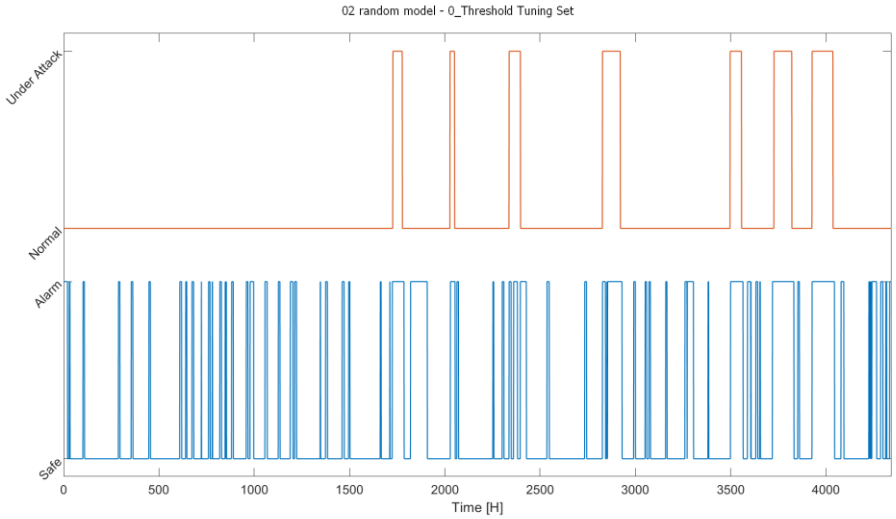


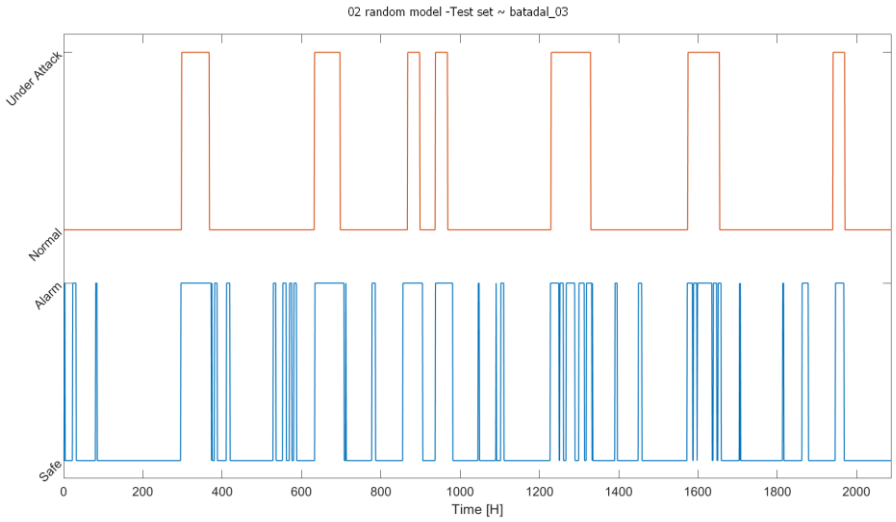


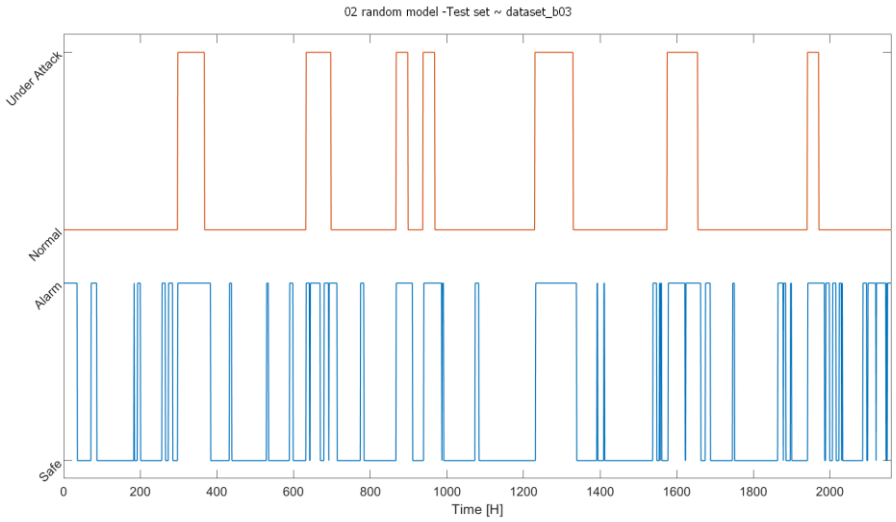

01 batadal model -Test set ~ dataset_b03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

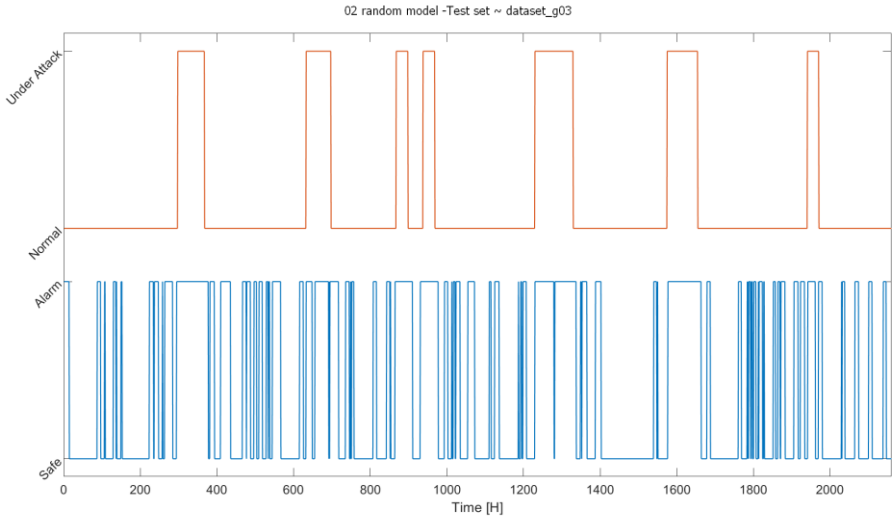

01 batadal model -Test set ~ dataset_g03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

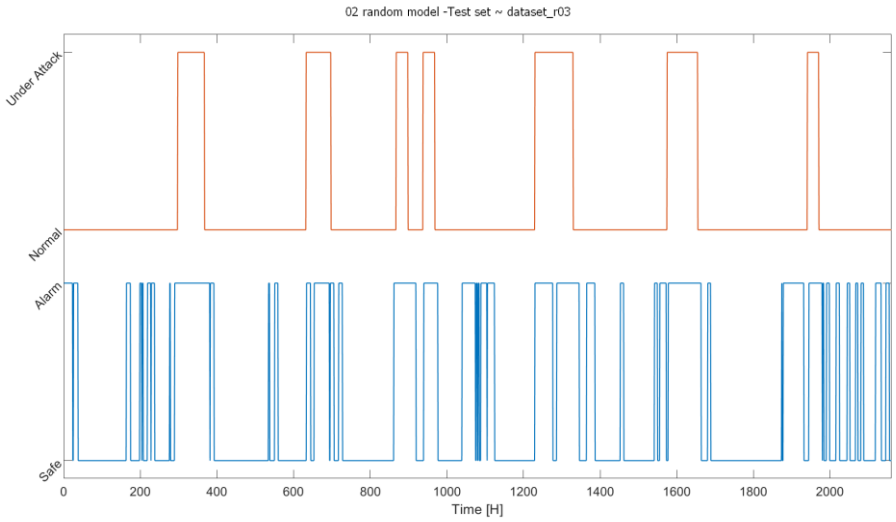

01 batadal model -Test set ~ dataset_r03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

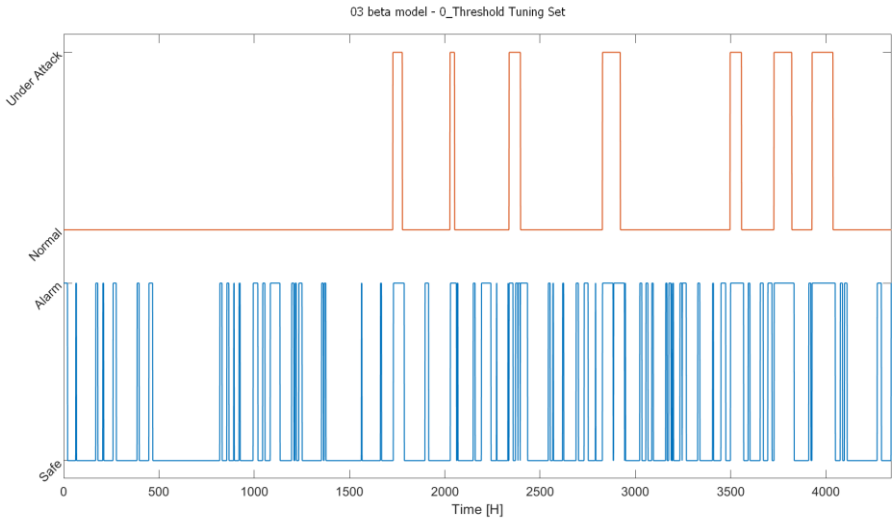

02 random model - 0_Threshold Tuning Set
Under Attack
Normal
Alarm
Safe
0
500
1000
1500
2000
2500
3000
3500
4000
Time [H]

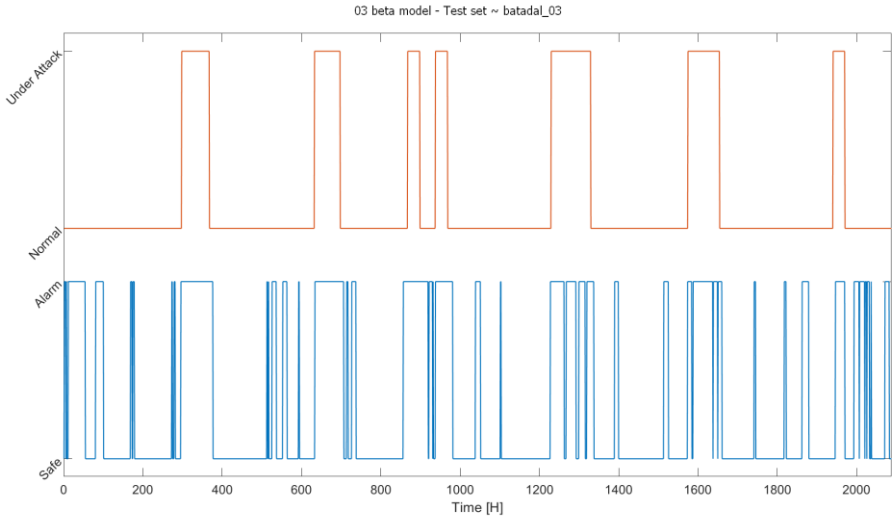

02 random model -Test set ~ batadal_03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

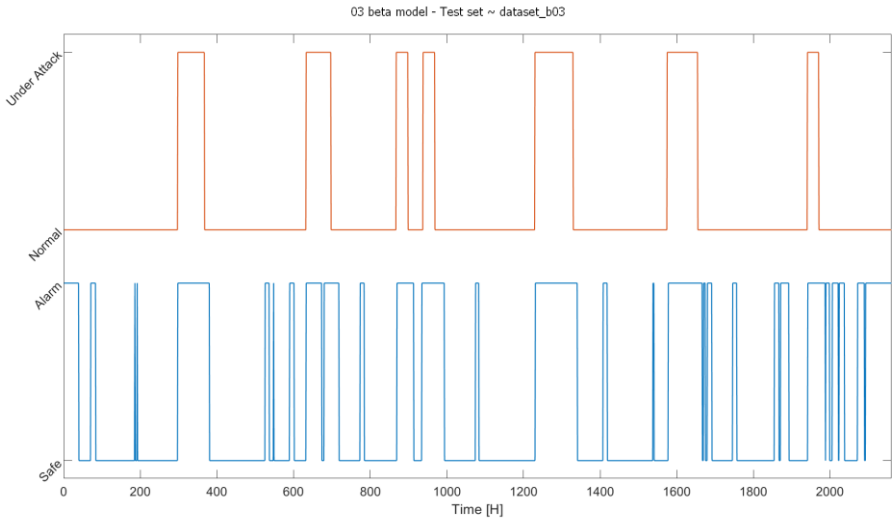

02 random model -Test set ~ dataset_b03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

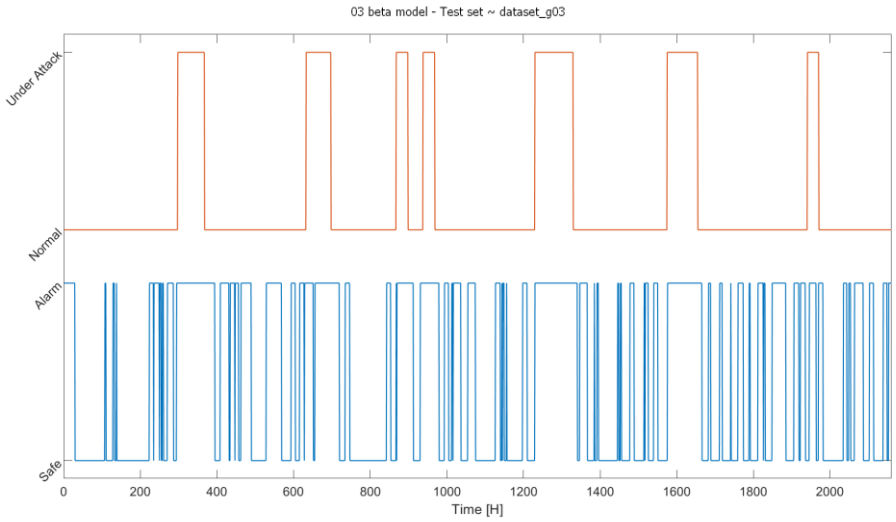

02 random model -Test set ~ dataset_g03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

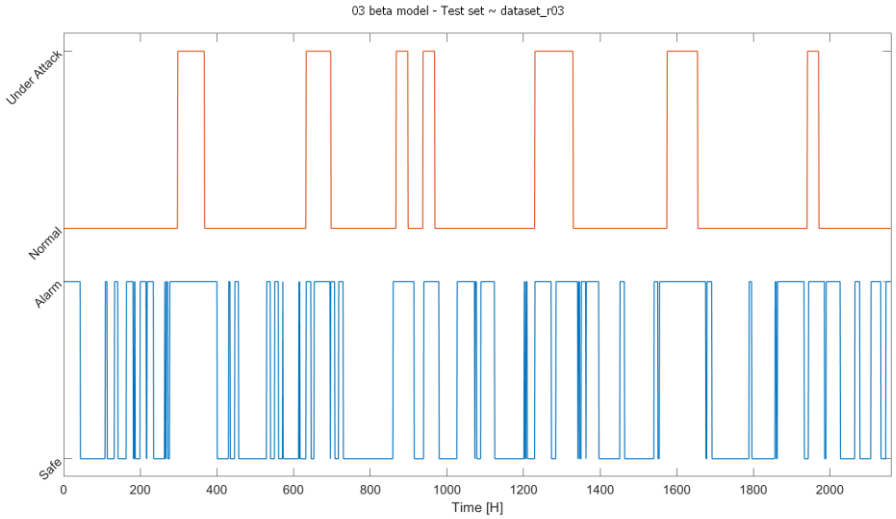

02 random model -Test set ~ dataset_r03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

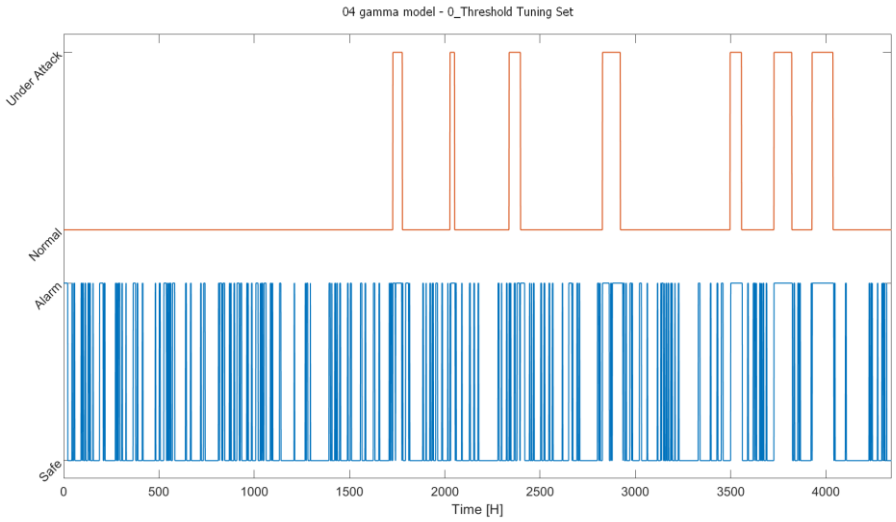

03 beta model - 0_Threshold Tuning Set
Under Attack
Normal
Alarm
Safe
0
500
1000
1500
2000
2500
3000
3500
4000
Time [H]

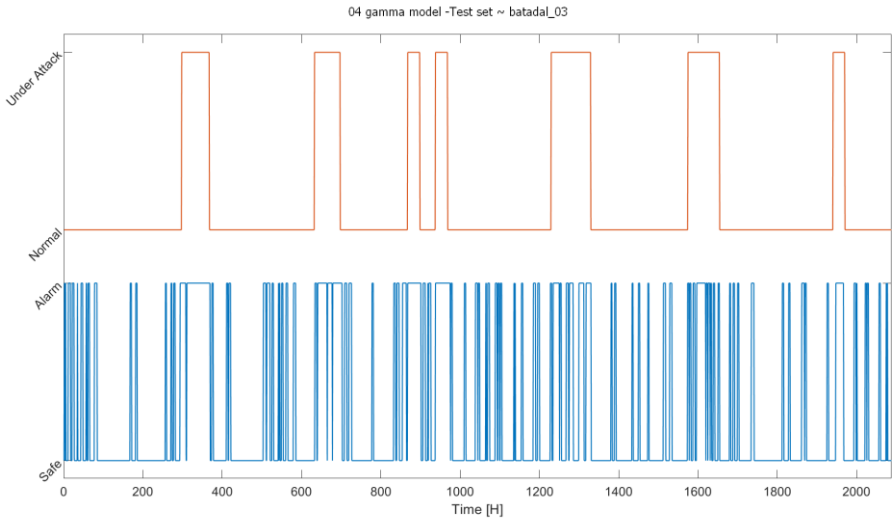

03 beta model - Test set ~ batadal_03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

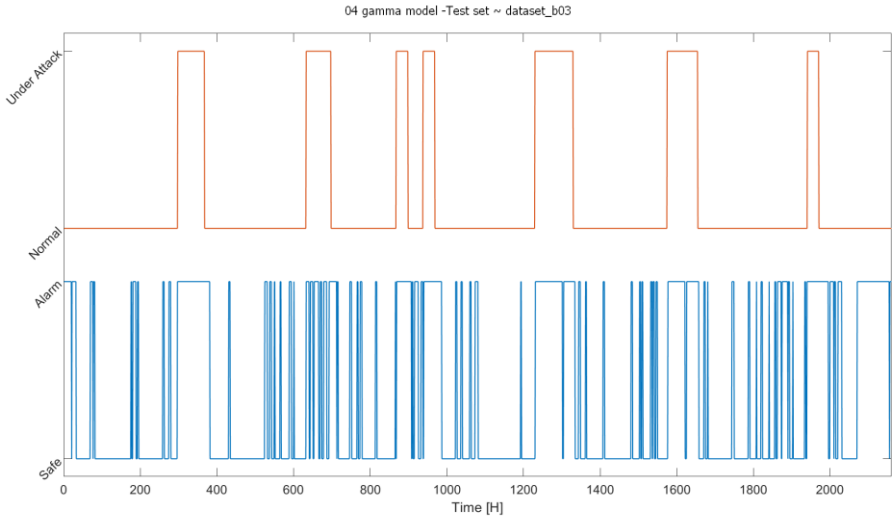

03 beta model - Test set ~ dataset_b03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

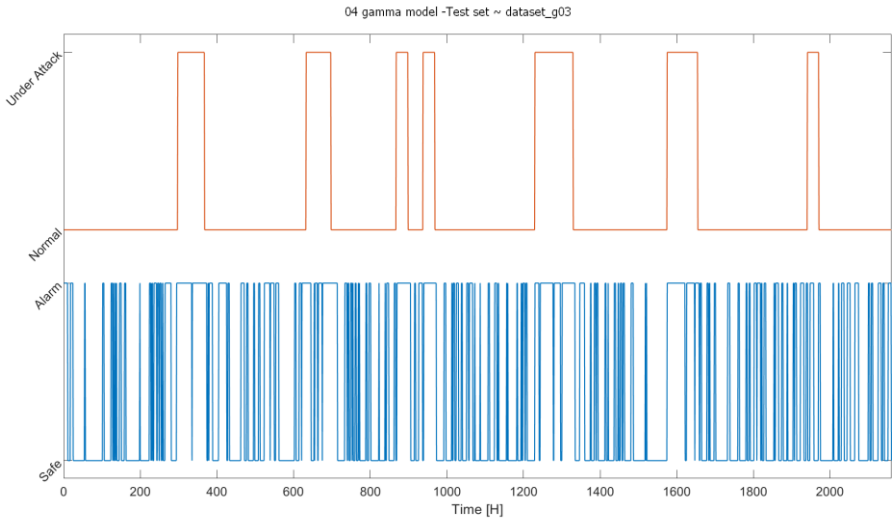

03 beta model - Test set ~ dataset_g03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

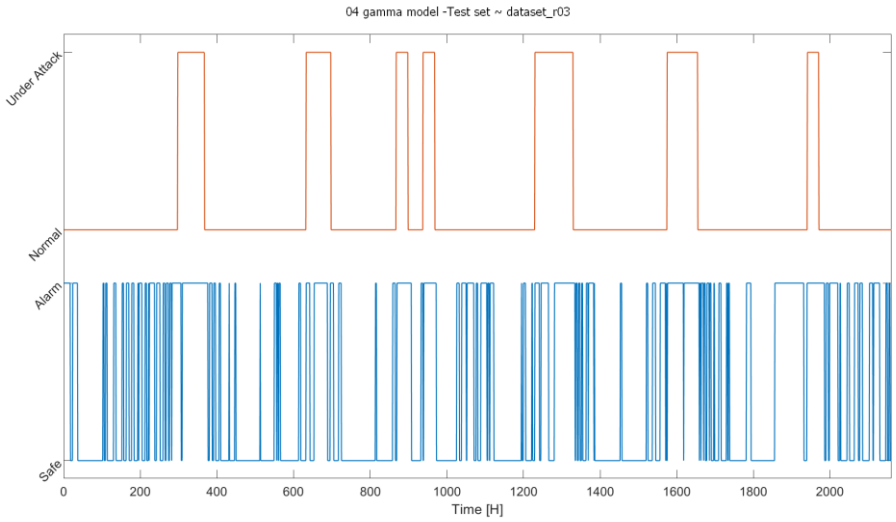

03 beta model - Test set ~ dataset_r03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]


04 gamma model - 0_Threshold Tuning Set
Under Attack
Normal
Alarm
Safe
0
500
1000
1500
2000
2500
3000
3500
4000
Time [H]


04 gamma model -Test set ~ batadal_03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

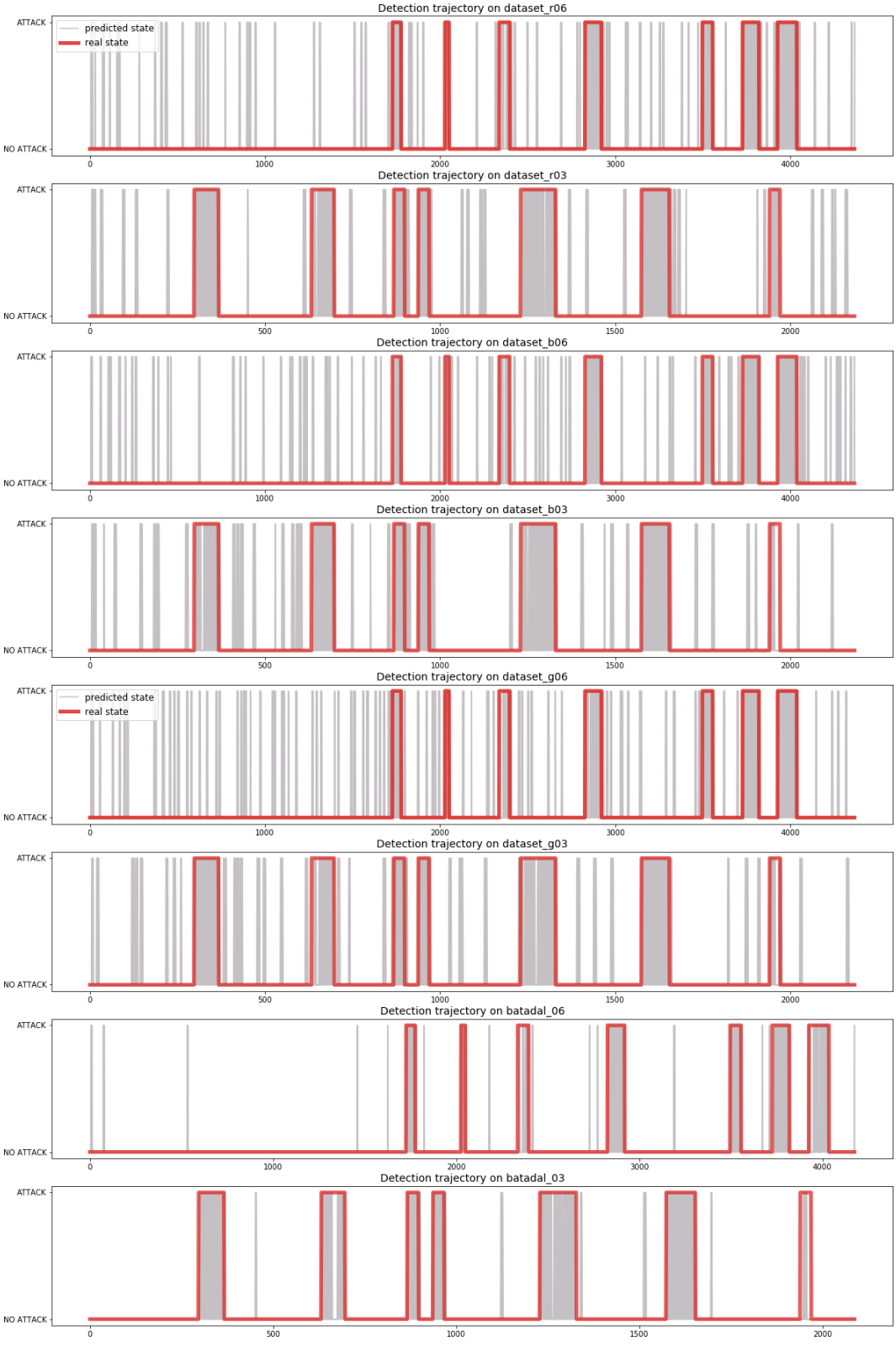

04 gamma model -Test set ~ dataset_b03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]


04 gamma model -Test set ~ dataset_g03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

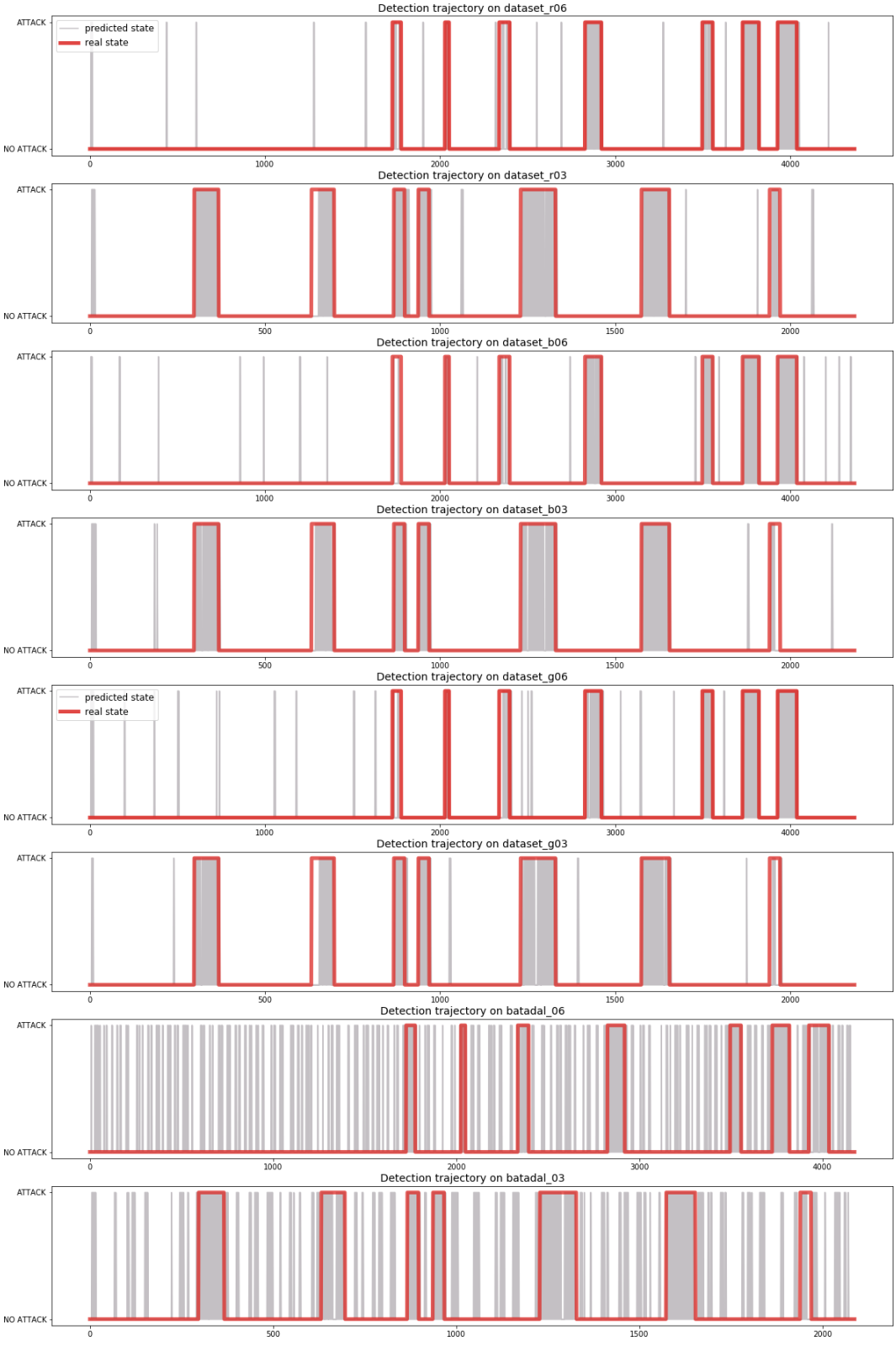

04 gamma model -Test set ~ dataset_r03
Under Attack
Normal
Alarm
Safe
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time [H]

Autoencoder detection trajectories

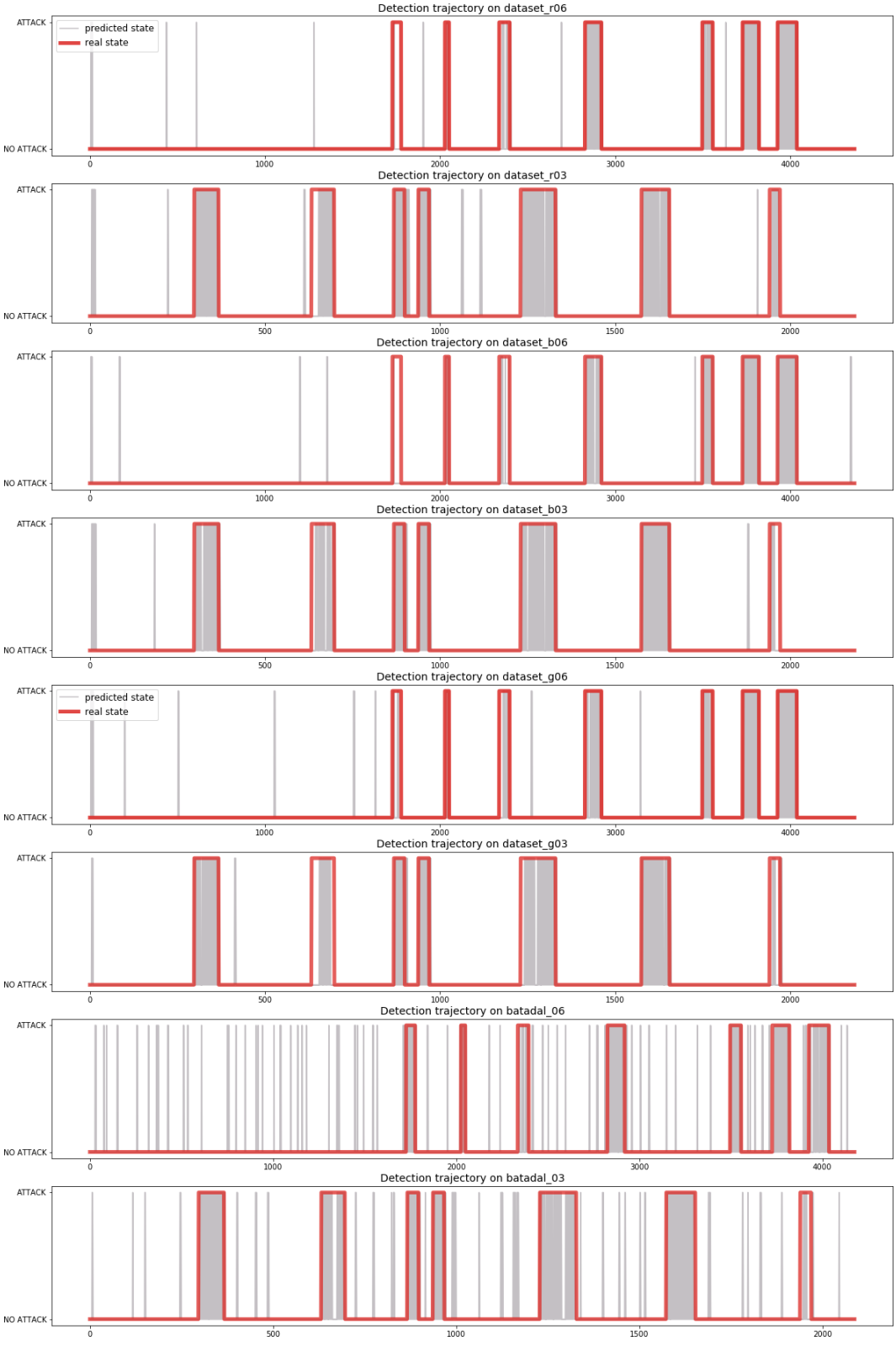
Train with batadal_12-tune threshold with batadal_06



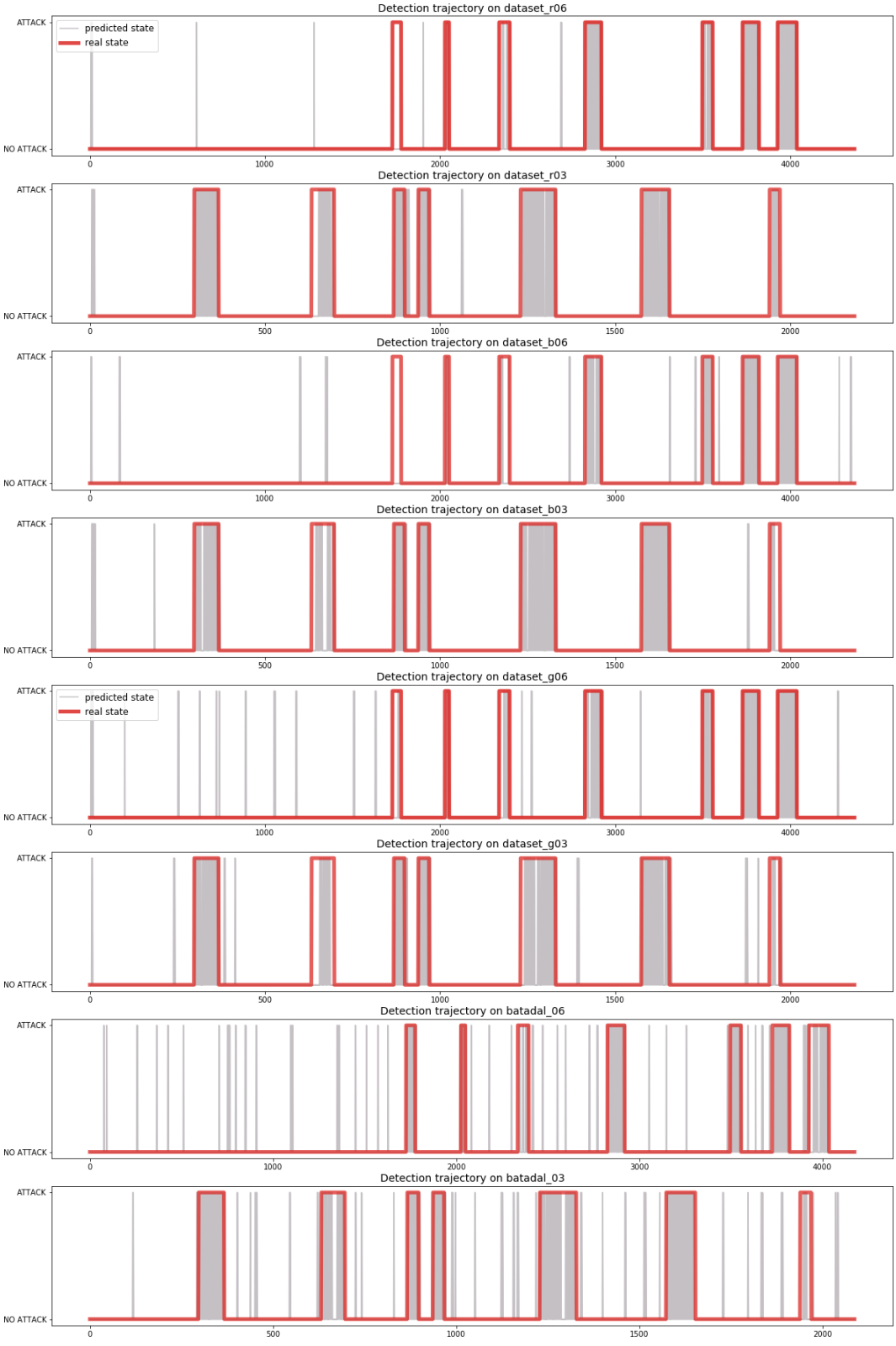
Train with dataset_b12 - tune threshold with dataset_b06



Train with dataset_g12 - tune threshold with dataset_g06

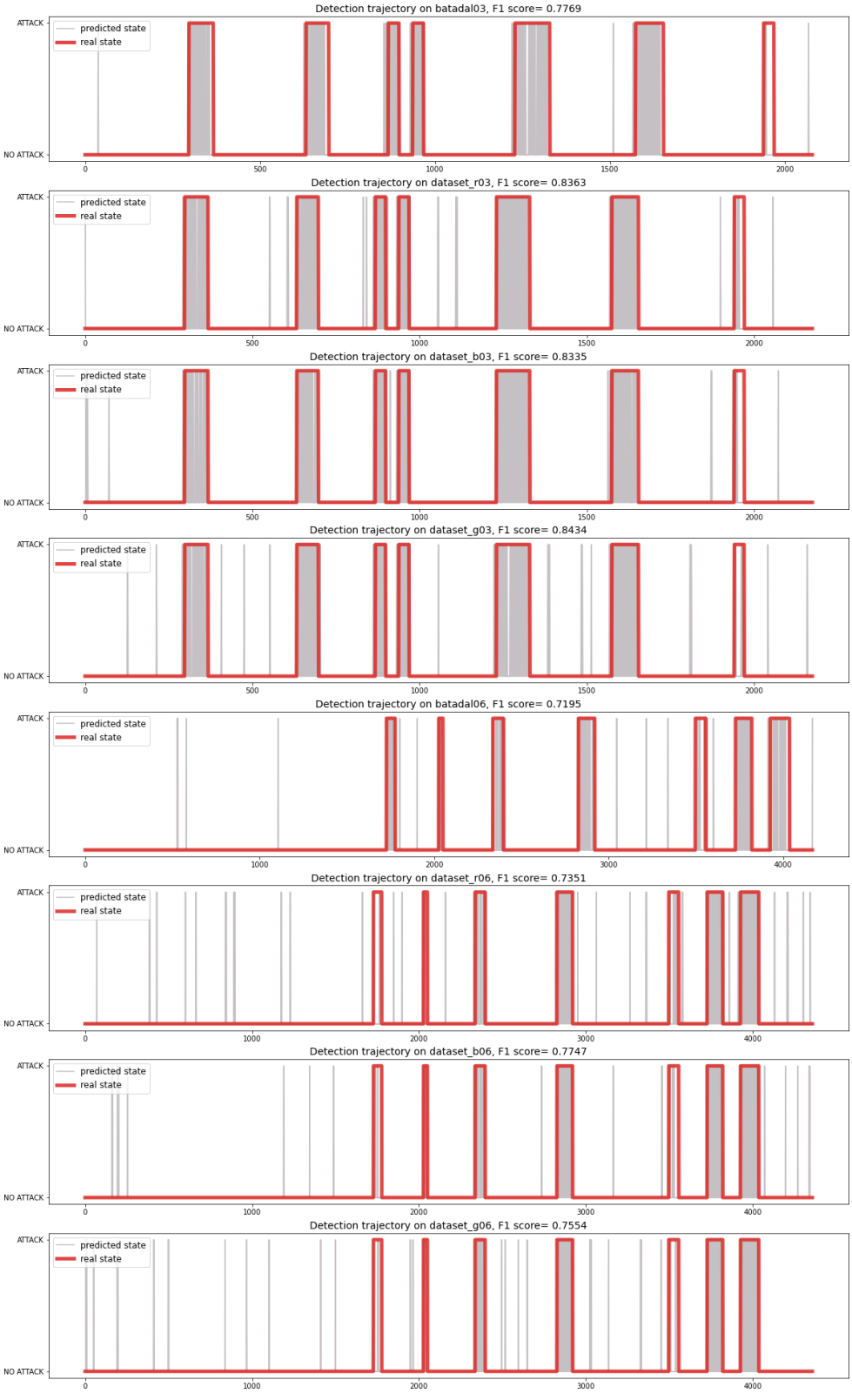


Train with dataset_r12 - tune threshold with dataset_r06

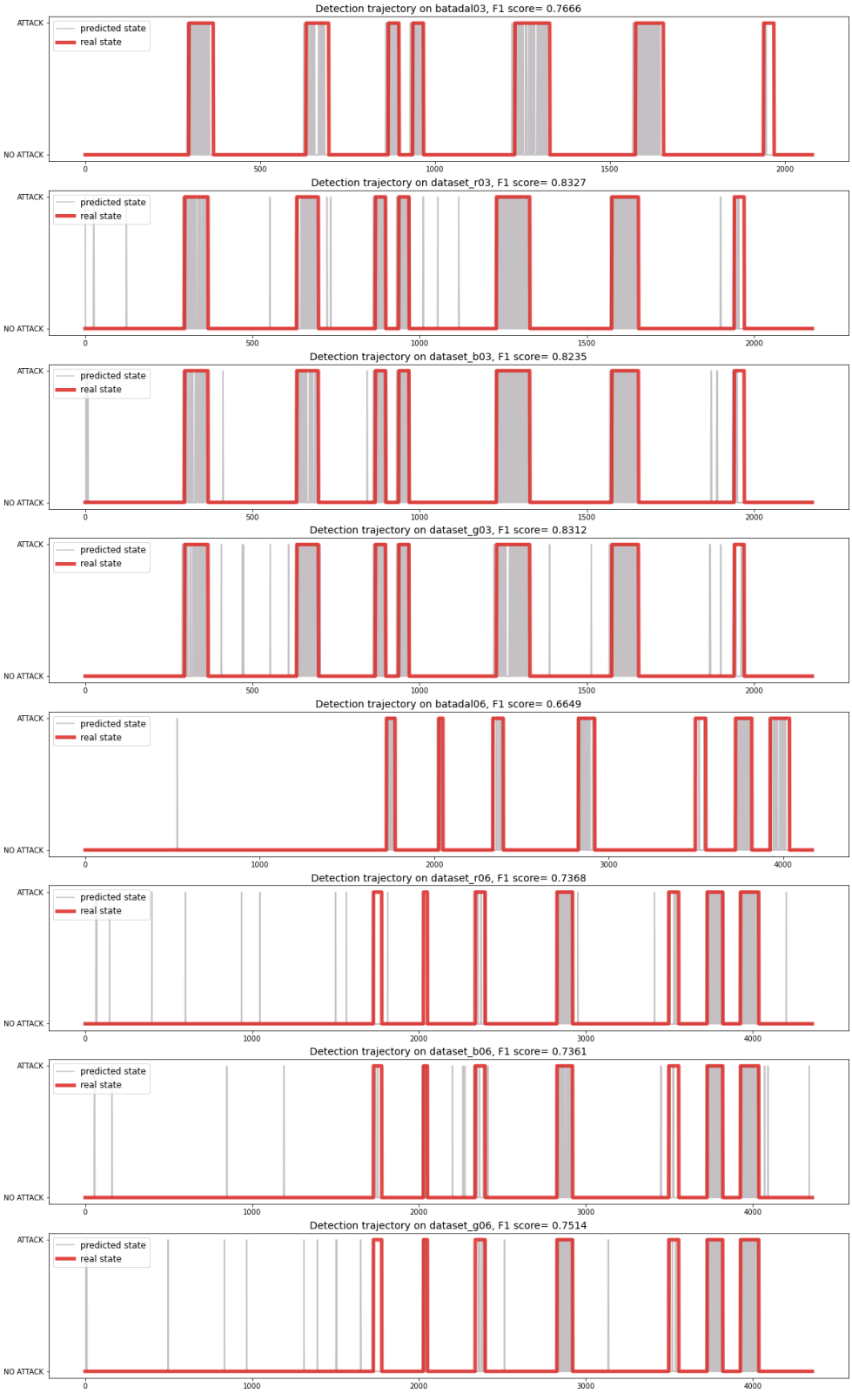


SCNN detection trajectories

Train with beta dataset



Train with gamma dataset



Train with random dataset