SAPIENZA
UNIVERSITÀ DI ROMA

Utilizzo di Sensor Noise Fingerprint per il Rilevamento di Cyber-Attacchi in CPS

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea in Informatica

Candidato

Andrei Laurentiu Lepadat

Matricola 1677093

Relatore

Prof. Enrico Tronci

Anno Accademico 2020/2021

Utilizzo di Sensor Noise Fingerprint per il Rilevamento di Cyber-Attacchi in CPS

Tesi di Laurea. Sapienza – Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Versione: 28 novembre 2021

Email dell'autore: lepadat.1677093@studenti.uniroma1.it

Ringrazio tutti i familiari e gli amici che mi sono stati vicini, che mi hanno incoraggiato quando pensavo di non farcela e che mi hanno sopportato quando non sapevo fare altro che parlare di università. Grazie.

Indice

Sommario **vii**

1 Introduzione **1**
1.1 Contesto 1
1.2 Motivazioni 1
1.3 Contributi 1
1.4 Struttura 1

2 Background **3**

3 Metodi **7**

4 Implementazione **9**

5 Risultati sperimentali **11**
5.1 Obiettivi 11
5.2 Configurazione 11
5.3 Casi di studio 12
5.4 Correttezza 12
5.5 Valutazione tecnica 15

6 Conclusioni **19**

Sommario

È proposta una procedura per autenticare i sensori e rilevare attacchi che cercano di minare l'integrità dei dati in Sistemi Cyber-Fisici (CPS). Questa tecnica si basa su [1]: si seguono i passi fondamentali e si adatta la metodologia in base ai casi di studio e alla tecnologia utilizzati. Lo schema proposto utilizza le caratteristiche hardware dei sensori e la fisica dei processi per creare dei modelli univoci per ogni sensore (da qui il termine *fingerprint*). Il fingerprint di un sensore è quindi una funzione del rumore di un sensore, intrinseco nelle misurazioni che i sensori compiono.

Si cerca quindi di simulare il comportamento di un sensore con difetti di fabbricazione, che contenga nelle sue misurazioni un errore. Il fingerprint di un sensore è generato in base alle misurazioni ottenuto durante il normale funzionamento del sistema. Si mostra che durante l'alterazione delle misurazioni di un sensore (ovvero durante l'injection di dati), il pattern del rumore di un sensore devia da quello che ha generato il fingerprint, permettendo quindi di rilevare attacchi.

Gli esperimenti sono eseguiti su due modelli differenti di autoveicoli. Un modello che simula le dinamiche di un'automobile a cambio automatico e un modello di un'automobile con controllo climatico automatico. Una classe di attacchi è ideata per mettere in difficoltà lo schema proposto e un'analisi dei risultati è dunque effettuata. Si vedrà il tempo minimo (relativamente al sistema considerato) necessario per identificare con maggiore precisione determinati attacchi.

1. Introduzione

1.1 Contesto

Un Sistema Cyber-Fisico è composto da vari componenti che sono in continua comunicazione tra di loro, come attuatori, sensori, controllori e reti di comunicazione. Un esempio di questi sistemi sono reti di distribuzione dell'energia elettrica, impianti di trattamento dell'acqua, veicoli autonomi, dispositivi medici impiantabili. I dati dei sensori sono trasmessi verso un controllore logico programmabile (PLC) installato nell'hardware del sistema che ha il compito di prendere decisioni appropriate sulla base delle misurazioni dei sensori.

1.2 Motivazioni

La comunicazione con l'esterno, oltre agli innumerevoli vantaggi di monitorazione e controllo che porta, sottopone a vari rischi la sicurezza del sistema [2]. Un abile avversario capace di intercettare e rimpiazzare (spoofing) dati nel dominio digitale o fisico può portare il sistema in stati non sicuri, con risultati anche catastrofici. Vi è quindi un livello, quello fisico, che può essere utilizzato per garantire una maggiore sicurezza.

Dunque, l'obiettivo non è quello di conservare la confidenzialià dei dati come nei tipici metodi della sicurezza informatica ma è quello di garantira l'integrità e l'affidabilità dei dati [4].

1.3 Contributi

Questo lavoro implementa lo schema descritto in [1], adattandolo e modificandolo a diversi sistemi di interesse in base a differenti scelte software, cercando di produrre risultati sperimentali concreti nei confronti di una determinata classe di attacchi.

1.4 Struttura

Nel Capitolo 2 vengono definite le basi teoriche utili a comprendere determinati concetti e scelte presenti nelle successive sezioni del lavoro. Le principali scelte metodologiche vengono descritte nel Capitolo 3 e il modo in cui queste sono adattate nel presente lavoro vengono definite nel Capitolo 4. Nel Capitolo 5 sono descritti le scelte tecnologiche, i sistemi di interesse, gli attacchi e i risultati sperimentali con

relativa analisi. Vengono infine tratte le conclusioni e valutati possibli lavori futuri come continuazione del presente nel Capitolo 6.

2. Background

Ogni sistema cyber-fisico che si rispetti è dotato di almeno un sensore che ha il compito di misurare una determinata "qualità" fisica di interesse per il sistema stesso. I dati che vengono rilevati dai sensori spesso vengono memorizzati localmente e/o in modo remoto e possono essere impiegati, come nel lavoro qui presentato, per fini paralleli o trasversali a quelli per cui sono stati installati. Una sequenza di dati estratti da sensori ordinata temporalmente viene chiamata *serie temporale* (*time-series* in inglese).

Comunemente i sensori sono imperfetti per costruzione e contengono intrinsecamente un'incertezza (*rumore*) nelle misurazioni che eseguono, dovuta a imperfezioni di fabbricazione. Sia

$$\bar{y}_k = y_k + \delta_k \tag{2.1}$$

il valore misurato da un determinato sensore nell'istante di tempo k, composto da y_k, il valore effettivo in quell'istante della grandezza misurata, più δ_k, il rumore aggiunto.

In un determinato istante di tempo, il valore di ogni sensore del sistema costituisce lo *stato* del sistema. La sfida di estrarre il fingerprint dai sensori è data dal fatto che questi stati sono dinamici. Prendendo in considerazione, per esempio, un termometro, se la temperatura dell'ambiente che misura rimane costante nel tempo è facile estrarre il fingerprint del rumore e costruirne il profilo, ma in processi reali non è così semplice, gli stati cambiano continuamente, per esempio l'aumento di velocità di una macchina per via della pressione sul pedale dell'acceleratore. È importante catturare queste variazioni affinché le misurazioni dinamiche dei sensori possano essere stimate. In [1] questo problema viene affrontato definendo un modello analitico del sistema interessato, rappresentato tramite il modello *State-Space*. Viene così definito il modello lineare tempo inviariante (LTI) del sistema, rappresentato dal sistema di equazioni

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + \vartheta_k, \\ y_k = Cx_k + \eta_k : \end{cases} \tag{2.2}$$

in cui $x_k \in \mathbb{R}^n$ rappresenta lo stato del sistema, $u_k \in \mathbb{R}^p$ l'input di controllo e ϑ_k il rumore al tempo k. $y_k \in \mathbb{R}_m$ e $\eta_k \in \mathbb{R}_m$ rappresentano, rispettivamente, la misurazione e il rumore del sensore al tempo k. A, B, C sono le matrici dello spazio di stato di dimensioni adeguate che rappresentato la dinamica del sistema.

Definito il precedente sistema, ci sono molti punti che un attaccante mal intenzionato potrebbe bersagliare. Nel lavoro presentato, così come in [1], vengono presi in considerazione *spoofing attack* ai sensori che potrebbero essere portati a termine

tramite uno schema *Man-in-The-Middle*. L'equazione lineare che rappresenta questa tipologia di attacchi è data da

$$\bar{y}_k = y_k + \delta_k = Cx_k + \eta_k + \delta_k,^1 \tag{2.3}$$

in cui $\delta_k \in \mathbb{R}_m$ rappresenta un attacco ai sensori.

In [1], dato l'output $\bar{y}_k$, viene adoperato il *filtro di Kalman* per stimare lo stato del sistema e il vettore dei *residui*, definito, in questo contesto, come la differenza tra la reale misurazine effettuata dal sensore e la stima della misurazione calcolata dal filtro nell'istante k:

$$r_k := \bar{y}_k - \hat{y}_k, \tag{2.4}$$

dove $\hat{y}_k$ è l'output del filtro di Kalman.

Detto ciò, per quantificare la bontà del modello del sistema, viene utilizzato l'*Errore Quadratico Medio (RMSE)*, definito come

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}. \tag{2.5}$$

Questa metrica rappresenta la distanza tra il valore stimato e quello misurato, ovvero quanto il primo è lontano dal secondo. Nella letteratura della teoria del controllo, modelli con un'accuratezza superiore al 70% sono considerati accettabili approssimazioni della dinamica di sistemi reali.[2]

Per ogni momento statistico (media, deviazione standard, ...) di una serie storica (ma non solo) si può definire un *intervallo di confidenza* che esprime la probabilità che il valore calcolato sugli N campioni della serie approssimi il valore effettivo del momento statistico. Questo intevallo, nel caso del valore medio, si definisce come

$$Pr\{\bar{x} - \epsilon \leq \mu \leq \bar{x} + \epsilon\} = 1 - \delta, \tag{2.6}$$

in cui μ e $\bar{x}$ sono, rispettivamente, la media effettiva e quella calcolata. ϵ e δ sono valori che dipendono da N, e mantenendo δ costante e incrementando N, anche ϵ cresce, allargando l'intervallo di confidenza. Tale intervallo può essere definito anche per momenti di ordine superiore.

Viene ora data una definizione basilare di cosa è un problema di *Machine Learning*. Un problema di M.L. può essere definito come una funzione

$$f : X \to Y, \tag{2.7}$$

dato un insieme D (dataset) contenente informazioni riguardanti f. Fare il *learning* della funzione f significa trovare un'altra funzione $\hat{f}$ che approssima e ritorna valori più vicini possibile ad f, specialmente per elementi non presenti in D. Nel presente lavoro il problema viene definito come un problema di *classificazione*, ovvero in cui f è definita tale che

$$\begin{aligned} X &:= \mathbb{R}^m, \\ Y &:= \{C_1, C_2, \ldots, C_k\}; \end{aligned} \tag{2.8}$$

[1] Notare l'ugualianza con l'equazione 2.1: un attacco è considerato come un'introduzione di rumore nella misurazione fatta da un sensore.

[2] Se l'RMSE rappresenta un errore, l'accuratezza viene calcolata come 100 - RMSE.

e *supervised*, cioé in cui il datasert è del tipo

$$D = \{(x_i, y_i)_{i=1}^{N}\}, x_i \in X, y_i \in Y \tag{2.9}$$

Quindi f associa ad ogni elemento di X, cioè un vettore di m reali, un elemento di Y, cioè la classe C_i di appartenenza.

Ogni elemento $x_i \in X$ è composto da uno o più valori ed ognuno di questi valori definisce una proprietà, o *feature* utilizzando il terminre più preciso, di x_i. Queste feature sono definite, localmente a questo lavoro, come alcuni valori statistici (di cui si è parlato precedentemente), mostrati nella Tabella 3.1, estratti dai vettori dei residui.[3]

Sia $\hat{f}$ la funzione che approssima f. L'errore di $\hat{f}$ rispetto ad $S \subset X$ è uguale alla proporzione con cui $\hat{f}$ classifica erroneamente i sample in S, ovvero

$$error_S(\hat{f}) = \frac{1}{n} \sum_{x \in S} \phi(\hat{f}(x), f(x)), \tag{2.10}$$

dove

$$\phi(\hat{f}(x), f(x)) = \begin{cases} 0 & \text{se } \hat{f}(x) \neq f(x) \\ 1 & \text{altrimenti} \end{cases} \tag{2.11}$$

L'accuratezza di $\hat{f}$ è ugale a

$$accuracy(\hat{f}) = 1 - error(\hat{f}). \tag{2.12}$$

In seguito, questa metrica verrà utilizzata spesso per determinare la bontà del classificatore trovato.

[3]Il vettore dei residui è quindi parte fondamentale per la definizione dei fingerprint dei sensori.

3. Metodi

Nel contesto del presente lavoro, come si vedrà, volendo giudicare la legittimità delle misurazioni di un determinato sensore, determinate proprietà statistiche delle nuove misurazioni (nuove nel contesto di normale funzionamento del sistema *aperto* ad attacchi) verranno confrontate con le stesse proprietà di misurazioni effettuate in condizioni *sicure* (questi valori sono chiamati valori di *reference*). Per le nuove misurazioni, prendendo ancora in esempio il valore medio e volendo avere un intervallo di confidenza il più piccolo possibile (quindi un ϵ il più piccolo possibile), bisogna essere attenti per via di valori di N non molto grandi, caratteristica preferibile in quanto non si vogliono campionare troppi valori in situazioni real-time (equivarrebbe ad aspettare di più per prendere una decisione, e quindi essere potenzialmente per più tempo sotto attacco). Scegliere un'intervallo di condifenza delle nuove misurazioni troppo grande potrebbe portare ad una sovrapposizione tra il nuovo intervallo ed quello di reference.[1] Per avere una sensibilità migliore contro gli attacchi il problema viene affrontato avvalendosi dell'aiuto di un determintao modello di Machine Learning, che ha un buon comportamento verso valori che non sono perfettamente discriminabili.

Il prossimo passo è quello di definire correttamente il problema di M.L. ed utilizzare un algoritmo di learning adeguato. A questo scopo bisogna preparare i dati estratti dai sensori (e di conseguenza il vettore dei residui) per poter essere utilizzati efficacemente dall'algoritmo di learning.

Il dataset contiene coppie in cui il primo elemento x_i è un vettore che contiene le feature elencate nella Tabella 3.1, e il secondo elemento y_i è la classe di apparteneza che rappresenta il sensore che ha prodotto le misurazioni utilizzate per costruire x_i.

Data la natura statistica delle feature utilizzate, sarebbe inadeguato estrarre le stesse basandosi solamente sulla misurazione avvenuto all'istante di tempo k.[2] Detto ciò, bisogna partizionare ogni serie temporale in blocchetti (*chunk*) di dimensione d. Da ogni chunk verranno poi estratte le feature precedentemente menzionate ed etichettate con la classe (quindi sensore) di appartenenza. Scegliere la giusta dimensione d dei vari chunk è una scelta sensibile in quanto forzerà anche la dimensione dei chunk composti da residui di sensori che verranno calcolati mentre il sistema è in uno stato in cui potrebbe essere soggetto ad attacchi. L'idea è di avere chunk di serie temporali abbastanza grandi da catturare la dinamica del processo e piccoli abbastanza da ridurre il tempo di attesa per discriminare nuove misurazioni.

[1] Il caso ideale sarebbe o di inclusione del primo nel secondo o di disgiunzione, rilevando nel primo caso un valore ammesso e nel secondo caso un attacco.

[2] Estrarre delle feature come media e varianza da un'unica misurazione non ha senso in quanto la media equivarrebbe al valore stesso e la varianza sarebbe necessariamente nulla.

Feature	Descrizione
Media	$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
Varianza	$\sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$
Dev. Med. Ass.	$D_{\bar{x}} = \frac{1}{N}\sum_{i=1}^{N}\lvert x_i - \bar{x}\rvert$
Asimmetria	$\gamma = \frac{1}{N}\sum_{i=1}^{N}(\frac{x_i-\bar{x}}{\sigma})^3$
Curtosi	$\beta = \frac{1}{N}\sum_{i=1}^{N}(\frac{x_i-\bar{x}}{\sigma})^4 - 3$

Tabella 3.1. Lista delle feature utilizzate; x è la serie temporale di dimensione N proveniente dal sensore.

Tutte le istanze di X che appartengono allo stesso sensore determinano il fingerprint del sensore e il primo compito della funzione f (Funzione 2.7) è quello di distinguere correttamente i sensori, quindi generare un fingerprint il più preciso e privo di ambiguità possibile. Nel Capitolo 5, l'esistenza del fingerprint verrà provata empiricamente.

L'algoritmo di M.L. utilizzato è il *Support Vector Machine* (SVM) [5], impiegato come un classificatore a 2 classi (One vs. One) che etichetta i dati provenienti dal legittimo sensore (*ground truth*) come un *1* e come *0* i dati provenienti dagli altri sensori. In questo modo gli attacchi possono essere considerati come un tipo dato appartenente ad altre classi.

L'algoritmo SVM verrà allenato su un campione di dati che dipenderà dal sistema considerato e verrà validato usando la tecnica della *k-fold cross validation* [7]. Successivamente, siccome nel dataset non sono presenti dati relativi ad attacchi, il classificatore a 2 classi potrebbe mancare alcuni degli attacchi.

Per affrontare questo problema viene utilizzato la modalità di SVM ad una classe (*one-class* SVM) per fare il learning del modello di M.L., basandosi su una classe di dati normali (appartenenti quindi ad un solo sensore). Nella fase di testing tutto ciò reputato estraneo viene considerato come un attacco.

Facendo dunque il punto della situazione, il problema di trovare la giusta dimensione dei chunk è affrontato come un problema di classificazione cercando di creare un fingerprint dei sensori il più preciso possibile. In questa fase dati appartenenti ad attacchi non sono conosciuti, e di conseguenza, usare il precedente modello potrebbe portare a farsi sfuggire alcuni (se non molti) attacchi. Per ovviare a questo problema, viene adoperato un classificatore ad una classe che, avendo come mezzo di confronto il fingerprint (di un determinato sensore), ha più possibilità di rilevare possibili *outlier*.

4. Implementazione

Alla luce di quanto detto nel Capitolo 2 per quanto riguarda l'estrazione dei vettori dei residui, nel presente lavoro viene presa una strada alternativa e più semplice di quella presentata in [1].

Viene utilizzato un modello di sistema dinamico ideale, che non produce a livello delle misurazioni dei sensori alcun tipo di rumore. Successivamente, il rumore viene inserito "artificialmente", sotto forma di rumore gaussiano.

Sia y_k l'output di un sistema del tipo definito in 2.2 all'istante k e siano $\mathcal{X}_j \sim \mathcal{N}(0, \sigma_j^2)$, $j = 1, \ldots, m$,[1] m (una per sensore) distribuzioni di probabilità gaussiane con media nulla e varianza σ_j^2. Il valore in output del sistema dopo l'introduzione di rumore è uguale a

$$\bar{y}_k = y_k + \delta_k, \tag{4.1}$$

dove δ_k è il vettore che contiene le m estrazioni secondo $\mathcal{X}_j$ all'istante k.

Il vettore dei residui viene quindi ottenuto eseguendo la sottrazione vettoriale $r_j = \bar{y}_j - y_j$, in cui y_j (la serie storica, ovvero la collezione, delle misurazioni relative al sensore j all'istante n) agisce come il valore stimato secondo le dinamiche del sistema. Dal momento che ogni r_j deve contenere valori di un sistema reale sottoposto a sensor noise, le varianze σ_j^2 devono essere tali che le distribuzioni $\mathcal{X}_j$ generino valori per $j = 1, \ldots, m$, dalla 2.5, tali che

$$100 - \sqrt{\frac{\sum_{k=1}^{n} (\bar{y}_{j,k} - y_{j,k})^2}{n}} \geq D_j, \forall j \in \{1, \ldots, n\}. \tag{4.2}$$

D_j è un valore sempre maggiore o uguale di 70 e, in base al ruolo del sensore all'interno del sistema, può assumere valori più o meno grandi. Sensori che hanno compiti sensibili devono essere "di buona fattura" e cercare di ridurre al minimo l'errore introdotto nelle misurazioni (qualità che soliatamente aumenta, e a volte non di poco, il costo del sensore). Sostituendo dall'Equazione 4.1, si nota che l'RMSE per ogni sensore j dipende banalmente dalla serie di valori estratti secondo $\mathcal{X}_j$.

Calcolati i vettori dei residui e preparati (divisi in chunk ed estratte le feature) per essere processati dall'algoritmo SVM, e fatto il learning del modello di M.L., quest'ultimo viene validato tramite la k-fold cross validation con $k = 2, \ldots, 15$.[2]

[1] m è la dimensione di y_k, uguale al numero di sensori del sistema.

[2] Questa operazione, con un k variabile, ha una duplice funzionalità: quella di validare il modello di M.L. trovato e quella di capire quanto il modello è sensibile allla quantità di dati utilizzati nella fase di learning.

Questo significa, per valori di k crescenti, il dataset D (per come è stato definito nel Capitolo 3) viene suddiviso in k partizioni e di queste, $k-1$ vengono utilizzate per fare il training e una per il testing. In questo modo si cerca di evitare problemi di sovradattamento, tipico della suddivisione del dataset in due partizioni. Per ogni k, l'accuratezza finale del modello è data dalla media delle accuratezze calcolate durante le k fasti di test. Dato che si vuole un modello di M.L. robusto nella dimensione del dataset, l'accuratezza di ognuna delle $k-1$ ($k = 2, \ldots, 15$) esecuzioni della k-fold cross validation viene confrontata con le altre, e minore la variazione maggiore la robustezza del modello.

5. Risultati sperimentali

Descritte le basi teoriche e metodologiche prese in considerazione, queste sono state implementate e testate per via software. In questo capitolo vengono delineati gli obiettivi della fase sperimentale, il software utilizzato per simulare i sistemi considerati e che quindi produce i valori di output che sono punto centrale di questo lavoro. Vengono presentati di conseguenza due modelli utilizzati come banco di prova per le tecniche descritte nel Capitolo 3.

5.1 Obiettivi

Gli obiettivi principali di questa fase sono molteplici.

- Trovare la giusta varianza del rumore gaussiano introdotto nei sensori.
- Mostrare l'esistenza del fingerprint.
- Trovare la giusta dimensione dei chunk in cui saranno divisi i vettori dei residui.
- Mostrare che la quantità di dati di sensori acquisita non incide particolarmente sul fingerprint.
- Definire la tipologia di attacchi che possono essere identificati e quindi valutare la precisione con cui vengono riconosciuti.

Come si vedrà, ognuno di questi punti è strettamente legato agli altri e mostrare la correttezza di uno implica poter proseguire con il successivo.

5.2 Configurazione

Il software che implementa le funzionalità qui presentate è stato scritto utilizzando diversi linguaggi di programmazione, tra cui principalmente Python e MATLAB. Il software utilizzato per ottenere dati provenienti da simulazioni di sistemi cyber-fisici è Simulink, un software per la simulazione e la modellazione di sistemi dinamici. La libreria sfruttata per quanto riguarda gli algoritmi Support Vector Machine è scikit-learn [6], basata su LibSVM [3].

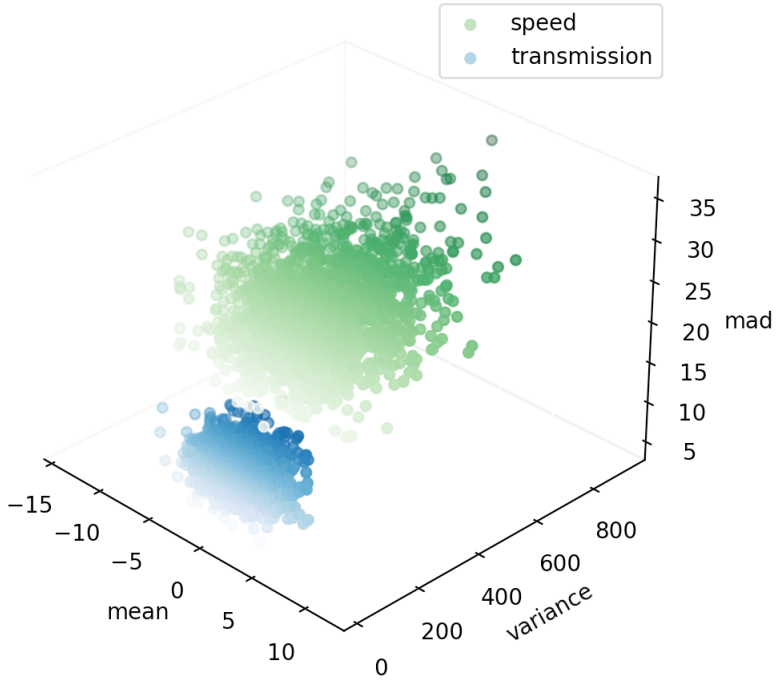

Figura 5.1. Esistenza del fingerprint per il modello in Figura 5.2.

5.3 Casi di studio

I due modelli di cui si è parlato rappresentano entrambi sottosistemi presenti in sistemi di trasporto.

- Modello di un'automobile a tramissione automatica (Figura 5.2);
- Modello di un sistema di controllo automatico del clima dell'ambiente interno di un'automobile (Figura 5.3).

Nei diagrammi dei due modelli, i rispettivi segnali importanti in questa fase sono stati marcati con un numero (ID del sensore) inscritto in un cerchio rosso e sono rispettivamente, in ordine di apparizione, per il primo modello velocità del veicolo e velocità della trasmissione, e per il secondo modello le temperatura dell'abitacolo, la temperatura emessa dal climatizzatore e la temperatura t, che è prodotta come somma dei contributi di alcuni valori non rilevanti in questo contesto.

5.4 Correttezza

Si vuole ora provare alcuni dei punti anticipati nella Sezione 5.1.

Alla luce di quanto detto nella Sezione 5.1, per quanto riguarda l'introduzione del rumore gaussiano nei sensori, sono state trovate empiricamente le varianze delle distrubizioni tale che valga l'Equazione 4.2. Nella Tabella 5.3 possiamo vedere i valori per entrambi i modelli.

Dimensione Chunk / Sensore	1	2
5	0.879	0.772
10	0.946	0.909
15	0.960	0.945
20	0.978	0.977
25	0.992	0.988
30	0.996	0.992
35	1.000	0.995
40	0.997	1.000
45	1.000	1.000
50	1.000	0.993
55	1.000	1.000

Tabella 5.1. Dimensione dei chunk differenti e accuratezza del classificatore per il modello di un'automobile a cambio automatico.

Dimensione Chunk / Sensore	1	2	3
5	0.785	0.561	0.748
10	0.850	0.723	0.906
15	0.882	0.793	0.963
20	0.910	0.855	0.979
25	0.911	0.894	0.990
30	0.940	0.930	0.995
35	0.958	0.930	0.996
40	0.969	0.946	0.998
45	0.976	0.957	1.000
50	0.980	0.969	1.000
55	0.984	0.984	1.000
60	0.991	0.985	1.000
65	0.986	0.980	0.998
70	0.991	0.983	0.998
75	0.993	0.983	1.000
80	0.998	0.993	1.000
85	0.995	1.000	1.000
90	0.997	0.995	1.000
95	1.000	0.997	1.000
100	0.997	0.991	1.000

Tabella 5.2. Dimensione dei chunk differenti e accuratezza del classificatore per il modello del controllo automatico del clima di un'automobile.

I Modello			II Modello			
Sensore	1	2	**Sensore**	1	2	3
Varianza	390	100	**Varianza**	9.2	4.2	1
(1 - RMSE)*100	83.5%	92.0%	**(1 - RMSE)*100**	96.4%	97.2%	99.0%

Tabella 5.3. Valori della varianza delle distribuzioni associate ai sensori dei due modelli considerati.

Nella Figura 5.1 si può vedere invece la rappresentazione grafica delle feature statistiche dei vettori dei residui dei due sensori che appartengono al modello in Figura 5.2. È facile in questo modo intuire il significato del fingerprint, questo ci dà un'idea di come il profilo del rumore sia una peculiarità di ogni sensore. Nell'esempio proposto vengono utilizzate tre feature, nello specifico media, varianza e deviazione media assoluta.

Come già detto in precedenza, per catturare le giuste dinamiche del sistema, vi è la necessità di suddividere il dataset in blocchi, e questi non devono avere una dimensione tale da non allungare eccessivamente l'intervallo di tempo che deve passare affinché un attacco venga identificato.

Nel caso del primo modello, il modello SVM viene allenato e validato con dataset composti da feature estratte da chunk che hanno una dimensione variabile e crescente tra un'esecuzione e la successiva. La dimensione dei chunk va da un minimo di 5 fino ad un massimo di 55 misurazioni, con un incremento di 5. Come si può vedere dalla Tabella 5.1 l'accuratezza del classificatore converge verso valori molto vicini ad 1 per chunk di dimensioni maggiore di 25. Un valore ragionevole potrebbe essere uno tra 25 e 30 in quanto hanno un buonissimo livello di accuratezza ma mantengono una dimensione abbastanza ridotta. La scelta che è stata fatta è quella di prendere come dimensione dei blocchi 25, in quanto il modello ha un sample time di $\frac{1}{25}$ secondi e questo permetterebbe di estrarre le feature dal chunk ogni secondo. Se fosse stata scelta una dimensione di 30 avremmo ottenuto risultati poco differenti.

Nel caso del secondo modello, il modello SVM viene allenato e validato con una dimensione dei chunk che va 5 a 100, con un incremento di 5. I risultati sperimentali sono mostrati nella Tabella 5.2 e si può notare che il classificatore raggiunge un'accuratezza per chunk con una dimensione di 80. Dato che il sample time del modello è di 1 secondo, significa che per poter estrarre le feature dai chunk bisogna attendere 80 secondi.

Ora, con l'obiettivo di mostrare il comportamento del classificatore al variare della dimensione del dataset si valutano i risultati forniti dalla cross validation. Entrambi i modelli vengono simulati per un determinato tempo per permettere la raccolta dei dati. Anche in questa situazione bisogna trovare un giusto compromesso in quanto si cerca di raggiungere tutti gli stati del sistema affinché la probabilità che durante il periodo di vita del sistema (in cui si potrebbero verificare attacchi) ci siano falsi positivi (stati non raggiunti durante la simulazione raggiunti poi durante il normale funzionamento del sistema) sia il più bassa possibile.

Le simulazioni hanno una durata di 24 ore e 12 ore, rispettivamente per il primo e il secondo modello. I risultati della cross validation, per differenti valori di k, vengono mostrati nella Tabella 5.4 per il primo modello e nella Tabella 5.5 per il

Split	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Accuratezza	0.989	0.989	0.990	0.990	0.990	0.990	0.990	0.990	0.990	0.990	0.991	0.990	0.990	0.990

Tabella 5.4. Risultati, relativi al primo modello, della k-fold cross validation per diversi valori di k, con dimensione dei chunk uguale a 25.

Split	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Accuratezza	0.987	0.988	0.988	0.987	0.987	0.988	0.988	0.988	0.988	0.988	0.988	0.988	0.988	0.988

Tabella 5.5. Risultati, relativi al secondo modello, della k-fold cross validation per diversi valori di k, con dimensione dei chunk uguale a 80.

secondo. Si può notare che il classificatore è robusto e mostra che non è importante quale range delle serie temporali vengono utilizzate per il training e il testing.

5.5 Valutazione tecnica

In questo capitolo verrà presa in considerazione una classe di attacchi ai sensori e la risposta dell'algoritmo di rilevamento.

L'attacco viene pensato come un'introduzione di rumore gaussiano nelle misurazioni dei sensori. Dato che nell'attuale implementazione viene già introdotto del rumore gaussiano, come discusso nel Capitolo 4. Un attacco può essere visto come una variazione della varianza del rumore già presente. Nel seguito verrà visto solo il caso con un incremento della varianza.

Per ogni sensore, viene aggiunto del rumore con una varianza che varia tra il 5% e il 20% attaccando diverse misurazioni di determinati chunk. I chunk sono scelti casualmente. Come sì può vedere dalle Tabelle 5.6 e 5.7, attaccando $\frac{1}{4}$ di chunk con una varianza incrementata del 5%, circa solamente un attacco su due viene riconosciuto, performance che cresce aumentando la porzione attaccata. Attaccando la totalità del chunk il rilevamento aumenta sensibilmente.

% Varianza / % Chunk	25%		50%		75%		100%	
	I	II	I	II	I	II	I	II
5%	53.2	55.8	73.0	72.8	89.2	90.1	93.1	92.9
10%	66.5	64.1	79.2	80.8	91.5	91.7	94.8	95.4
15%	71.1	73.8	83.2	84.2	93.7	94.0	96.3	97.7
20%	77.4	76.3	88.2	90.0	94.6	95.1	98.6	99.2

Tabella 5.6. Percentuale con cui gli attacchi sui sensori del primo modello vengono riconosciuti. Vengono mostrati i dati per i due sensori, con varianza del rumore introdotto variabile e porzione del chunk attaccato crescente.

% Varianza / % Chunk	25%			50%			75%			100%		
	I	II	III	I	II	III	I	II	III	I	II	III
5%	50.1	45.8	49.5	60.4	58.3	60.6	75.2	73.8	74.9	85.1	93.9	84.6
10%	54.7	49.9	54.1	65.3	63.2	66.1	78.1	75.2	77.6	89.4	86.3	90.1
15%	60.2	57.1	60.9	69.9	64.5	68.4	88.3	85.3	87.4	94.3	92.1	94.5
20%	65.1	61.4	66.0	73.1	69.9	73.3	94.7	90.1	93.9	97.3	94.6	98.1

Tabella 5.7. Percentuale con cui gli attacchi sui sensori del secondo modello vengono riconosciuti. Vengono mostrati i dati per i tre sensori, con varianza del rumore introdotto variabile e porzione del chunk attaccato crescente.

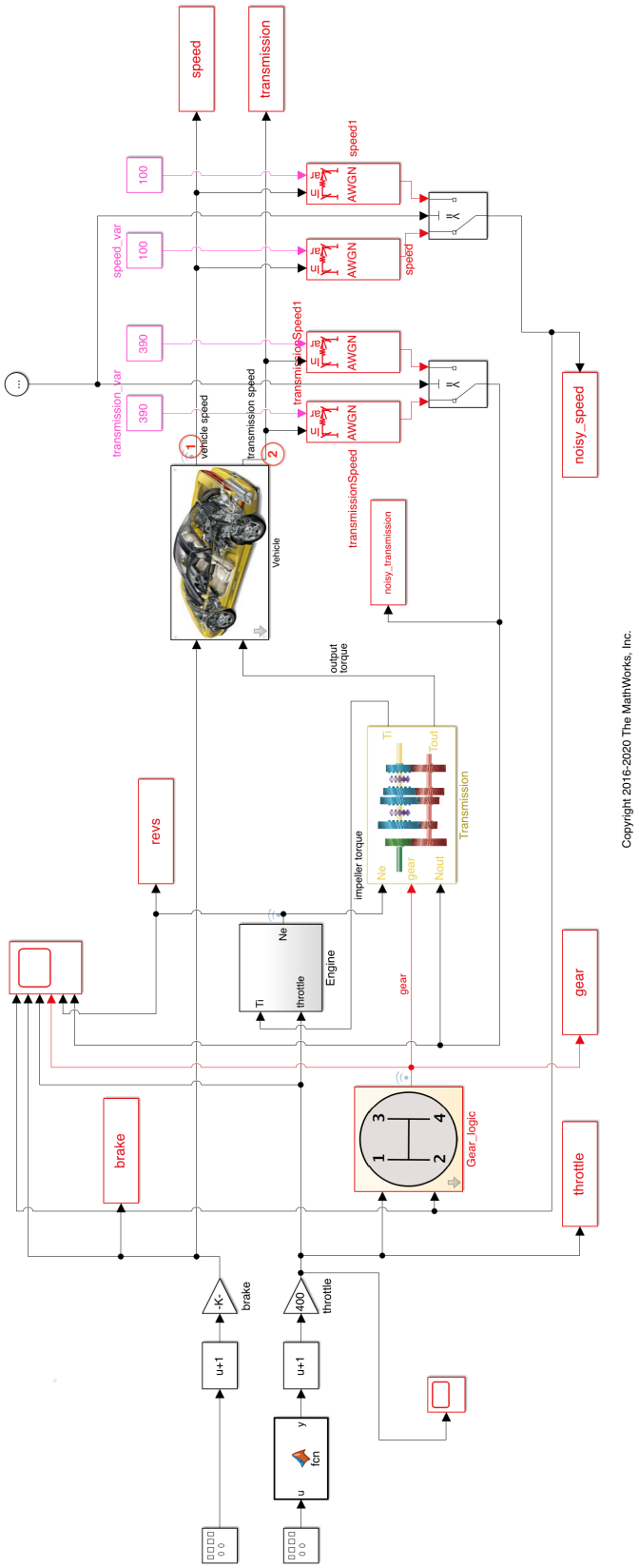

Figura 5.2. Modello Simulink di un'automobile a trasmissione automatica.

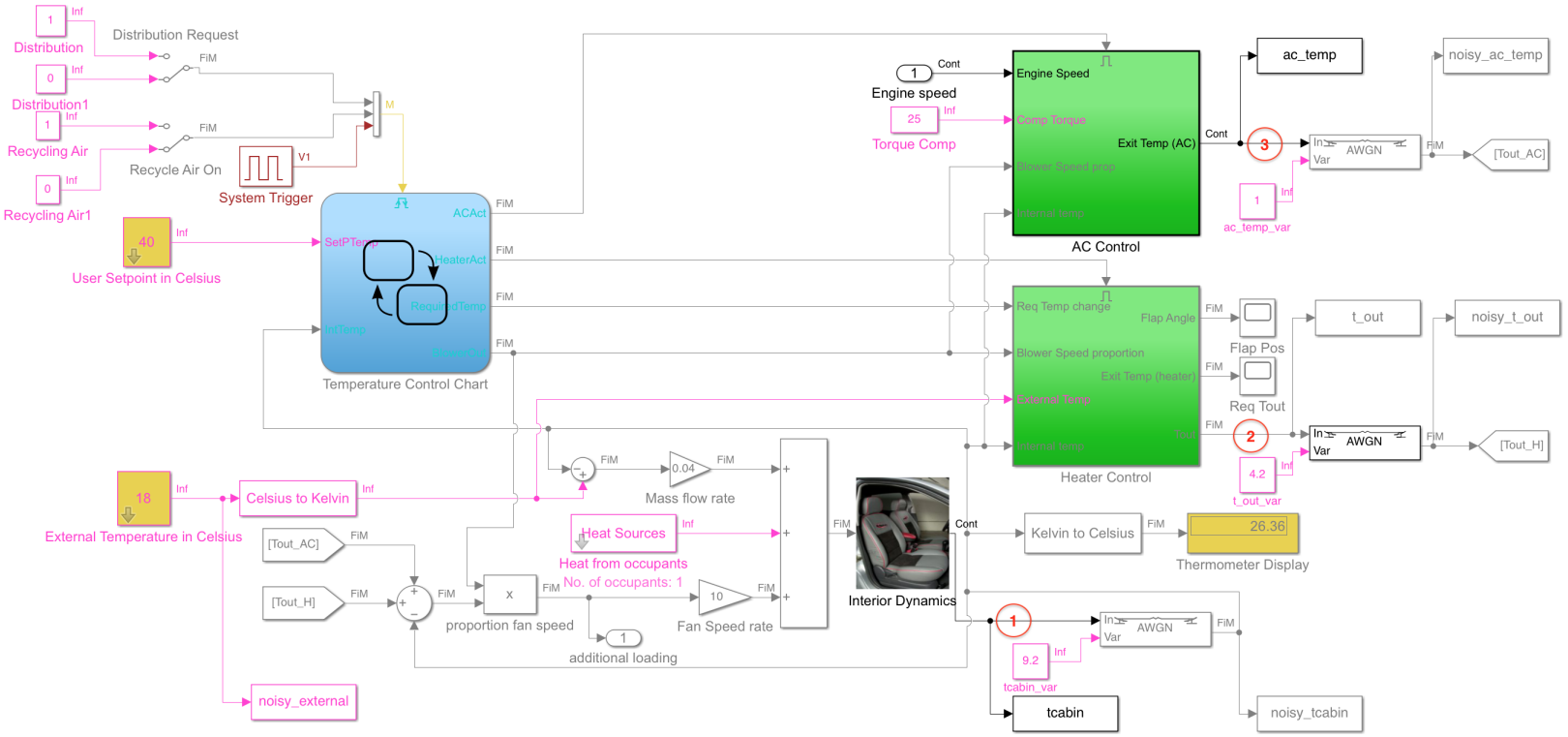

Figura 5.3. Modello Simulink di un sistema di controllo automatico del clima dell'ambiente interno di un'autovettura.

6. Conclusioni

Si è visto che è possibile identificare univocamente un sensore in base alle sue proprietà fisiche e manifatturiere e che nuovi dati possono essere confrontati con il pattern costruito dai valori che i sensori misurano durante il normale funzionamento.

Il metodo presentato si comporta bene nei confronti di attacchi messi in atto come introduzione di ruomore gaussiano, a partire da valori della varianza superiore a circa il 20% rispetto a quella già presente nel rurmore introdotto dagli stessi sensori. La tipologia di attacchi presentata ha un riscontro nel mondo reale e potrebbero essere messi in atto in attacchi reali, ma attacchi più sofisticati, come gli attacchi stealthy, potrebbero mettere in difficoltà lo schema proposto.

Futuri sviluppi potrebbero riguardare l'implementazione di nuove tipologie di attacchi e l'implementazione delle tecniche viste con casi di studio che descrivano le dinamiche dei sistemi in modo più realistico.

Bibliografia

[1] Ahmed, C. M., Zhou, J., and Mathur, A. P. Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in cps. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 566–581 (2018).

[2] Cardenas, A., Amin, S., Sinopoli, B., Giani, A., Perrig, A., Sastry, S., et al. Challenges for securing cyber physical systems. In *Workshop on future directions in cyber-physical systems security*, vol. 5. Citeseer (2009).

[3] Chang, C.-C. and Lin, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, **2** (2011), 1.

[4] Gollmann, D. and Krotofil, M. Cyber-physical systems security. In *The New Codebreakers*, pp. 195–204. Springer (2016).

[5] Noble, W. S. What is a support vector machine? *Nature biotechnology*, **24** (2006), 1565.

[6] Pedregosa, F., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, **12** (2011), 2825.

[7] Refaeilzadeh, P., Tang, L., and Liu, H. Cross-validation. *Encyclopedia of database systems*, **5** (2009), 532.