

CAN Network Anomaly Detection

정보보호 R&D 데이터 챌린지

CANDIS

이휘원, 김영준, 김보겸

Agenda

- Our Research
- CAN Network Attack Framework
- Attacks on CAN
- Anomaly Detection Methods
- Visualization
- Conclusion and Future Work

Our Research

- **CoWork!** by using github
- There are three researchers in our team
- The things that we developed is the following
 - CAN anomaly detection for DoS/Fuzzy/Replay Attack
 - Data sequence modeling based on RNN(LSTM) algorithm
 - Realtime visualization by using PyQt

Our Research

develacker / candis Private

Unwatch 3

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

CAN Network Anomaly Detection

Edit

Add topics

65 commits

1 branch

0 releases

3 contributors

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

develacker Merge branch 'master' of https://github.com/develacker/candis Latest commit 054c4b5 6 hours ago

final/output	result	6 hours ago
install	add install file	11 hours ago
models	update calculating avg intervals in replay detector	10 hours ago
result	add fuzzy result	9 days ago
.gitignore	remove .DS_Store	11 hours ago
README.md	reference update	23 hours ago
anomalies.csv	last commit before challenge	6 hours ago
attack.py	init	10 days ago
candis.py	Merge branch 'master' of https://github.com/develacker/candis	6 hours ago
candis.ui	Update candis.ui	21 hours ago
csv2hdf.py	last commit before challenge	6 hours ago
requirements.txt	update calculating avg intervals in replay detector	10 hours ago
rnn.py	update calculating avg intervals in replay detector	10 hours ago
timewindow.txt	add rnn modeling	12 hours ago
txt2csv.py	last commit before challenge	6 hours ago

Attack Framework

- Frequency effects
 - Insertions: extra packets
 - Erasures: missing packets
- Data: Altering packet data contents
 - Data replay
 - Data field modifications

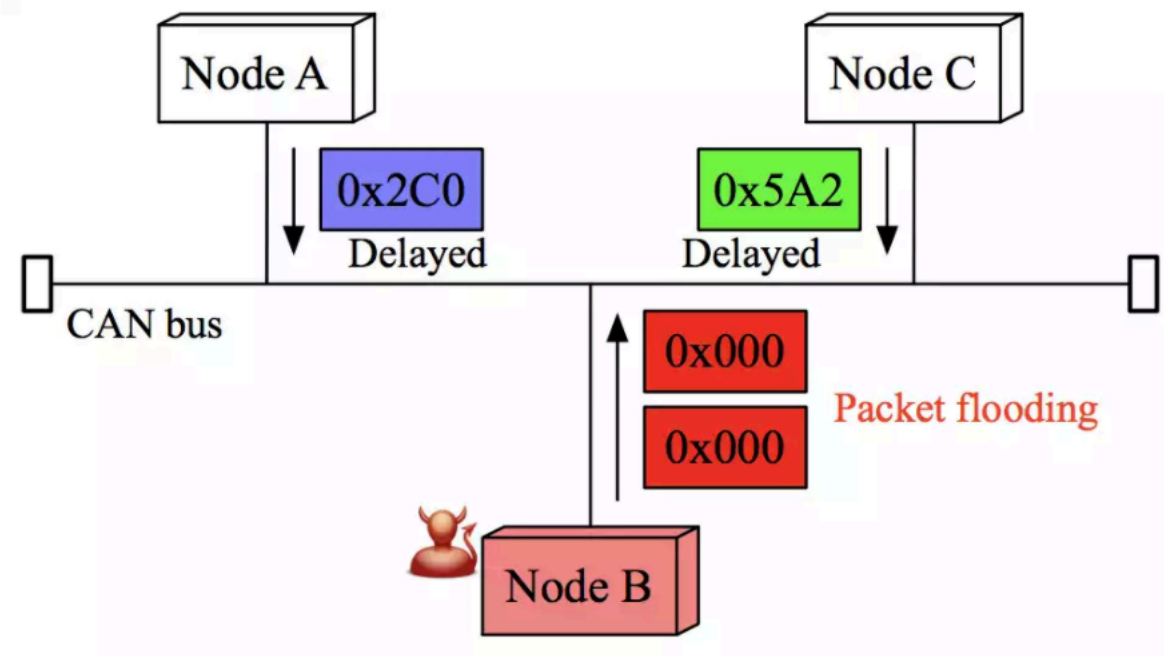
Frequency effects

- Each ID has a **fixed frequency of occurrence** on the bus
- The frequencies of normal packets are very **consistent**
- **Anomalies** in terms of frequencies will involve additional packets, or missing packets that were expected
- The majority of attacks involve inserted packets with specific IDs and data
- Some attacks can manifest as the absence of packets that should arrive at regular intervals

Altering packet data contents

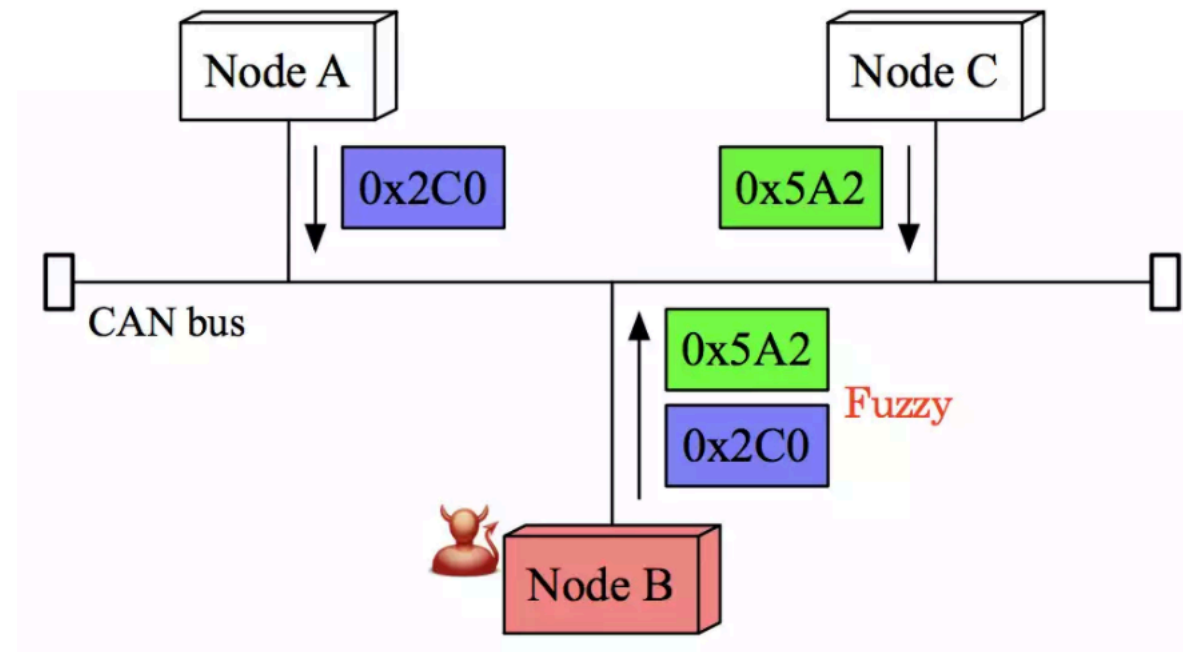
- Many bits in each **data sequence** are **constant**
- The second main signature of attacks is a **change** in the **data sequence** of some ID
- The **only indication of *replay attack*** is that the **data sequence of the ID** being replayed **has changed** from one context to another
- The replaced data is a legitimate subsequence, but **incongruous with preceding data sequence**

Attacks on CAN



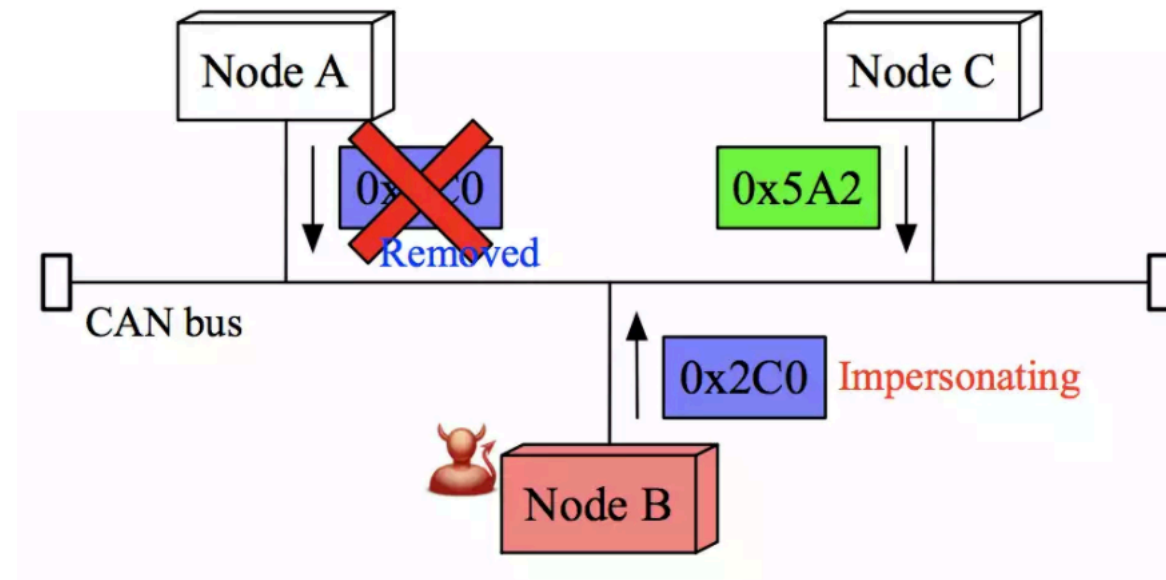
DoS Attack

Attacks on CAN



Fuzzy Attack

Attacks on CAN

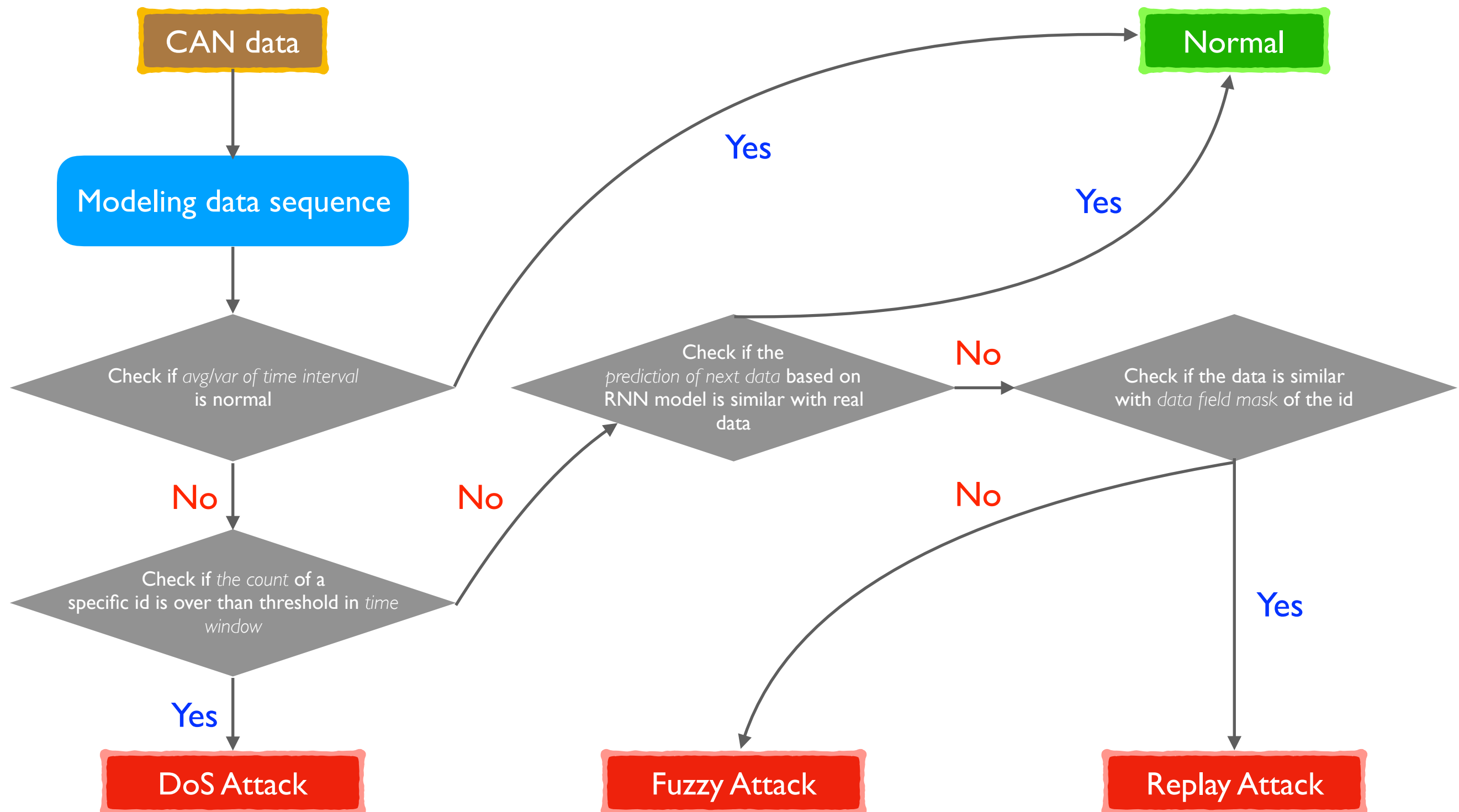


Impersonation Attack

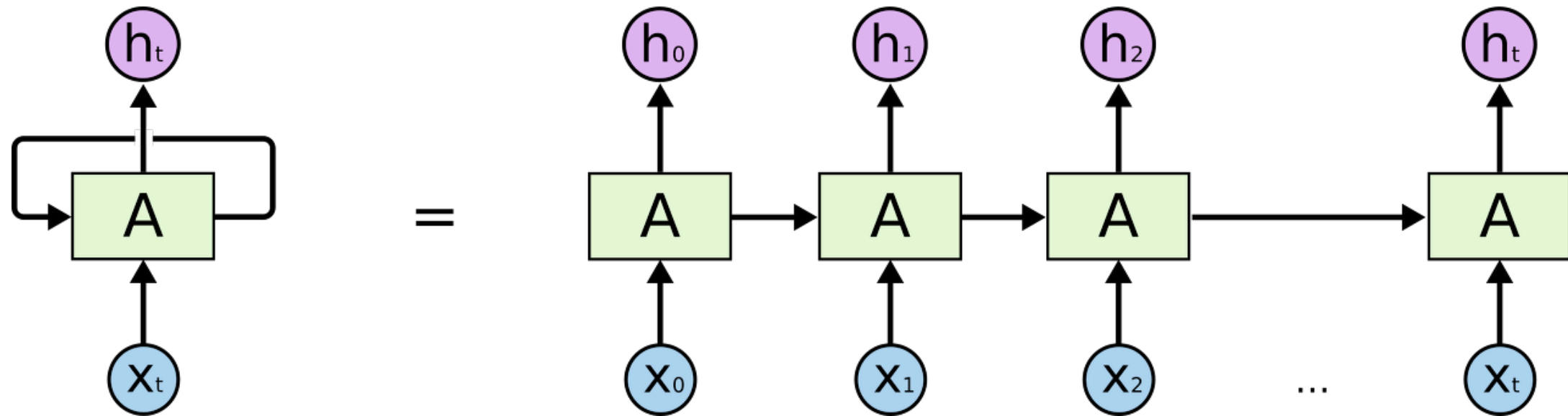
Anomaly Detection Methods

- **Frequency anomaly detection**
 - Average/Deviation of *Time Interval*
- **Data sequence anomaly detection**
 - RNN(Recurrent Neural Network) - *LSTM(Long Short Term Memory)*
 - *Data field mask* by ANDing all data sequence of each ID

Anomaly Detection Methods

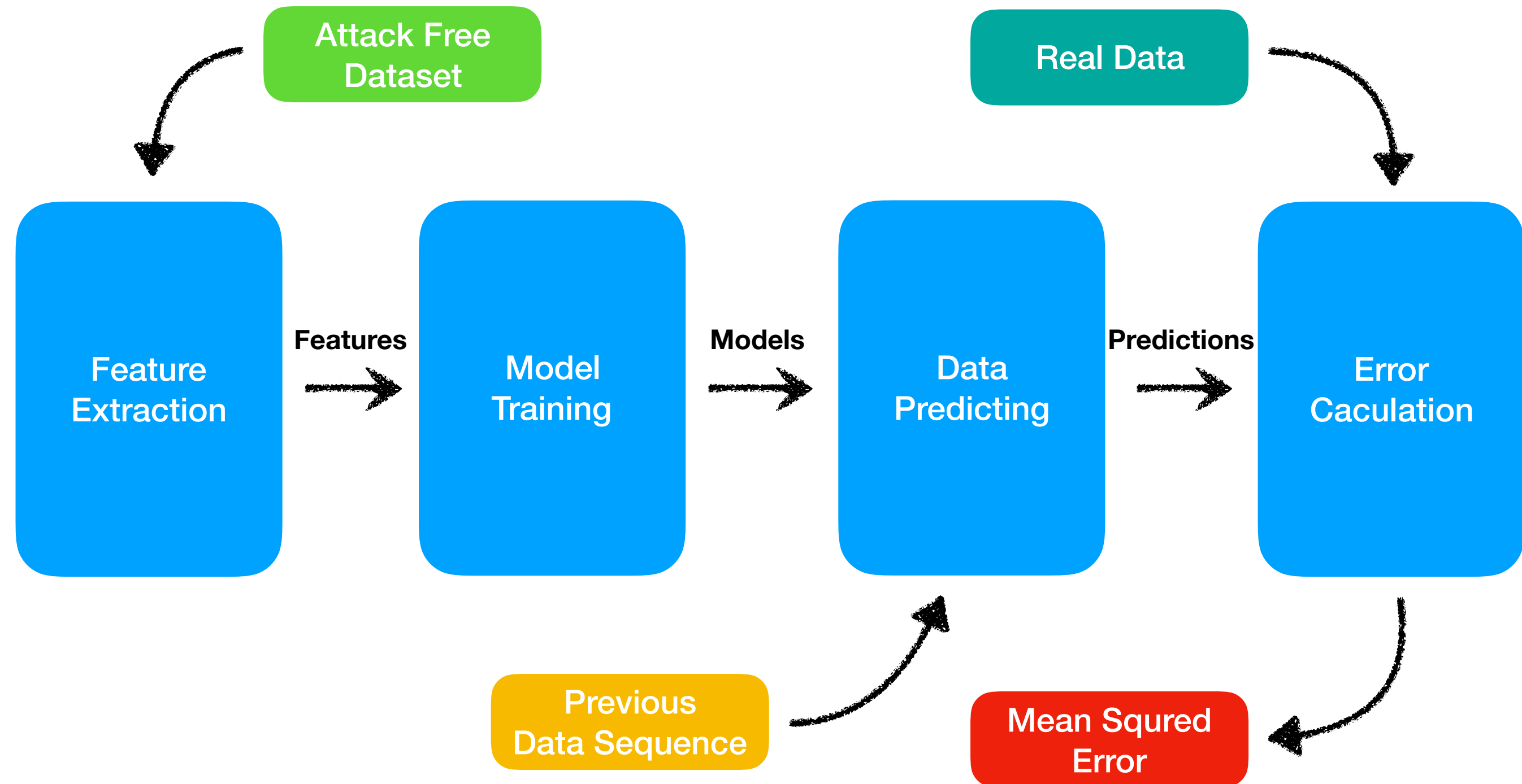


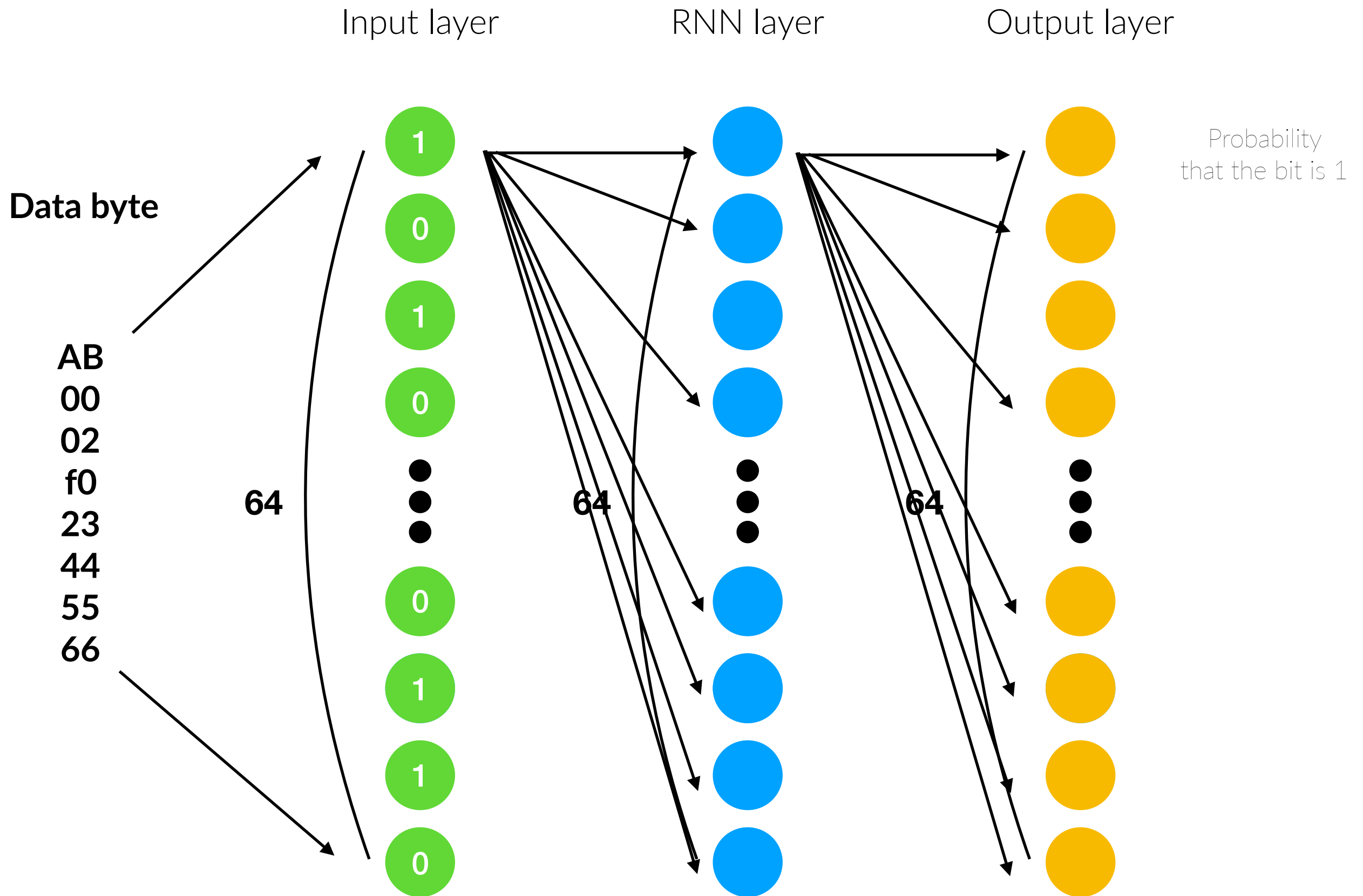
Recurrent Neural Network



- Save previous input data in network
- Predict next sequence from previous data
- In our case, predict next data bytes in real can packets

Recurrent Neural Network





Fully Connected Network

Recurrent Neural Network

RNN modeling by using Keras

```
model = Sequential()
model.add(SimpleRNN(64, input_shape = (steps, 64)))
model.add(Dense(64))

model.compile(loss='mse', optimizer='rmsprop', metrics=["accuracy"])

x_train = feature[:, :-1, :]
y_train = feature[:, -1, :]

history = model.fit(x_train, y_train, epochs=epochs, verbose=1)

model.save("./models/rnn_model_" + key + ".h5")
```

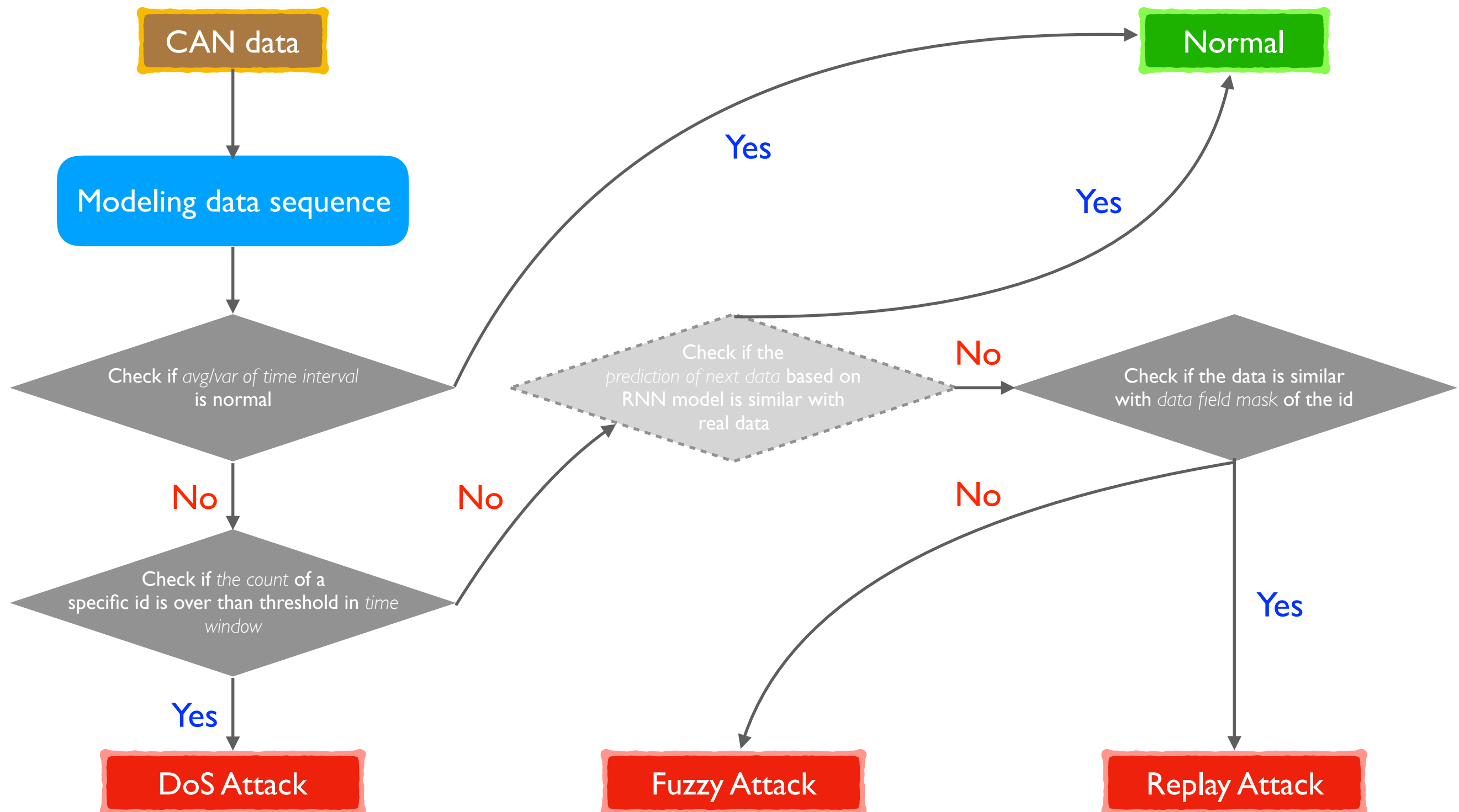

Weakness

- Too slow for real time detecting
 - Use only in warning state
- Vehicle model is changed
 - Train new model with new datasets... but not enough time :(

Actually...

- **There are some tricky issues to detect anomalies like**
 - Too many resources are needed to adjust deep learning to CAN network packet
 - There are some ambiguous concepts on CAN attack types
 - Very hard to speed up the rate of next data predictions based on RNN (LSTM) and very hard to understand the algorithm :(
 - Maybe there is no perfect algorithm which detects all the attack vectors at a time

Anomaly Detection Methods



Visualization

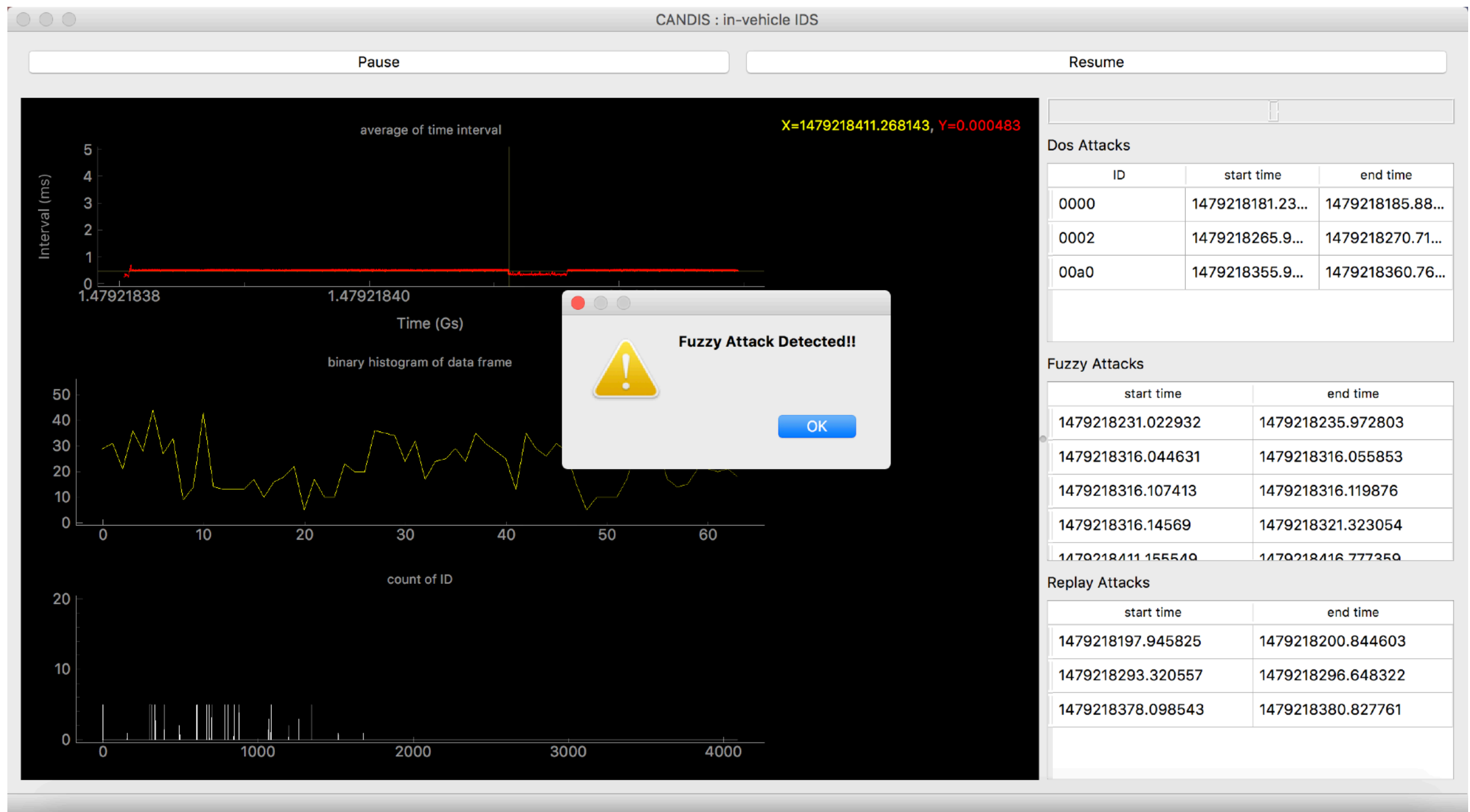
- Environment
 - Python 3
 - Numpy
 - PyQt5, PyQtGraph
- Focusing on real time plotting
- Cross checking whether the detection is correct

Visualization



Before

Visualization



After

Demo

Conclusion and Future Work

- There are many types of attack vector on CAN network
- Realtime and precision are the most important elements in anomaly detection
- More faster data sequence predictions
- User friendly visualization tool
- Improve the speed of detection
- Make it more easy to adopt in new vehicles

Reference

- Adrian Taylor, “Anomaly-based detection of malicious activity in in-vehicle networks”, 2017
- Mohammad Raashid Ansari, “Low-Cost Approaches to Detect Masquerade and Replay Attacks on Automotive Controller Area Network”, 2016
- Hyunsung Lee, Seong Hoon Jeong, Huy Kang Kim, “OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame”, 2016

Thank you!