Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Национальный исследовательский университет «МЭИ»

КУРСОВОЙ ПРОЕКТ

по дисциплине « Применение методов ИИ в электроэнергетике»

на тему «Детектирование подмены SV потоков IEC61850 9-2 LE с помощью рекуррентных сетей»

Группа Э-13м-18

Студент Ахмеров Т.А.
(Фамилия И.О.)

Оценка _____
(прописью)

Руководитель Темкина Р.В. _____
(Фамилия И.О.) *(подпись)*

Консультант Рыжков А.К. _____
(Фамилия И.О.) *(подпись)*

Рецензент _____ _____
(Фамилия И.О.) *(подпись)*

Москва, 2024 г.

ОГЛАВЛЕНИЕ

1. Введение

1.1.Цели и задачи курсовой работы

В современном мире энергетические системы становятся все более сложными и требующими постоянного мониторинга и контроля. Для обеспечения надежной и эффективной работы энергосистем необходимо применение современных информационно-телекоммуникационных технологий и методов управления, с целью повышения надежности и управляемости и автоматизации объектов электроэнергетики. Одной из ключевых задач в управлении энергосистемами является обеспечение безопасности и надежности передачи данных между устройствами.

Стандарт IEC61850 9-2 LE предусматривает использование SV (Sampled Values) для передачи мгновенных значений токов и напряжений между устройствами в энергосистемах. Однако, существует риск подмены SV потоков, что в свою очередь может привести к нарушению работы энергосистемы и серьезным последствиям.

В связи с этим, актуальной задачей является разработка эффективных методов детектирования подмены SV потоков. Одним из перспективных подходов к решению этой задачи является применение методов искусственного интеллекта, в частности, рекуррентных нейронных сетей. Таким образом в рамках курсового проекта можно выделить следующие задачи:

- Построить модель электрической сети с параметрами, приближенными к реальной сети, для снятия мгновенных значений токов и напряжений
- Разработать алгоритм генерации SV потока по полученным значениям аналоговых сигналов и последующий алгоритм захвата SV трафика;

– Подготовить данные для обучения модели, с учетом специфики детектирования аномалий SV потока.

– Обучить рекуррентную нейронную сеть (РНС) на подготовленных данных, с целью детектирования подмены SV трафика.

– Протестировать обученную сеть на новых данных.

2. Теоретическая часть

2.1. Обзор стандарта МЭК 61850 9-2 LE

Цифровая Подстанция (ЦПС)– это подстанция, все системы которой имеют очень высокий уровень автоматизации. Все вторичное и первичное оборудование такой подстанции ориентировано на цифровую передачу данных. Обмен данными выстраивается в соответствии с протоколами передачи, описанными в стандарте МЭК 61850. Типовая архитектура ЦПС представлена на рисунке 1.

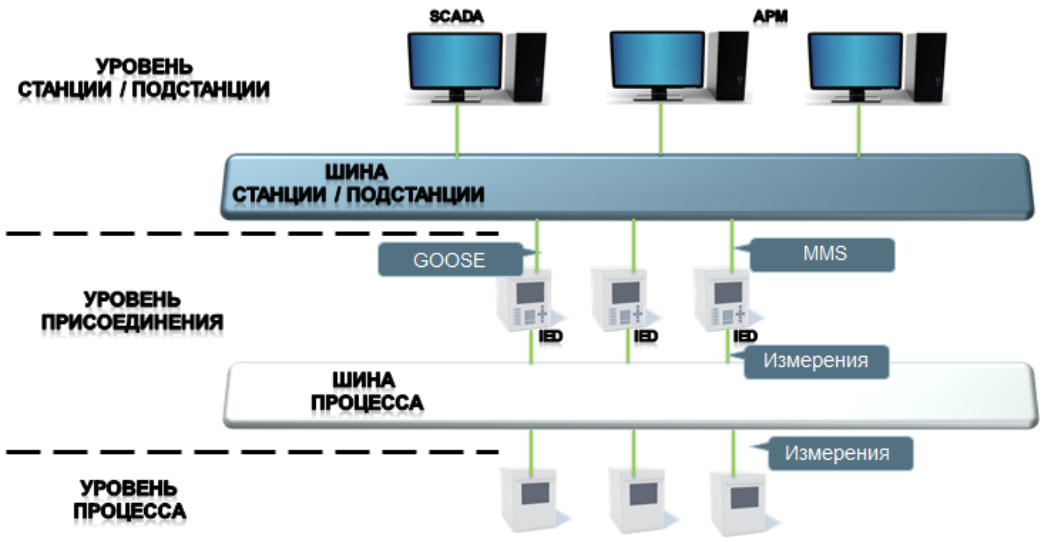


Рисунок 2.1 Типовая архитектура ЦПС

Уровень станции/подстанции включает в себя АРМы и сервера.

Уровень присоединения включает в себя все технологическое оборудование.

Уровень процесса включает в себя измерительное оборудование.

Также есть две шины для объединения уровней:

- Шина станции/подстанции, объединяющая в себе функции шины «Мониторинг/Управление» и шины «Передача сигналов РЗА».
- Шина процесса, выполняющая функции передачи мгновенных значений напряжения и тока.

Протокол МЭК 61850-9-2 (Sampled Values – SV) предназначен для передачи выборок данных в технологическую локальную сеть (шина процесса),

Использование протокола неразрывно связано с термином «шина процесса». Шиной процесса по МЭК 61850-1 называется коммуникационная шина данных, к которой подключены устройства полевого уровня подстанции (коммутационные аппараты, измерительные трансформаторы). В общем случае к шине процесса могут быть подключены не только измерительные преобразователи, но также выключатели, разъединители и другое оборудование. Однако именно передача мгновенных значений от измерительных трансформаторов производит наибольшую нагрузку на информационную сеть «шины процесса».

В традиционной схеме подключения устройств РЗА цепи от измерительных трансформаторов тока и напряжения, находящихся на открытые распределительные устройства (ОРУ), прокладываются до терминалов РЗА, размещенных в общеподстанционном пункте управления (ОПУ). Использование концепции шины процесса предполагает, что все сигналы, включая мгновенные значения токов и напряжений, оцифровываются непосредственно в аппарате и передаются устройствам защиты и автоматики в виде цифрового потока данных по информационной сети.

2.2.Структура SV пакета

Концепция SV потоков заключается в том, что издатель периодически отправляет сообщения с точно определенными временными интервалами. Временной интервал зависит от двух факторов: измеренной частоты сигнала и числа выборок за период (SPP). МЭК 61850-9-2 определяет два значения числа выборок за период: 80 и 256. Так, например, если измеренная частота сигнала составляет 50 Гц, а SPP равно 80, временной интервал отправки будет 1/50/80, или 250 мкс. Все сообщения публикуются под topic'ом. Подписчик получает все сообщения, но фильтрует и анализирует только сообщения, под конкретным topic'ом, на который был подписан. Согласно протоколу, структура SV сообщения имеет вид:

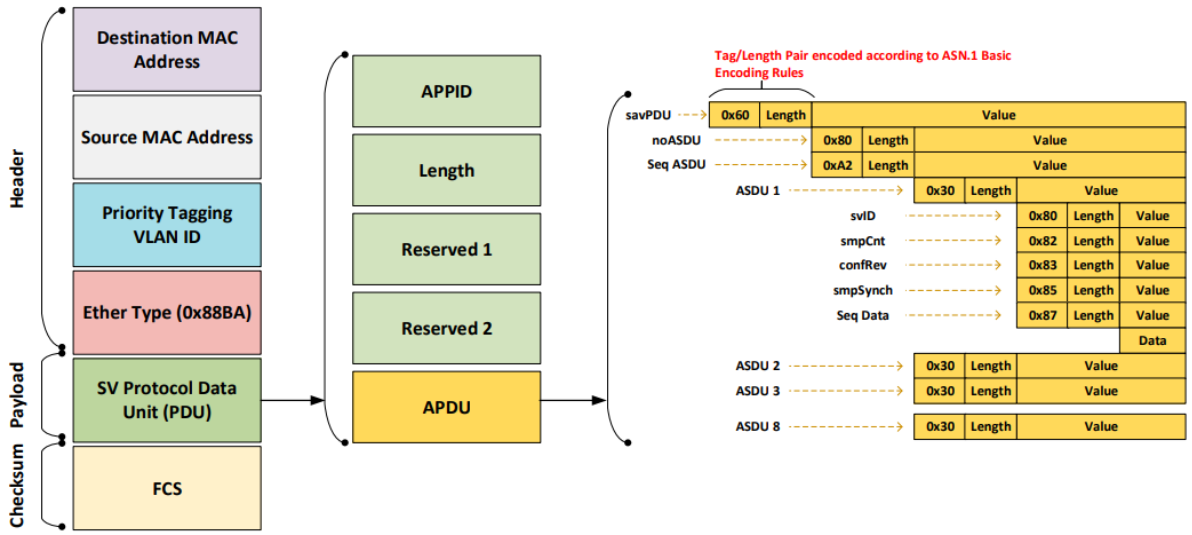


Рисунок 2.2 Структура SV сообщения

Preamble – содержит последовательность 8 битов и указывает на начало Ethernet фрейма.

Destination address – MAC адрес назначения. Рассылка может быть одноадресной и многоадресной. Стандарт МЭК 61850 регламентирует набор адресов формата 01:0C:CD:04:xx:xx, с возможным диапазоном 00:00 - 01:FF. Длина 6 байт.

Source address –MAC адрес отправителя. Длина 6 байт.

Ethertype – Идентификатор L3 протокола. 0x88BA – Sampled Value Transmission. Длина 2 байта.

APPID – идентификатор приложения, используется для фильтрации SV кадров. Длина 2 байта. Принимаемые значения – от 0x4000 - 0x7FFF;

Length – суммарная длина полей APPID, Length, reserved 1, reserved 2 и APDU. Длина 2 байта.

Reserved 1 – зарезервированное поле, в котором указывается режим работы устройства. Длина 2 байта. (По умолчанию значение 0x0000)

Reserved 2 – зарезервированное поле для передачи параметров безопасности. Длина поля 2 байта. (По умолчанию значение 0x0000)

APDU (Application Protocol Data Unit) – блок данных прикладного уровня, содержит измерения.

Frame Check Sequence – значение контрольной суммы для проверки целостности данных при передаче. Длина 4 байта.

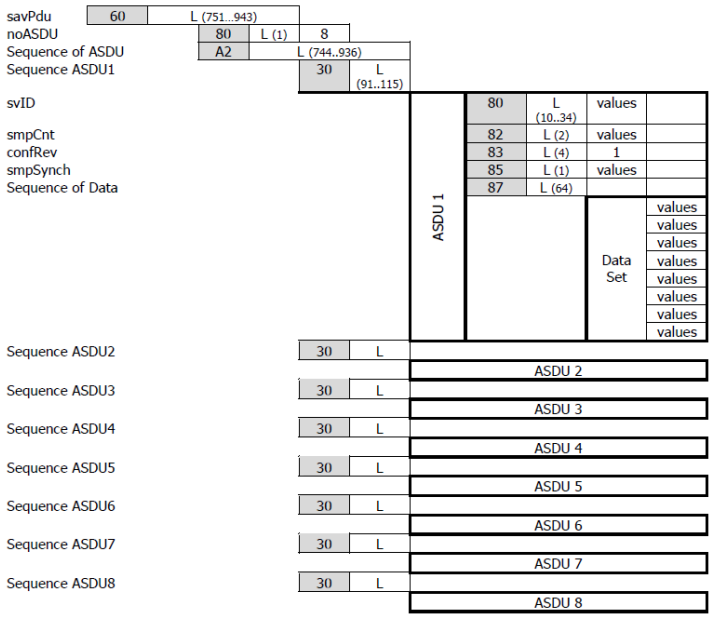


Рисунок 2.3 - Структура APDU

savPdu – начало сообщения. Длина поля 4 байта.

noASDU – количество блоков данных (ASDU) в одном кадре.Длина поля 3 байта, первый байт 0x10 – метка поля, второй байт 0x01 – длина значения, третий байт 0x01 – значение.

seqASDU – начало блоков данных. Длина поля 4 байта, первый байт 0xA2 – метка поля, второй байт 0x82, с третьего байта общая длина блоков данных.

seqASDU 1 – идентификатор начала блока данных. Длина поля 2 байта, первый байт 0x30 – метка поля, второй байт – длина блока данных.

svID – идентификатор мгновенных значений (SV потока). Длина поля от 21 до 69 байт, первый байт 0x80 – метка поля, второй байт – длина значения идентификатора мгновенных значений (от 10 до 34 байт), с третьего байта значение идентификатора мгновенных значений длиной от 10 до 34 байт соответственно.

smpCnt – номер выборки. Длина поля 4 байта, первый байт 0x82 – метка поля, второй байт – 0x02 длина значения, с третьего байта значение номера выборки.

confRev – номер конфигурации. Длина поля 6 байт, первый байт 0x83 – метка поля, второй байт – 0x04 длина значения, с третьего байта значение номера конфигурации.

smpSynch – метка наличия синхронизации. Длина поля 3 байта, первый 0x85 – метка поля, второй байт – 0x01 длина значения, третий байт – значение.

Sequence of Data – последовательность мгновенных значений. Длина поля 66 байт, первый байт 0x87 – метка поля, второй байт – 0x40 длина последовательности мгновенных значений, с третьего байта последовательность мгновенных значений.

Поле Sequence of Data содержит информацию о мгновенных значениях токов и напряжений фаз A, B, C и нейтрали N. Каждое измеренное значение кодируется 8-ми байтным кодом.

Мгновенные значения тока:

- InnATCTR1.Amp.instMag.i
- InnBTCTR2.Amp.instMag.i
- InnCTCTR3.Amp.instMag.i
- InnNmTCTR4.Amp.instMag.i

Мгновенные значения напряжения:

- UnnATVTR1.Vol.instMag.i
- UnnBTVTR2.Vol.instMag.i
- UnnCTCVR3.Vol.instMag.i
- UnnNmTCVR4.Vol.instMag.i

Quality – статусная информация о токах и напряжениях:

- InnATCTR1.Amp.q
- InnBTCTR2.Amp.q
- InnCTCTR3.Amp.q
- InnNmTCTR4.Amp.q
- UnnATVTR1.Vol.q
- UnnBTVTR2.Vol.q
- UnnCTCVR3.Vol.q
- UnnNmTCVR4.Vol.q

Статусная информация включает в себя Validity и DetailQuality.

– Validity – оценка качества передаваемых данных:

– Good – неисправности не обнаружены;

– Invalid – получаемая информация неверна и ее нельзя использовать;

– Questionable – проблемы с сервером, однако передаваемая информация может оставаться актуальной.

Поле DetailQuality – дополнительный идентификатор качества. Длина поля 1 байт. Устанавливаемые флаги идентификатора DetailQuality приведены в 2.

Таблица 1 - Идентификатор DetailQuality

Бит	**Флаг**
1	overflow (переполнение)
2	outOfRange (за пределы диапазона)
3	badReference (требуется калибровка)
4	oscillatory (колебательный)
5	failure (повреждение)
6	oldData (старые данные)
7	inconsistent (несогласованный)
8	inaccurate (неточный)

2.3. Описание проблемы подмены SV потоков

Как подтверждают исследования, протокол SV не является защищенным. Сообщения SV напрямую отображаются на уровне 2 модели OSI и вполне могут поставить под угрозу устройства шины процесса ЦПС. Возможная атака на сеть ЦПС может исходить из нескольких источников:

– персонал, имеющий доступ к сети МЭК 61850, может загрузить вредоносное ПО либо намеренно (недовольный сотрудник), либо непреднамеренно (неправильное использование зараженных устройств);

– Нарушение цепочки поставок (Supply Chain). В этом случае вредоносное ПО загружается на этапе производства устройства;

– Удаленный доступ. Злоумышленник может получить доступ к локальной сети через корпоративные сети или персональные устройства сотрудников. В этом случае нарушители часто используют брутфорс, спам рассылки на корпоративные аккаунты и др.

В качестве примера успешных атак на объекты электроэнергетики можно привести атаку направленную на промышленные логические контроллеры (ПЛК) атомной станции в Иране с помощью компьютерного червя Stuxnet, и атака на подстанцию «Северная» в Киеве с помощью ПО CrashOverRide (другое возможное название Industroyer).

Получив представление о возможных атаках на объекты электроэнергетики, рассмотрим конкретные кибератаки, направленные на изменение/подмену SV трафика.

1) Атака посредника (Man-in-the-Middle или MITM)

При атаке MITM злоумышленник перехватывает трафик между двумя устройствами с помощью так называемого ARP-спуфинга. ARP — это протокол связи, который используется для преобразования IP-адресов в MAC-адреса.

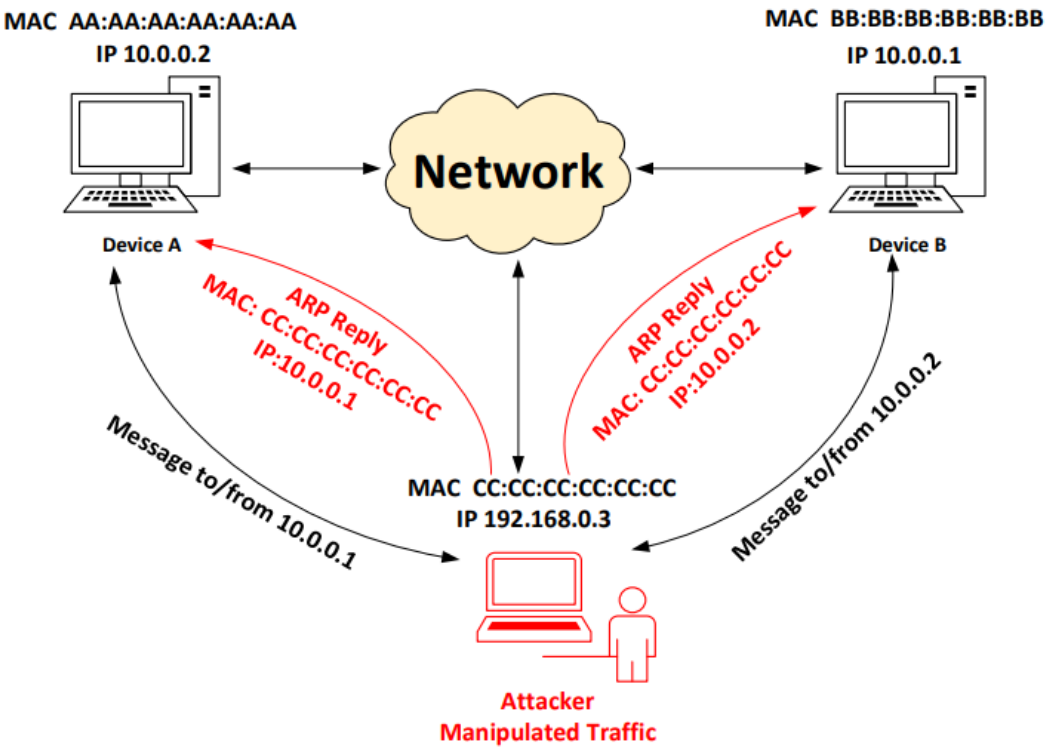


Рисунок 4 Атака Man-In-The-Middle

Как показано на рисунке 4, Злоумышленник отправляет ложный ARP Reply в котором объявляет себя искомым узлом, чтобы изменить ARP-таблицу (сопоставление IP-адресов и MAC-адресов в данной сети). Следовательно, будет неправильная связь между IP-адресами и MAC-адресами. Это позволит злоумышленнику прерывать сообщения, передаваемые между взаимодействующими устройствами, и манипулировать ими. Однако такая атака невозможна для протокола SV, поскольку сообщения SMV транслируется на уровне 2 OSI. Исключением является ситуация, когда инсайдер скомпрометировал сеть.

2) SV-spoofing

Пакет SV с измененными полями данных передается на шину процесса устройством злоумышленника, сохраняя при этом MAC-адрес настоящего издателя неизменным, как показано на рисунке 5.

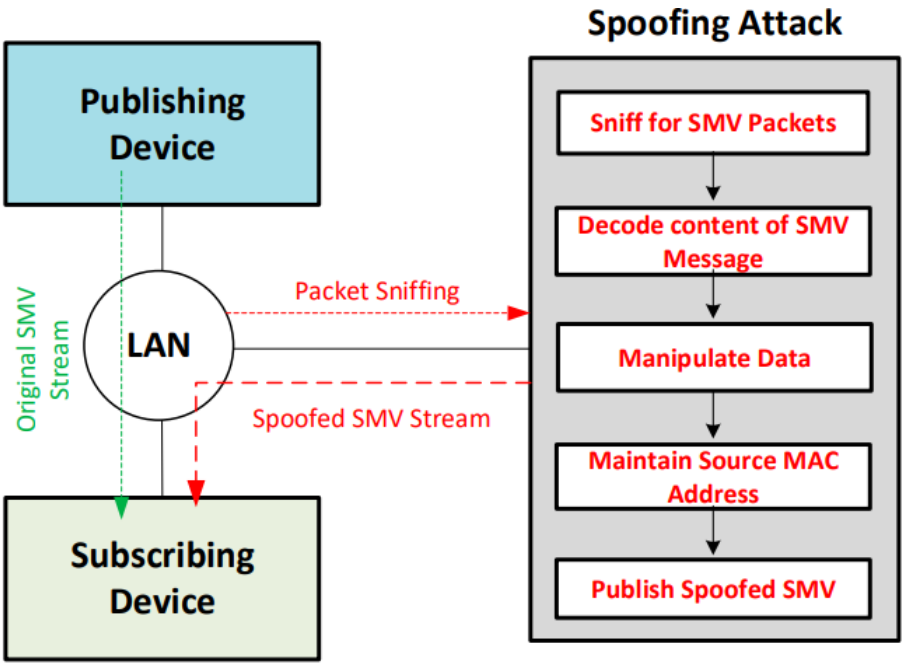


Рисунок 5 SV спуфинг

Это возможно, поскольку пакеты SV не шифруются. Злоумышленник может записать SV пакет, декодировать его содержимое, манипулировать определенными полями и передать поддельное сообщение на шину процесса. Стоит отметь, что в данном случае в результате атаки, исходное сообщение, отправленное издателем, дойдет до подписчика, независимо от действий злоумышленника.

Стандарт IEC 62351 рекомендует использовать асимметричный алгоритм шифрования с открытым ключом RSA для аутентификации пакетов SV и обеспечение их целостности. Однако поскольку аутентификация на основе RSA требует больших вычислительных затрат и не подойдет для SV, т.к. задержка между сообщениями недолжны превышать 250 мкс (при частоте электрической сети 50 Гц и SPP 80).

Манипуляции над захваченным пакетом приводят к тому, что пакеты в сети либо дублируются (повторяющийся smpCnt или выбивающееся из последовательности значение) либо отправляются с задержкой в этом случае необходим алгоритм, который определяет ложный SV пакет.

3. Архитектура решения и методы его реализации

В соответствие с заданием, предлагается следующая архитектура:

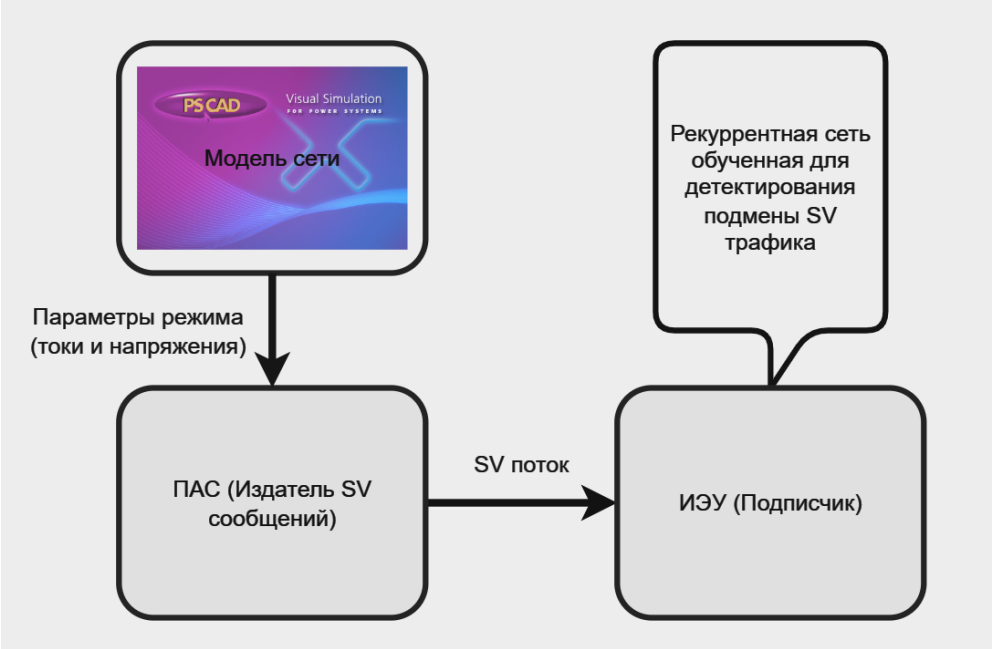


Рисунок 6 Архитектура проекта

Модель PSCAD генерирует значения токов и напряжений в формате COMTRADE файла и передает его издателю SV (ПАС), в котором токи и напряжения преобразуются согласно протоколу SV и отправляются в шину процесса. Подписчик SV (ИЭУ) получает SV поток, декодирует его и отправляет в рекуррентную сеть для детектирования подмены SV трафика.

3.1.Описание модели электрической сети в PSCAD

Полная модель ЭЭС представлена в приложении, ниже представлена схема сети.

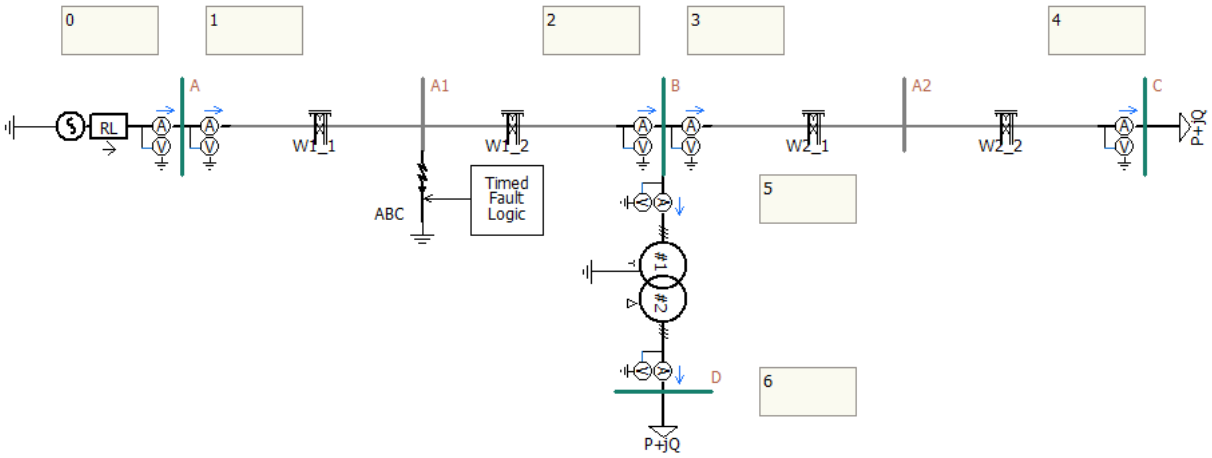


Рисунок 7 Схема модели ЭЭС

Модель сети 220 кВ позволяет снимать параметры сети как в нормальном, так и в аварийном режиме. Указанная выше конфигурация позволяет проводить опыт трехфазного КЗ на линии W1. В результате кривые тока и напряжения имеют вид:

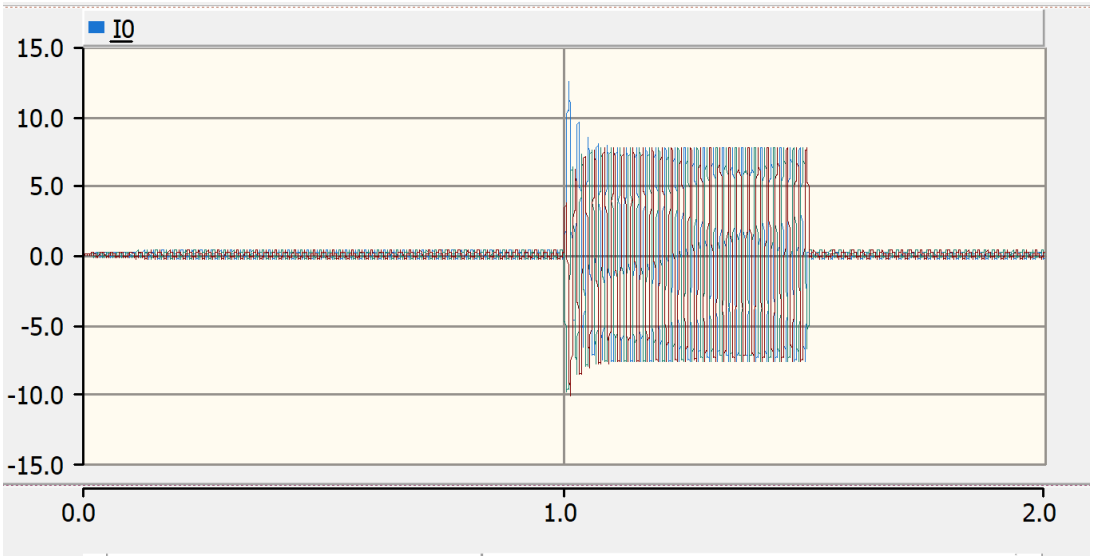


Рисунок 8 Мгновенные значения тока трех фаз

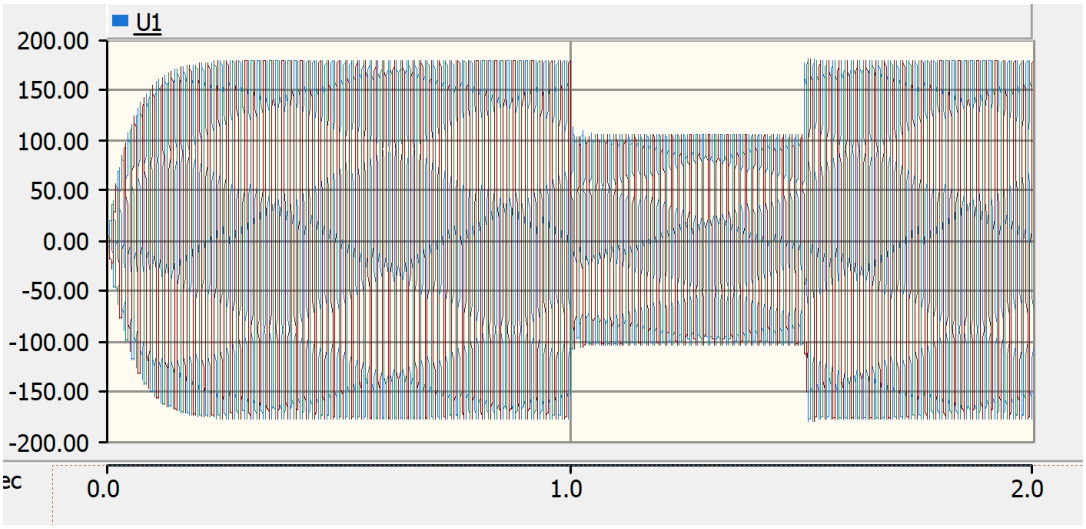


Рисунок 9 Мгновенные значения напряжения трех фаз

Как уже было сказано ранее результаты работы модели записываются в формате COMTRADE 99, представляющий из себя формат регистрации осциллограмм переходных процессов (аварий) в энергосистемах.

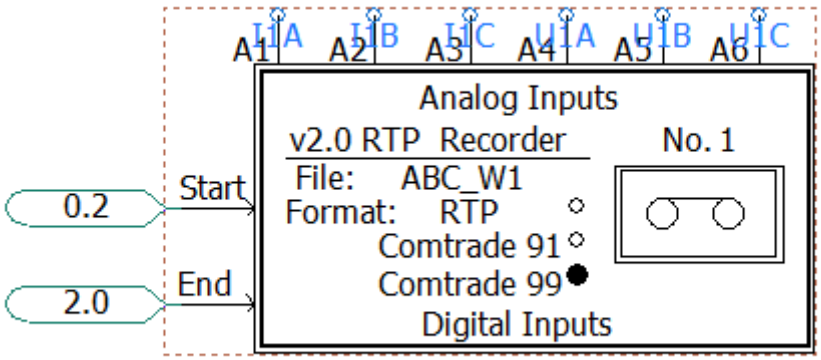


Рисунок 10 Модуль записи осциллограмм

На выходе получается два файла:

с расширением *.cfg* - содержит информацию о формате *.dat*, частоту, продолжительность, количество каналов и их тип.

с расширением *.dat* - записанные выборки в формате текста или бинарном формате.

В данном случае на рисунке 10 блок осуществляет запись осциллограмм тока и напряжения в начале линии W1 с момента 0,2с

3.2.Описание рекуррентной нейронной сети для решения поставленной задачи

Рекуррентные нейронные сети (RNN) способны обрабатывать последовательности данных и учитывать зависимости между соседними элементами последовательности. Кроме того, RNN могут учитывать информацию о предыдущих пакетах в потоке, при обработке текущего пакета. Это необходимо, поскольку подмена SV потоков может происходить не только в одном пакете, но и в последовательности пакетов.

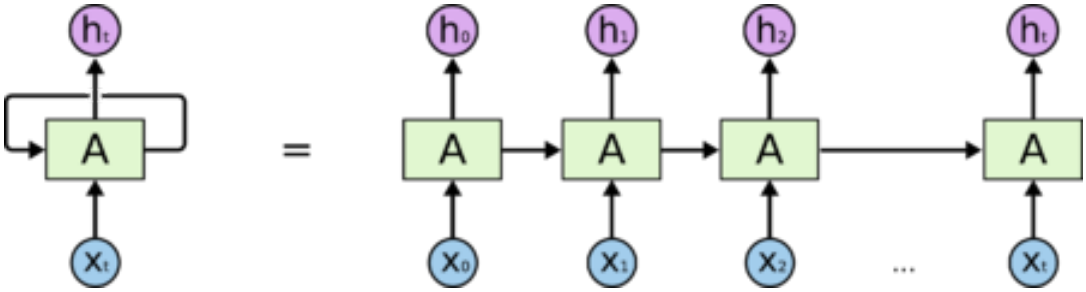


Рисунок 11 RNN и её развертка

Поскольку одни и те же параметры используются на всех временных этапах в сети, градиент на каждом выходе зависит не только от расчетов текущего шага, но и от предыдущих временных шагов. Например, чтобы вычислить градиент для четвертого элемента последовательности, необходимо «распространить ошибку» на 3 шага и суммировать градиенты. Этот алгоритм называется «алгоритмом обратного распространения ошибки сквозь время» (Backpropagation Through Time).

RNN добавляют память к искусственным нейронным сетям, но реализуемая память получается короткой — на каждом шаге обучения информация в памяти смешивается с новой и через несколько итераций полностью перезаписывается. LSTM-модули в свою очередь разработаны специально, чтобы избежать проблемы долговременной зависимости, запоминая значения как на короткие, так и на длинные промежутки времени. Это объясняется тем, что LSTM-модуль не использует функцию активации внутри своих рекуррентных компонентов. Таким образом, хранимое значение

не размывается во времени и градиент не исчезает при использовании метода обратного распространения ошибки во времени при тренировке сети.

В LSTM повторяющийся блок имеет более сложную структуру, состоящую не из одного, а из четырех слоев.

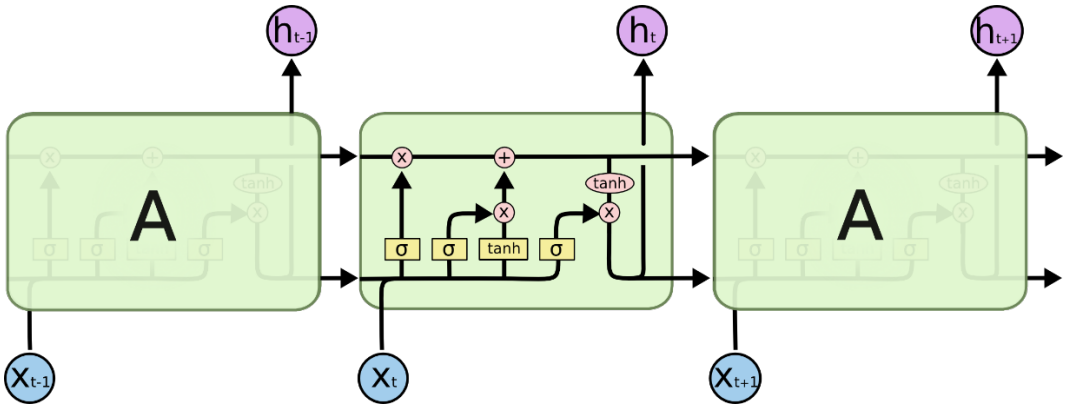


Рисунок 12 Повторяющийся блок LSTM

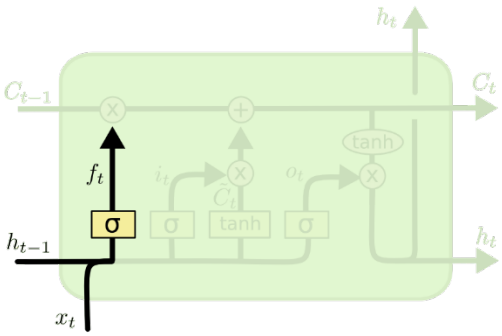
Кроме скрытого состояния h_n, в LSTM появляется понятие состояния блока (**cell state**, c_n). Cell state c_n будет играть роль внутренней, закрытой информации LSTM-блока, тогда как скрытое состояние h_n теперь становится передаваемым наружу (не только в следующий блок, но и на следующий слой или выход всей сети) значением. LSTM может добавлять или удалять определенную информацию из cell state с помощью специальных механизмов, которые называются **gates** (ворота или вентили в русскоязычной литературе). Рассмотрим этот механизм подробнее.

Основное назначение вентиля – контролировать количество проходящей через него информации. Для этого матрица, проходящая по каналу, который контролирует вентиль, поточечно умножается на выражение вида

$$\sigma(W_1 h_{n-1} + W_2 x_n)$$

Сигмоида выдает значение от 0 до 1, означающее, какая доля информации сможет пройти через вентиль. Рассмотрим типы вентилей в том порядке, в каком они применяются в LSTM.

Forget gate (вентиль забывания) позволяет на основе предыдущего скрытого состояния h_{t-1} и нового входа x_t определить, какую долю информации из c_{n-1} (состояния предыдущего блока) стоит пропустить дальше, а какую забыть. Доля сохраняемой информации из предыдущего состояния вычисляется по формуле ниже.



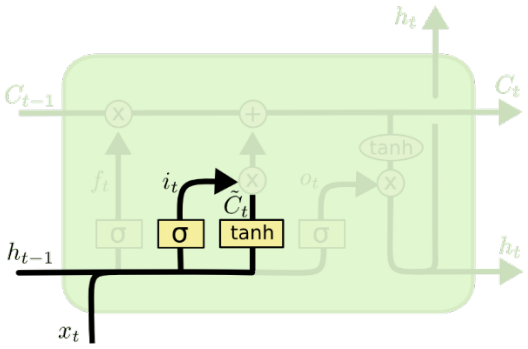
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Дальше f_t поэлементно умножается на c_{t-1}.

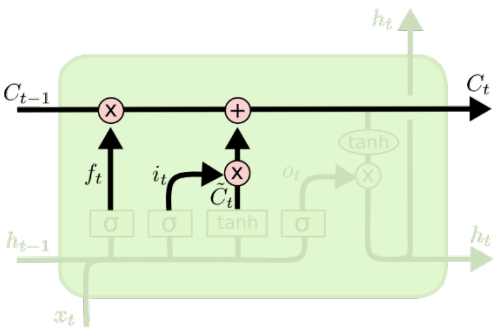
Следующий шаг – определить, что нового мы внесём в cell state.

$$\widetilde{C_t} = \tanh\left(h_{t-1}{W_1}^C + x_t{W_2}^C + \; b_c\right)$$

Input gate (вентиль входного состояния).



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

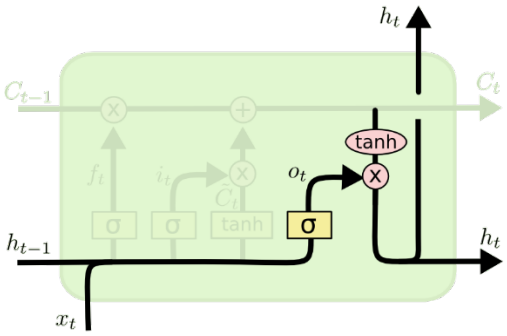


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

где * – это поэлементное умножение. Первое слагаемое отвечает за то, что «забывание» не релевантной информации из c_{t-1}, а второе – за привнесение новой, релевантной.

Роль выходного вектора LSTM-блока будет играть h_n. Он вычисляется по cell state с помощью последнего вентиля.

Output gate (вентиль выходного состояния). Он отвечает на вопрос о том, сколько информации из cell state следует отдавать на выход из LSTM-блока. Доля вычисляется следующим образом:



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Теперь пропускаем cell state через гиперболический тангенс, чтобы значения были в диапазоне от −1 до 1, и умножаем полученный вектор на o_n, чтобы отфильтровать информацию из cell state, которую нужно подать на выход.

Далее рассмотрим практическую реализацию

4. Практическая часть

4.1.Работа издателя и подписчика SV

Как уже было сказано ранее, параметры режима передаются в виде COMTRADE файла и сохраняются на жестком диске ПК. Сеть работает в двух режимах: нормальном и аварийном (трехфазное КЗ на линии W1).

Простейшая схема сети выглядит следующим один коммутатор, один издатель SV и один подписчик SV. Оба подключены к коммутатору. В данной работе виртуальная сеть построена с помощью виртуального сетевого адаптера гипервизора (VMware, VirtualBox) и с помощью библиотеки pcap4j и языка Java были созданы два класса SvSubscriber и SvPublisher, первый отправляет в виртуальную сеть SV пакеты, второй читает их и записывает в файл формата .csv для дальнейшей работы с нейронной сетью.

Рассмотрим более подробно программную реализацию:

С помощью класса WatchService происходит наблюдение за директорией, в которую происходит запись Comtrade файлов модели:

```
for (WatchEvent<?> event : key.pollEvents()) {
    WatchEvent.Kind<?> kind = event.kind();

    Thread.sleep( 150 );
    // Если произошло изменение файла
    if (kind == StandardWatchEventKinds.ENTRY_MODIFY) {
        @SuppressWarnings("unchecked")
        WatchEvent<Path> ev = (WatchEvent<Path>) event;
        Path filename = ev.context();

        // Проверяем, что изменение произошло именно в нужном файле
        if (filename.toString().equals("ABC_W1.dat")) {
            isProcessingEvent = true;
            System.out.println("File was modified: " + filename;
            Thread.sleep(20000);
            comParser.CreateCSV();
            isProcessingEvent = false;


        }
    }
}
```

В результате при изменении файла (т.е. запуске модели PSCAD), с задержкой в 20 с. вызывается метод CreateCSV() и запускается цикл отправки и приема сообщений.

В методе CreateCSV() в цикле происходит

1) чтение строки данных .dat файла объектом класса LCOM

2) отправка SV пакета по данным LCOM (замер 3 токов и 3 напряжений в один момент времени) объектом класса SvPublisher

3) захват SV пакета объектом класса SvSubscriber и запись данных в файл группами по 7200 пакетов

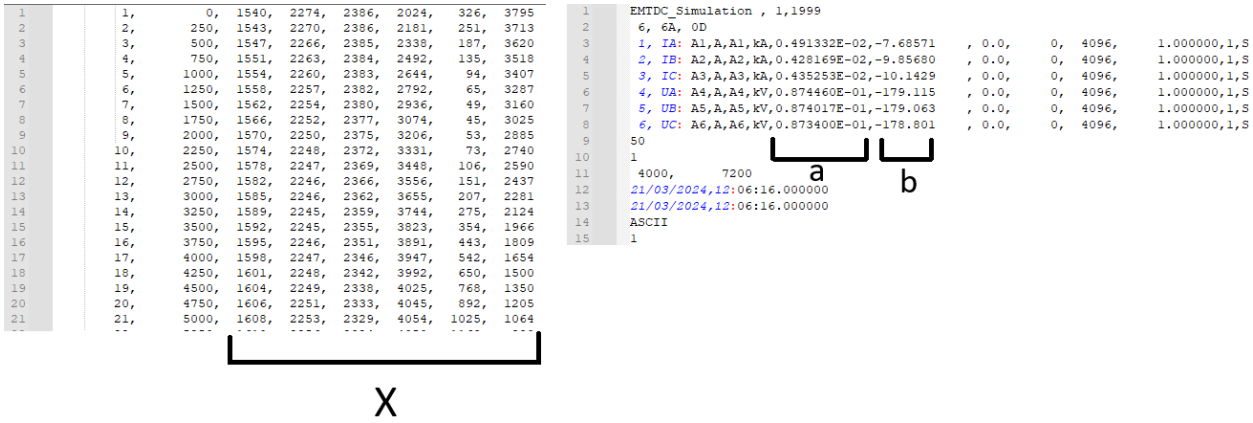
```
// создание и запуск SV подписчика
        SvSubscriber SvSubscriber = new SvSubscriber();
        SvSubscriber.process();


        // создание узла для чтения и
        LCOM lcom = new LCOM();
        inList.add(lcom);


        SvPublisher svPublisher = new SvPublisher();
        svPublisher.checkNic();
//          System.out.println(EtListen.getNicArray());


        int nicSRC = 5;
        String srcMAC = svPublisher.getNicMAC().get(nicSRC);

        int nicDST = 6;
        String dstMAC = svPublisher.getNicMAC().get(nicDST);

        String broadMAC = "ff:ff:ff:ff:ff:ff";

        svPublisher.setNickName(svPublisher.getNicArray().get(nicSRC));
        svPublisher.initializeNetworkInterface();

        svPublisher.setSrcMAC(srcMAC);
        svPublisher.setDstMAC(broadMAC);

        inList.add(svPublisher);


        svPublisher.phsAInst = lcom.OUT.get(0);
        svPublisher.phsBInst = lcom.OUT.get(1);
        svPublisher.phsCInst = lcom.OUT.get(2);

        svPublisher.phsAUnst = lcom.OUT.get(3);
        svPublisher.phsBUnst = lcom.OUT.get(4);
        svPublisher.phsCUnst = lcom.OUT.get(5);


        lcom.setFilePath(

"E:\\DZ\\11sem\\AI_Enregy\\KP\\pythonProject\\PSCAD_files\\testGrid.gf42\\Rank_00001\\Run_00001\\",
                "ABC_W1");
```

```
while (lcom.hasNextData()) {
    inList.forEach(LN::process);
}
```

Более подробно по каждому из пунктов:

1) LCOM

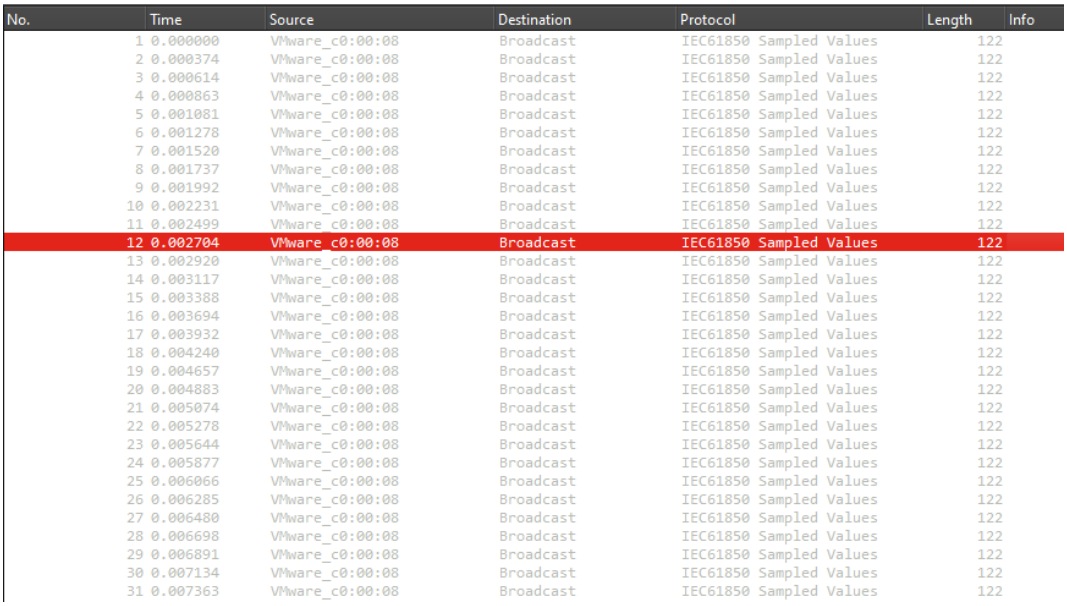
Объект класса LCOM по указанному пути построчно извлекает данные из.cfg файла и сохраняет мгновенное значение параметров режима полученное из строки .dat файла по формуле



$$val = ax + b$$

3) SvPublisher

Объект класса SvPublisher с помощью вспомогательных классов и методов преобразует значения токов и напряжений в последовательность байт и формирует пакет, для последующей отправки в сеть. Рассмотрим один из таких пакетов с помощью Wireshark:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
2	0.000374	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
3	0.000614	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
4	0.000863	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
5	0.001081	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
6	0.001278	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
7	0.001520	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
8	0.001737	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
9	0.001992	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
10	0.002231	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
11	0.002499	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
12	0.002704	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
13	0.002920	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
14	0.003117	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
15	0.003388	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
16	0.003694	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
17	0.003932	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
18	0.004240	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
19	0.004657	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
20	0.004883	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
21	0.005074	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
22	0.005278	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
23	0.005644	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
24	0.005877	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
25	0.006066	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
26	0.006285	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
27	0.006480	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
28	0.006698	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
29	0.006891	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
30	0.007134	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	
31	0.007363	VMware_c0:00:08	Broadcast	IEC61850 Sampled Values	122	

Рисунок 13 Поток SV в виртуальной сети

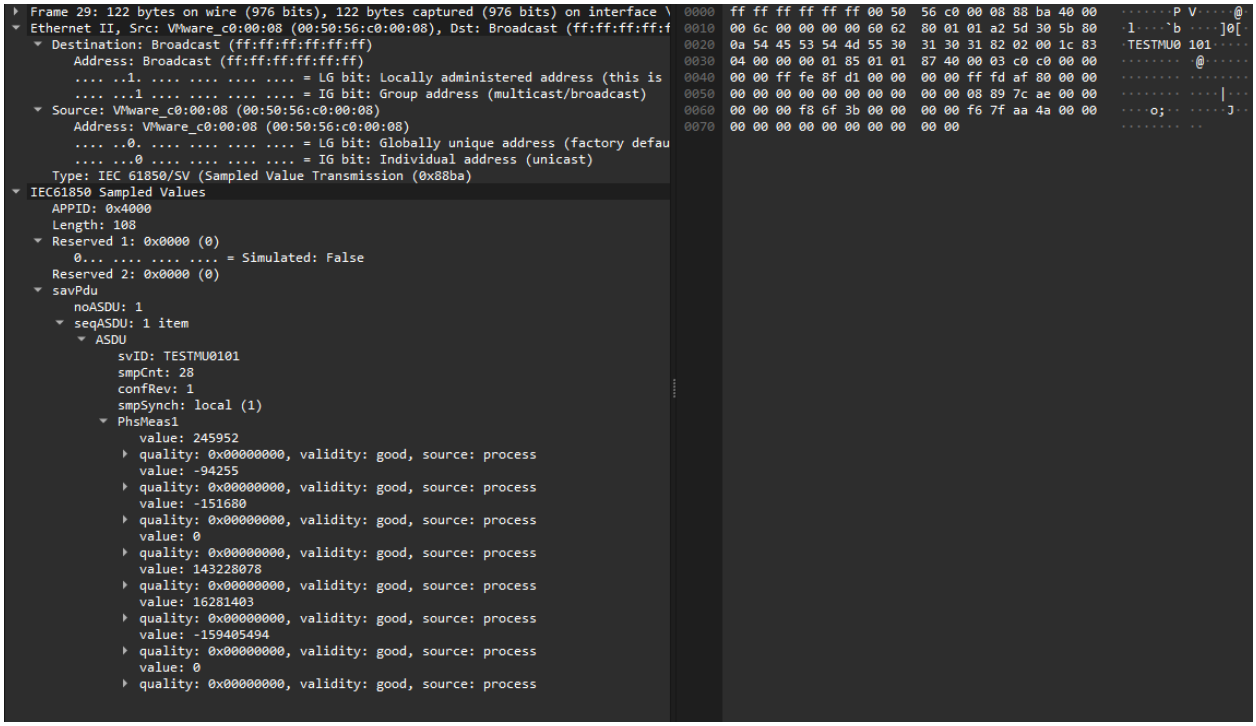


Рисунок 14 Структура полученного SV пакета

4) SvSubscriber

Объект класса SvSubscriber, в отдельном потоке производит захват пакетов Для этого, в соответствие с протоколом МЭК 61850-9-2 и описанием в разделе 2.2 были созданы соответствующие классы и атрибуты:

SvPacket:

```
public class SvPacket {
private String macDst;

private String macSrs;

private String type;

private String timestamp;

private String appId;

private int length;

private String reserved1;

private String reserved2;

private APDU apdu = new APDU();
}
```

APDU:

```
public class APDU {
    private int noASDU;

    private ArrayList<ASDU> seqASDU = new ArrayList<ASDU>();
}
```

ASDU:

```
public class ASDU {

    private String svID;

    private int smpCnt;

    private int confRev;

    private int smpSynch;

    private PhsMeas Dataset;


}
```

PhsMeas:

```
public class PhsMeas {
    private int instIa;
    private Quality qIa;

    private int instIb;
    private Quality qIb;

    private int instIc;
    private Quality qIc;

    private int instIn;
    private Quality qIn;


    private int instUa;
    private Quality qUa;
```

```
    private int instUb;
    private Quality qUb;

    private int instUc;
    private Quality qUc;

    private int instUn;
    private Quality qUn;

}
```

Quality:

```
public class Quality {
    private int  validity;

    private int overflow;

    private int outOfRange;

    private int badReference;

    private int oscillatory;

    private int failure;

    private int oldData;

    private int inconsistent;

    private int inaccurate;

    private int source;

    private int test;

    private int operatorBlocked;

    private int derived;
```

Далее при достижении 7200 захваченных пакетов происходит их запись в файл TestRun.csv в следующем виде

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	time	source	destination	svID	smpCnt	Ia	Ia_quality	Ib	Ib_quality	Ic	Ic_quality	Ua	Ua_quality	Ub	Ub_quality	Uc	Uc_quality				
2	2024-03-22T22:34:12.117307Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	1	-104.457	0000000000000	-137.363	0000000000000	242.236	0000000000000	11604.725	0000000000000	-157125.173	0000000000						
3	2024-03-22T22:34:12.117541Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	2	-84.803	0000000000000	-154.49	0000000000000	237.884	0000000000000	25333.747	0000000000000	-162718.882	0000000000						
4	2024-03-22T22:34:12.117751Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	3	-65.15	0000000000000	-167.335	0000000000000	233.531	0000000000000	38800.431	0000000000000	-167263.77	0000000000						
5	2024-03-22T22:34:12.117949Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	4	-50.41	0000000000000	-180.18	0000000000000	229.178	0000000000000	52092.223	0000000000000	-170847.24	0000000000						
6	2024-03-22T22:34:12.118164Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	5	-30.757	0000000000000	-193.025	0000000000000	224.826	0000000000000	65034.231	0000000000000	-173381.889	0000000000						
7	2024-03-22T22:34:12.118359Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	6	-11.104	0000000000000	-205.87	0000000000000	216.121	0000000000000	77626.455	0000000000000	-174780.316	0000000000						
8	2024-03-22T22:34:12.118565Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	7	8.549	0000000000000	-214.434	0000000000000	203.063	0000000000000	89694.004	0000000000000	-175129.923	0000000000						
9	2024-03-22T22:34:12.118791Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	8	28.202	0000000000000	-222.997	0000000000000	194.358	0000000000000	101236.876	0000000000000	-174430.709	0000000000						
10	2024-03-22T22:34:12.118991Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	9	47.855	0000000000000	-231.56	0000000000000	181.301	0000000000000	112167.625	0000000000000	-172682.675	0000000000						
11	2024-03-22T22:34:12.119222Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	10	67.508	0000000000000	-235.842	0000000000000	168.243	0000000000000	122398.807	0000000000000	-169798.419	0000000000						
12	2024-03-22T22:34:12.119439Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	11	87.162	0000000000000	-240.124	0000000000000	155.185	0000000000000	131842.975	0000000000000	-165865.343	0000000000						
13	2024-03-22T22:34:12.119636Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	12	101.902	0000000000000	-240.124	0000000000000	137.775	0000000000000	140500.129	0000000000000	-160970.848	0000000000						
14	2024-03-22T22:34:12.119840Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	13	121.555	0000000000000	-244.405	0000000000000	124.718	0000000000000	148282.823	0000000000000	-155027.532	0000000000						
15	2024-03-22T22:34:12.120060Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	14	136.295	0000000000000	-244.405	0000000000000	107.308	0000000000000	155191.058	0000000000000	-148122.798	0000000000						
16	2024-03-22T22:34:12.120759Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	15	151.035	0000000000000	-240.124	0000000000000	89.898	0000000000000	161137.386	0000000000000	-140344.046	0000000000						
17	2024-03-22T22:34:12.120993Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	16	165.775	0000000000000	-235.842	0000000000000	68.135	0000000000000	166034.362	0000000000000	-131691.278	0000000000						
18	2024-03-22T22:34:12.121206Z	00:50:56:c0:00:08	ff:ff:ff:ff:ff:ff	TESTMU0101	17	180.515	0000000000000	-231.56	0000000000000	50.725	0000000000000	169969.431	0000000000000	-122251.894	0000000000						

Рисунок 15 Содержимое .csv файла

4.2.Описание процедуры подготовки данных для обучения

Для обучения модели необходимо подготовить данные, отражающие изменения при подмене SV пакета. Как уже было сказано ранее, при возможных сценариях атаки факт подмены/манипуляции с данными можно определить по задержке при получении SV пакета.

Таким образом, необходимо подготовить данные, которые содержат достаточное количество SV пакетов как нормальных, так и с подменой данных. В результате опытов, изменение каких-либо полей при отправке SV пакета незначительно влияет на задержку. Поэтому задержка добавляется искусственно с помощью следующего цикла:

```
if ((count > 1111) && (count <3214)){
    Thread.sleep(random.nextLong(10) + 1);
}else{

    Thread.sleep((long) 0.25);
}
```

В результате 1/3 всех сообщений приходит с переменной задержкой. Дальнейшая работа с данными и нейронной сетью проходит с помощью языка python

На первом этапе, используя библиотеку pandas с помощью функции generate_dataframe формируются интервалы между получением каждого SV пакета:

```
def generate_dataframe(path_csv, name):
    print('\nForming dataframe [time, interval]')
    data = pd.read_csv(path_csv, encoding="ISO-8859-1")
    print(data.info())

    data['time'] = pd.to_datetime(data['time'])

    print(data)

    time = pd.DataFrame(data, columns=['time'])

    print(time)

    time['interval'] = time['time'].diff()

    time.loc[0, 'interval'] = time.loc[1, 'interval']

    time['interval'] = time['interval'].dt.total_seconds() * 1000

    print(time)
```

```
time.to_csv(name, index=False)

return time
```

В результате данные приобретают следующий вид:

	time	interval
0	2024-03-21 09:07:04.224446+00:00	0.52800
1	2024-03-21 09:07:04.224974+00:00	0.52800
2	2024-03-21 09:07:04.225306+00:00	0.33200
3	2024-03-21 09:07:04.225612+00:00	0.30600
4	2024-03-21 09:07:04.225913+00:00	0.30100
5	2024-03-21 09:07:04.226269+00:00	0.35600
6	2024-03-21 09:07:04.226501+00:00	0.23200
7	2024-03-21 09:07:04.226759+00:00	0.25800
8	2024-03-21 09:07:04.226974+00:00	0.21500
9	2024-03-21 09:07:04.227227+00:00	0.25300
10	2024-03-21 09:07:04.227454+00:00	0.22700
11	2024-03-21 09:07:04.227656+00:00	0.20200
12	2024-03-21 09:07:04.227862+00:00	0.20600
13	2024-03-21 09:07:04.228069+00:00	0.20700
14	2024-03-21 09:07:04.228312+00:00	0.24300
15	2024-03-21 09:07:04.228537+00:00	0.22500
16	2024-03-21 09:07:04.228756+00:00	0.21900
17	2024-03-21 09:07:04.228985+00:00	0.22900
18	2024-03-21 09:07:04.229274+00:00	0.28900
19	2024-03-21 09:07:04.229520+00:00	0.24600
20	2024-03-21 09:07:04.229828+00:00	0.30800

Рисунок 16 первый этап подготовки данных

Рассмотрим две группы данных, первая группа не содержит подмены SV, вторая содержит. Полученные на данном этапе данные будут иметь вид:

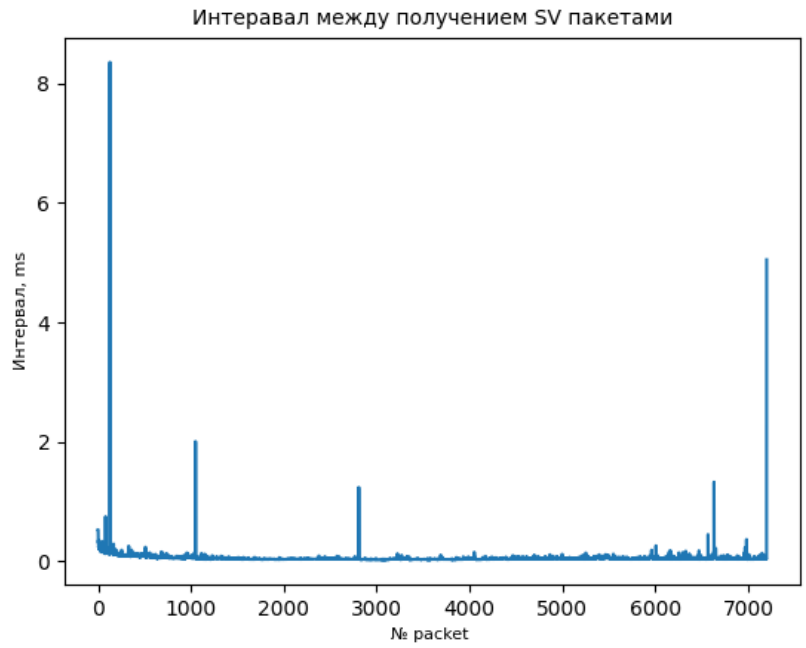


Рисунок 17 Интервал между SV в нормальном режиме

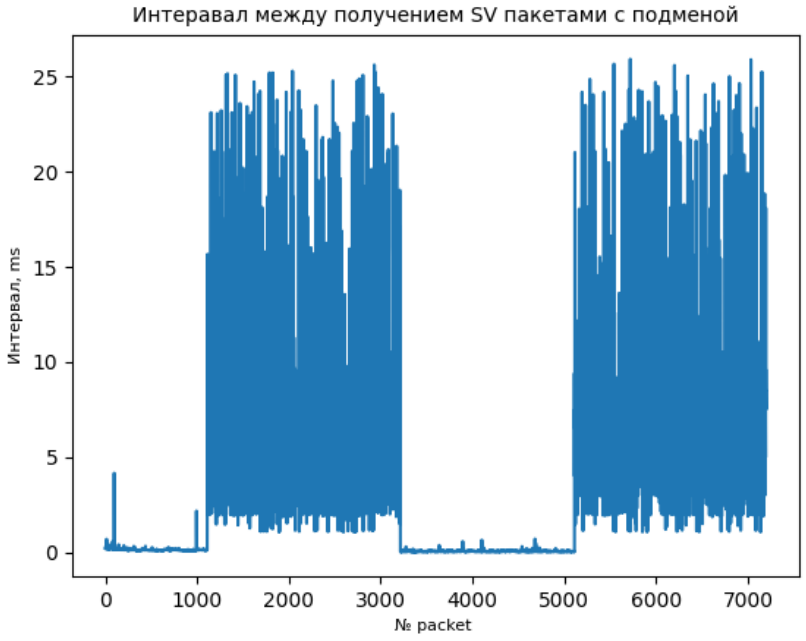


Рисунок 18 Интервал между SV при подмене

На втором этапе происходит определение меток для полученных интервалов на основе среднего значения и стандартного отклонения для каждого окна из 5 интервалов. На основе этих данных каждому интервалу присваивается метка (где 0 – подмены не обнаружено, 1 – пакет был подменен) в результате данные имеют следующий вид:

	time	interval	mean_interval	std_interval	label
4	2024-03-22 22:34:12.118164+00:00	0.21500	0.21820	0.01569	0
5	2024-03-22 22:34:12.118359+00:00	0.19500	0.21040	0.01557	0
6	2024-03-22 22:34:12.118565+00:00	0.20600	0.20480	0.00829	0
7	2024-03-22 22:34:12.118791+00:00	0.22600	0.20800	0.01271	0
8	2024-03-22 22:34:12.118991+00:00	0.20000	0.20840	0.01234	0
9	2024-03-22 22:34:12.119222+00:00	0.23100	0.21160	0.01601	0
10	2024-03-22 22:34:12.119439+00:00	0.21700	0.21600	0.01306	0
11	2024-03-22 22:34:12.119636+00:00	0.19700	0.21420	0.01522	0
12	2024-03-22 22:34:12.119840+00:00	0.20400	0.20980	0.01410	0
13	2024-03-22 22:34:12.120060+00:00	0.22000	0.21380	0.01344	0
14	2024-03-22 22:34:12.120759+00:00	0.69900	0.30740	0.21911	0
15	2024-03-22 22:34:12.120993+00:00	0.23400	0.31080	0.21748	0
16	2024-03-22 22:34:12.121206+00:00	0.21300	0.31400	0.21550	0

Рисунок 19 Данные после присвоения меток

После этого происходит формирование данных для обучения методом скользящего окна и разбиение на тренировочную и тестовую выборку. Метод скользящего окна продемонстрирован на рисунке ниже:

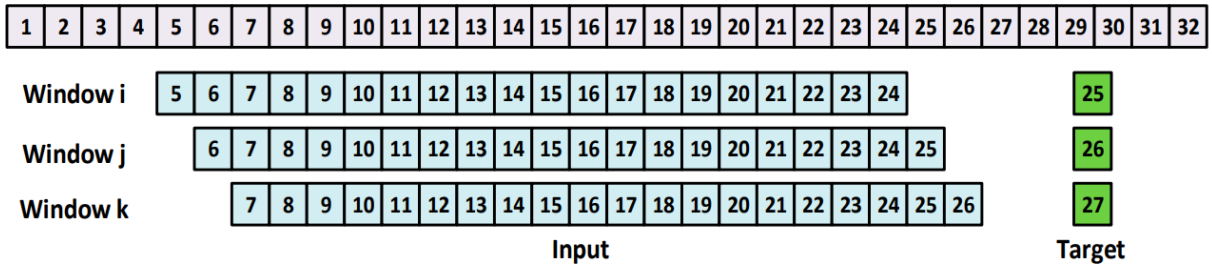


Рисунок 20 Метод скользящего окна

Начиная с 1-го образца, входными и выборочными будут 20 образцов. Тогда будет прогнозирование 21 значение (т. е. установлен как целевой результат). Далее окно сдвинется на один элемент, на вход пойдет выборка со 2 по 21 включительно, а прогнозируем 22 значение и т.д.

4.3.Обучение рекуррентной нейронной сети и тестирование

Нейронная сеть состоит из двух слоев:

Первый слой сети - это LSTM-слой с 50 нейронами. Входной тензор для этого слоя имеет форму (None, window_size, num_features), где None – это batch_size, Значение None позволяет подавать на вход выборки разной длины, window_size- это размер окна, в данном случае выбирается равным числу нейронов, num_features - это количество признаков, т.е. в данном случае он

равен 1. Здесь также используется L2 регуляризация ядра и dropout равный 0.2 для предотвращения переобучения.

Второй слой сети – это так называемый плотный слой (Dense) с одним нейроном, использующий функцию сигмовидную активации. Этот слой используется для бинарной классификации, так как sigmoid-функция возвращает значение в диапазоне от 0 до 1, которое можно интерпретировать как вероятность принадлежности к одному или другому классу.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 32)	4,352
dense (Dense)	(None, 1)	33

Total params: 4,385 (17.13 KB)
Trainable params: 4,385 (17.13 KB)
Non-trainable params: 0 (0.00 B)

Рисунок 21 Двухслойная сеть для детектирования подмены

Для оценки качества обучения модели используются две метрики: точность (accuracy) и среднеквадратичное отклонение (MeanSquaredError). Точность рассчитывается как доля правильно классифицированных образцов от общего количества образцов, а среднеквадратичное отклонение рассчитывается по формуле:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

В работе для расчета метрик используются в строенные в библиотеку keras функции.

В результате обучения модели на тренировочных данных: получены следующие метрики качества:

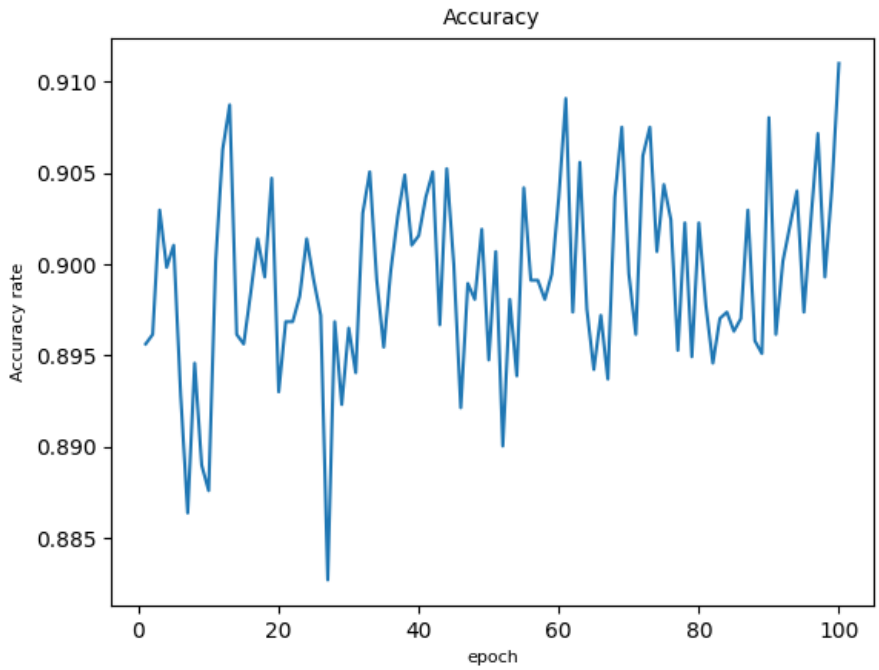


Рисунок 22 Точность при обучении на тренировочной выборке

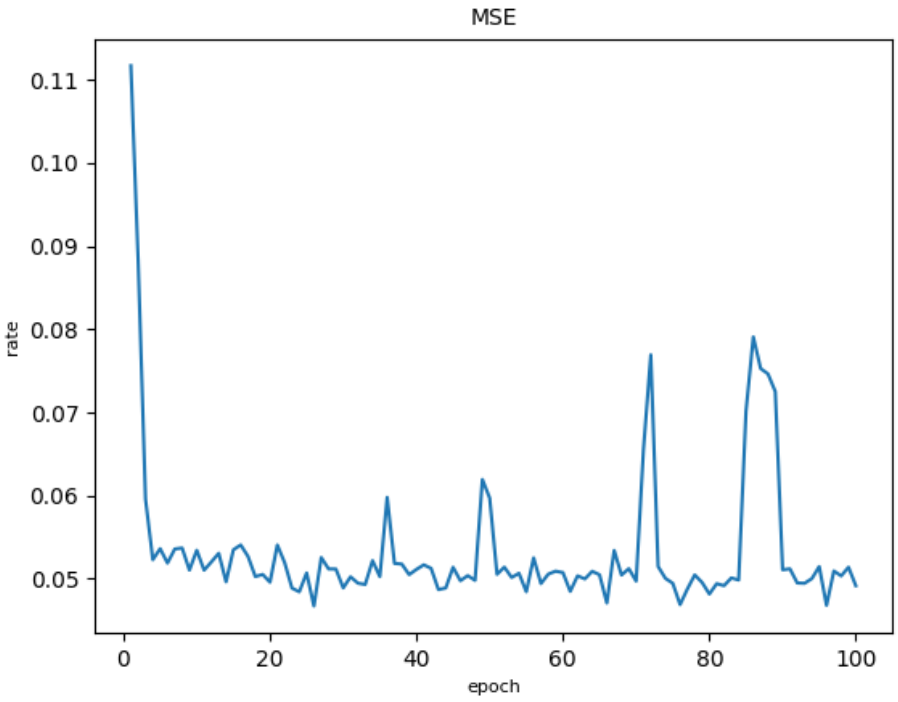


Рисунок 23 MSE при обучении на тренировочной выборке

В результате обучения в течении 100 эпох были получены достаточно высокие метрики качества. Рассмотрим работу модели на тренировочной, тестовой выборке, и новой выборке, не используемой при обучении:

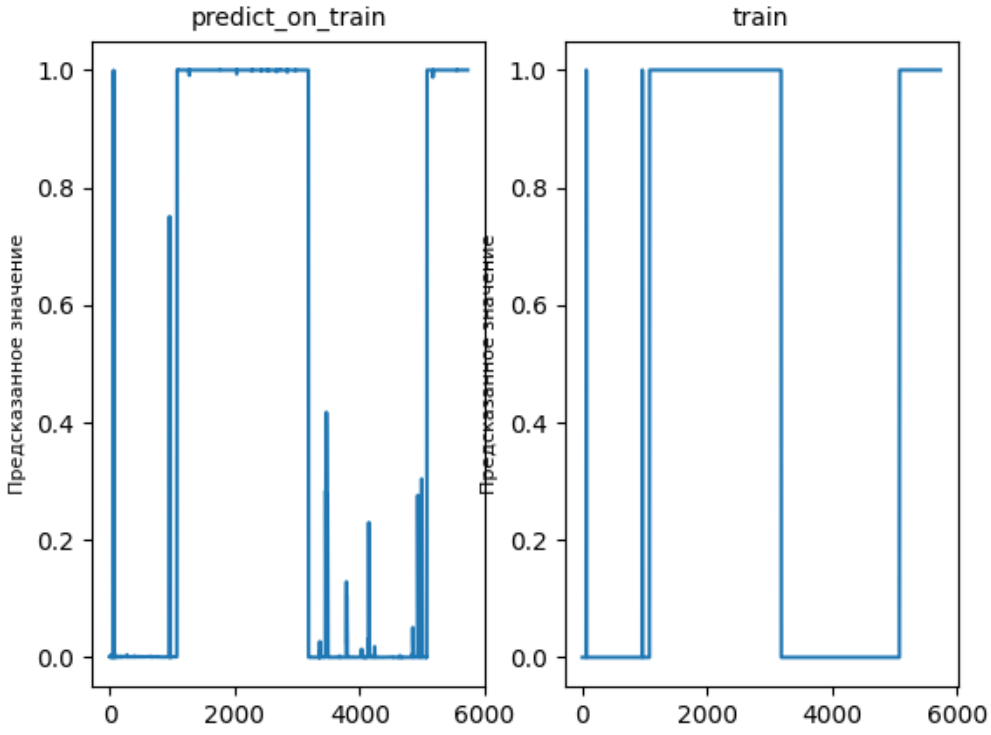


Рисунок 24 Сравнение предсказанного значения с тренировочной выборкой

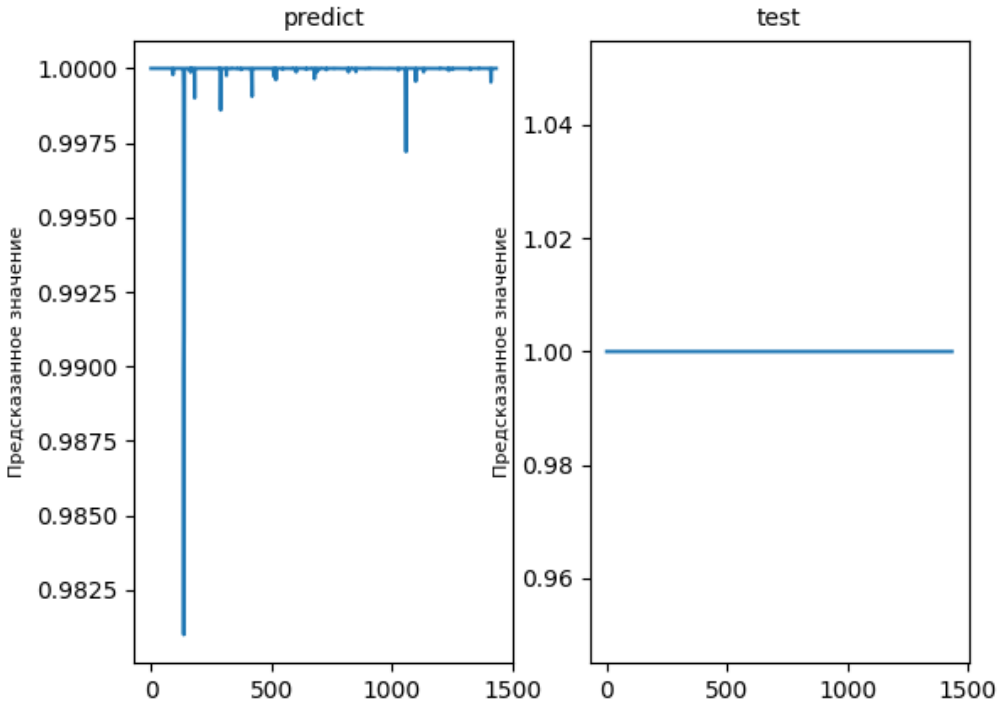


Рисунок 25 Сравнение предсказанного значения с тестовой выборкой

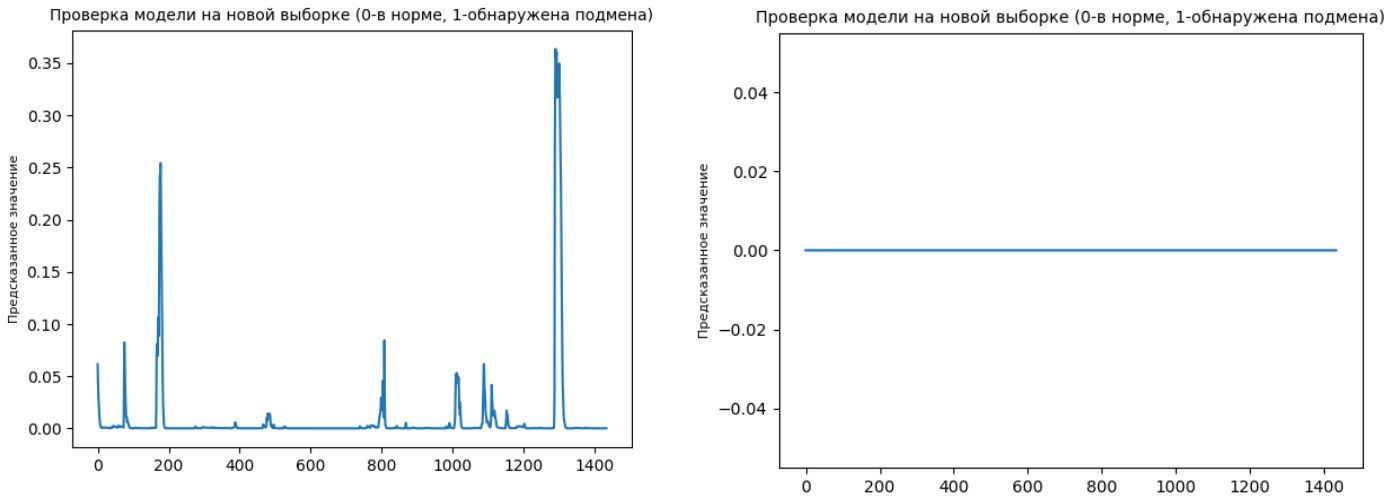


Рисунок 26 Сравнение предсказанного значения с новой выборкой

5. Выводы

В результате проведенной работы была разработан алгоритм детектирования подмены SV потоков IEC61850 9-2 LE с помощью рекуррентных нейронных сетей.

Было проведено обучение рекуррентной нейронной сети на подготовленных данных и протестирована ее эффективность в детектировании подмены SV потоков. Результаты тестирования показали, что разработанный алгоритм способен с достаточной точной определять задержку во время получения SV пакета.

СПИСОК ЛИТЕРАТУРЫ:

1. Description of the IEC 61850 Sampled Values (SV) protocol implementation in the Typhoon HIL toolchain. https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/iec_61850_sampled_values_protocol.html#reference_qhf_nmt_qz__fig_n43_sty_ddb (дата обращения 21.03.2024)
2. Техническое описание Профиля SV МЭК 61850-9-2 Binom3/Санкт-Петербург 2017, 35 стр.
3. Understanding LSTM Networks by Christoper Colah https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата обращения 21.03.2024)
4. IEC 61850 Communication networks and systems for power utility automation – Part 9-2: Specific communication service mapping (SCSM) – Sampled values over ISO/IEC 8802-3
5. The IEC 61850 Sampled Measured Values Protocol: Analysis, Threat Identification, and Feasibility of Using NN Forecasters to Detect Spoofed Packets; Published: 29 September 2019

Приложение А

КОД ИЗДАТЕЛЯ И ПОДПИСЧИКА SV ПОТОКА

AnalogValue

```
/** Аналоговый сигнал **/
public class AnalogueValue extends GenDataObject {

   private GenDataAttribute<Double> f = new GenDataAttribute<>(0d); // Значение аналогового
сигнала

   public GenDataAttribute<Double> getF() {
      return f;
   }

   public void setF(GenDataAttribute<Double> f) {
      this.f = f;
   }
}
```

GenDataAttribute

```
/** Атрибут данных*/
public class GenDataAttribute<T> extends GenDataObject {

   private T value;

   public GenDataAttribute(T v) {

   }

   public GenDataAttribute() {
   }

   public T getValue() {
      return value;
   }

   public void setValue(T value) {
      this.value = value;
   }
}
```

GenDataObject

```
import java.util.ArrayList;
import java.util.List;

/** Абстрактный класс для создания объектов данных**/
public abstract class GenDataObject {

   private String name;

   private String reference;

   private GenDataObject parent;

   private final List<GenDataObject> children = new ArrayList<>();

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getReference() {
      return reference;
   }

   public void setReference(String reference) {
      this.reference = reference;
   }

   public GenDataObject getParent() {
      return parent;
   }

   public void setParent(GenDataObject parent) {
      this.parent = parent;
   }

   public List<GenDataObject> getChildren() {
      return children;
   }
}
```

Quality

```
package org.example.common;


/** Класс качества*/
public class Quality extends GenDataObject {

   private int value;

   public int getValue() {
      return value;
   }

   public void setValue(int value) {
      this.value = value;
   }
}
```

Timestamp

```
package org.example.common;

/** Метка времени */
public class Timestamp extends GenDataObject {

   private long value = 0L;

   public long getValue() {
      return value;
   }

   public void setValue(long value) {
      this.value = value;
   }
}
```

MV

```
package org.example.dataobjects.measurements;


import org.example.common.AnalogueValue;
import org.example.common.GenDataObject;
import org.example.common.Quality;
import org.example.common.Timestamp;

/** Измеряемое значение **/
```

```
public class MV extends GenDataObject {

   private AnalogueValue instMag = new AnalogueValue();

   private Quality q = new Quality();

   private Timestamp t = new Timestamp();

   public AnalogueValue getInstMag() {
      return instMag;
   }

   public void setInstMag(AnalogueValue instMag) {
      this.instMag = instMag;
   }

   public Quality getQ() {
      return q;
   }

   public void setQ(Quality q) {
      this.q = q;
   }

   public Timestamp getT() {
      return t;
   }

   public void setT(Timestamp t) {
      this.t = t;
   }
}
```

LN

```
package org.example.logiclanodes.common;

import org.example.common.GenDataObject;

/** Абстрактный класс для создания узлов **/
public abstract class LN extends GenDataObject {

   /** Основная функция узла **/
   public abstract void process();
}
```

LCOM

```
package org.example.logiclanodes.input;

import org.example.dataobjects.measurements.MV;
import org.example.logiclanodes.common.LN;

import java.io.*;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/** Узел для считывания COMTRADE файла **/
public class LCOM extends LN {

   private File cfgFile; //Считываемый .cfg файл
   private File datFile; //Считываемый .dat файл

   /* Списки содержимого .dat и .cfg файлов (Построчно)*/
   private List<String> cfgDataFile = new ArrayList<>();
   private List<String> datDataFile = new ArrayList<>();

   /* Списки масштабирующих коэф. a, b */
   private List<Double> aCoefList = new ArrayList<>();
   private List<Double> bCoefList = new ArrayList<>();

   private int analogNumber = 0; // Число аналоговых сигналов
   private int discreteNumber = 0; // Число дискретных сигналов
   private Iterator<String> dataIterator; // Итератор для прохода по списку

   public final List<MV> OUT = new ArrayList<>();  // Список измеренных значений

   /* Конструктор узла создающий буфер измеренных значений  */
   public LCOM() {
      for (int i = 0; i < 20; i++) {
         OUT.add(new MV());
      }
   }

   /**Запись скорректированного измеренного значения каждого из каналов **/
   @Override
   public void process() {
      if(dataIterator.hasNext()) {
         String[] line = dataIterator.next().split(",");
         long t = Long.parseLong(line[1].trim());

         for (int i = 0; i < analogNumber; i++) {
            double value = Double.parseDouble(line[i + 2]);
            value *= aCoefList.get(i);
            value += bCoefList.get(i);
```

```
            OUT.get(i).getT().setValue(t);
            OUT.get(i).getInstMag().getF().setValue(Double.valueOf((int) (value * 1000000))); //возможно
1000 чтобы результат был в A и B
        }

//          System.out.println(OUT.get(3).getInstMag().getF().getValue());
    }
  }

  public boolean hasNextData() {
    return dataIterator.hasNext();
  }

  /** Считывание содержимого .cfg и .dat файлов **/
  public void setFilePath(String path, String name) {

    cfgFile = new File(path + name + ".cfg");
    datFile = new File(path + name + ".dat");


    cfgDataFile = readFileByLine(cfgFile);
    datDataFile = readFileByLine(datFile);

    dataIterator = datDataFile.listIterator();

    extractCfgFileData();
  }

  /** Запись содержимого .dat файла (число сигналов)**/
  private void extractCfgFileData() {
    analogNumber = Integer.parseInt(cfgDataFile.get(1).split(", ")[1].replace("A", ""));
    discreteNumber = Integer.parseInt(cfgDataFile.get(1).split(", ")[2].replace("D", ""));

    for (int i = 2; i < 2 + analogNumber; i++) {
      aCoefList.add(Double.parseDouble(cfgDataFile.get(i).split(",")[5]));
      bCoefList.add(Double.parseDouble(cfgDataFile.get(i).split(",")[6]));
    }
  }

  /** Построчное чтение файла и запись в список**/
  private static List<String> readFileByLine(File file){
    List<String> list = new ArrayList<>();

    try {
      FileReader fr = new FileReader(file);
      BufferedReader br = new BufferedReader(fr);

      String line = br.readLine();
      while(line != null){
```

```
            list.add(line);
            line = br.readLine();
        }

    } catch (FileNotFoundException e) {
        System.err.println("Неверно указан путь");;
    } catch (IOException e) {
        throw new RuntimeException(e);
    }

    return list;
  }
}
```

MMXU

```
package org.example.pcapFiles;

import com.opencsv.CSVWriter;
import lombok.Getter;
import lombok.Setter;
import lombok.SneakyThrows;
import lombok.extern.slf4j.Slf4j;
import org.example.logiclanodes.common.LN;
import org.example.logiclanodes.measurements.MMXU;
import org.example.packetStructureCapture.SvPacket;
import org.pcap4j.core.*;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

@Slf4j
public class SvSubscriber extends LN {
  /**
   * check this.setNickName("VMware Virtual Ethernet Adapter for VMnet8");
   */

  @Getter @Setter
  private String nickName;
  @Getter @Setter
  private ArrayList<String> nicArray = new ArrayList<>();
  @Getter @Setter
```

```
    private ArrayList<String> nicMAC = new ArrayList<>();
    @Getter @Setter
    private ArrayList<String> nicIP = new ArrayList<>();
    @Getter @Setter
    private PcapHandle handle;
    @Getter @Setter
    private static final int BUFFER_SIZE = 1000;
    @Getter @Setter
    private static LinkedBlockingQueue<String> csvBuffer = new LinkedBlockingQueue<>(BUFFER_SIZE);
    HashMap<Integer, SvPacket> svPacketMap = new LinkedHashMap<>();
//    int lastVal = 0;

    MMXU mmxu = new MMXU();


    final List <PacketListener> listeners = new CopyOnWriteArrayList<>();

    private PacketListener defaultPacketListener = packet -> {
      listeners.forEach(listener -> listener.gotPacket(packet));
    };

    public void checkNic(){
        try {
            for (PcapNetworkInterface nic : Pcaps.findAllDevs()) {
//                System.out.println(nic.getAddresses());
//                System.out.println(nic.getLinkLayerAddresses());
                nicMAC.add(String.valueOf(nic.getLinkLayerAddresses().get(0)));
                nicArray.add(nic.getDescription());
                nicIP.add(parseIP(String.valueOf(nic.getAddresses())));
            }
            System.out.println("NIC written down");
        } catch (PcapNativeException e) {
            throw new RuntimeException(e);
        }
    }

    public String parseIP(String ipString){
        String IPADDRESS_PATTERN =
                "(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)";

        Pattern pattern = Pattern.compile(IPADDRESS_PATTERN);
        Matcher matcher = pattern.matcher(ipString);
        if (matcher.find()) {
          return matcher.group();
        }
        else{
          return "0.0.0.0";
        }
    }
```

```
    @SneakyThrows
    public void start(){
        if (nickName != null && !nickName.isEmpty()){
            initializeNetworkInterface();

                if (handle != null) {
                    String filter = "ether proto 0x88ba";  //  proto 0x88ba = IEC SV

                    handle.setFilter(filter, BpfProgram.BpfCompileMode.OPTIMIZE);

                    Thread captureThread = new Thread(() -> {
                        try {
                            log.info("Starting Packet Capturing");
                            handle.loop(0, defaultPacketListener); //packetCount should be x2. 0 for unlimited
capture
                        } catch (PcapNativeException e) {
                            log.info("Finished Packet Capturing");
                        } catch (InterruptedException e) {
                            log.info("Finished Packet Capturing");
                        } catch (NotOpenException e) {
                            log.info("Finished Packet Capturing");
                        }


                    });

                    captureThread.start();
                }
        } else log.error("Select NIC first");
    }

    @SneakyThrows
    public void stop(){
        if (nickName != null && !nickName.isEmpty()) {
            if (handle != null) {
                handle.breakLoop();
//                  handle.close();
                log.info("Packet Capturing Stopped");
            }else {
                log.error("Packet Capturing ALREADY Stopped");
            }
        } else log.error("Select NIC first");
    }

    @SneakyThrows
    public void initializeNetworkInterface() {
        Optional<PcapNetworkInterface> nic = Pcaps.findAllDevs().stream()
                .filter(i -> nickName.equals(i.getDescription()))
                .findFirst();
```

```
        if (nic.isPresent()) {
            handle = nic.get().openLive(1500, PcapNetworkInterface.PromiscuousMode.PROMISCUOUS, 10);
            log.info("Network handler: {}", nic);
        }
        else {
            log.error("Network Interface not found");
        }
    }

    public void addListener(PacketListener listener) {
        listeners.add(listener);
    }

    @SneakyThrows
    @Override
    public void process() {
//      log.debug("EtListener method");

        File resultCSV = new File("src/main/resources/TestRun.csv");
        FileWriter fileWriter = new FileWriter(resultCSV);
        CSVWriter writer = new CSVWriter(fileWriter);

        writer.writeNext(new String[]{
                "time",
                "source",
                "destination",
                "svID",
                "smpCnt",
                "Ia",
                "Ia_quality",
                "Ib",
                "Ib_quality",
                "Ic",
                "Ic_quality",
                "Ua",
                "Ua_quality",
                "Ub",
                "Ub_quality",
                "Uc",
                "Uc_quality",
        });


        this.checkNic();
        this.setNickName("VMware Virtual Ethernet Adapter for VMnet8");
        this.getNicArray();


        SvParser svParser = new SvParser();
```

```
        AtomicInteger curCnt = new AtomicInteger();

//      HashMap<String, ArrayList<SvPacket>> sourceMap = new HashMap<>();

//      ArrayList<Optional> array = new ArrayList<>();

        this.addListener(packet -> {
            Optional<SvPacket> svPacket = svParser.decode(packet);
            int noASDU = svPacket.get().getApdu().getNoASDU();
            for (int i = 0; i < noASDU; i++) {


                if (svPacket.isPresent()
                        && curCnt.get() != svPacket.get().getApdu().getSeqASDU().get(i).getSmpCnt()
//                      && lastVal != svPacket.get().getApdu().getSeqASDU().get(i).getDataset().getInstIa()
                ) {


                    svPacketMap.put(svPacketMap.size(), svPacket.get());
                    mmxu.process(svPacket.get().getApdu().getSeqASDU().get(0).getDataset());
                    System.out.println(svPacketMap.size());

//                  if (svPacketMap.size() == 7199){
//                      try {
//                          Thread.sleep(1000);
//                      } catch (InterruptedException e) {
//                          throw new RuntimeException(e);
//                      }
//                      System.out.println(String.format("Captured packets: %d", svPacketMap.size()));
//                      for (int j = 0; j < svPacketMap.size(); j++) {
//
//                          writer.writeNext(new String[]{
//                                  String.valueOf(svPacketMap.get(j).getTimestamp()),
//                                  String.valueOf(svPacketMap.get(j).getMacSrs()),
//                                  String.valueOf(svPacketMap.get(j).getMacDst()),
//                                  String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSvID()),
//                                  String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSmpCnt()),
//                                  String.valueOf(mmxu.IphsA.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIa().toString()),
//                                  String.valueOf(mmxu.IphsB.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIb().toString()),
//                                  String.valueOf(mmxu.IphsC.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIc().toString()),
//                                  String.valueOf(mmxu.UphsA.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUa().toString()),
//                                  String.valueOf(mmxu.UphsB.get(j)[0]),
```

```
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUb().toString()),
//                          String.valueOf(mmxu.UphsC.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUc().toString())
//                      });
//                  }
//                  try {
//                      writer.close();
//                      stop();
//                      Thread.interrupted();
//
//                  } catch (IOException e) {
//                      throw new RuntimeException(e);
//                  }
//              }

                if (svPacketMap.size() == 7199){
                    System.out.println("DONE");
                    for (int j = 0; j < svPacketMap.size(); j++) {

                        writer.writeNext(new String[]{
                                String.valueOf(svPacketMap.get(j).getTimestamp()),
                                String.valueOf(svPacketMap.get(j).getMacSrs()),
                                String.valueOf(svPacketMap.get(j).getMacDst()),
                                String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSvID()),
                                String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSmpCnt()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstIa()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIa().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstIb()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIb().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstIc()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIc().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstUa()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUa().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstUb()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUb().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstUc()/1000d),
```

```
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUc().toString())
                    });
                }
                try {
                    writer.close();
                    stop();
                    Thread.interrupted();

                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }


//              if (setSvPckt.size() >= 37000) {
//                  System.out.println(setSvPckt);
//              }

            curCnt.set(svPacket.get().getApdu().getSeqASDU().get(i).getSmpCnt());  //update counter else
writes packet twice
        }

    }

});

    this.start();
  }

  public static synchronized void addToBuffer(String data) {
    try {
      csvBuffer.put(data);
    } catch (InterruptedException e) {
      System.err.println("Error adding data to buffer: " + e.getMessage());
    }
  }

  public static void handlePacket(Optional<SvPacket> svPacket) {
    // Extract the relevant information from the packet

    // Format the data as a CSV string


    // Add the data to the buffer
//      addToBuffer(csvData);
  }

//   public static void startWriterThread(String fileName) {
//      Thread writerThread = new Thread(() -> {
//        try (FileWriter fileWriter = new FileWriter(fileName, true)) {
```

```
//            while (true) {
//                String data = csvBuffer.take();
//                fileWriter.append(data);
//                fileWriter.append("\n");
//                fileWriter.flush();
//            }
//        } catch (IOException | InterruptedException e) {
//            System.err.println("Error writing data to CSV file: " + e.getMessage());
//        }
//    });
//    writerThread.start();
//  }


}
```

APDU

```
package org.example.packetStructureCapture;

import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;

@Getter @Setter
public class APDU {
   private int noASDU;

   private ArrayList<ASDU> seqASDU = new ArrayList<ASDU>();
}
```

ASDU

```
package org.example.packetStructureCapture;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class ASDU {

   private String svID;

   private int smpCnt;

   private int confRev;
```

```
    private int smpSynch;

    private PhsMeas Dataset;


}
```

PhsMeas

```
package org.example.packetStructureCapture;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class PhsMeas {
    private int instIa;
    private Quality qIa;

    private int instIb;
    private Quality qIb;

    private int instIc;
    private Quality qIc;

    private int instIn;
    private Quality qIn;


    private int instUa;
    private Quality qUa;

    private int instUb;
    private Quality qUb;

    private int instUc;
    private Quality qUc;

    private int instUn;
    private Quality qUn;

}
```

Quality

```
package org.example.packetStructureCapture;
```

```
import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class Quality {
   private int  validity;

   private int overflow;

   private int outOfRange;

   private int badReference;

   private int oscillatory;

   private int failure;

   private int oldData;

   private int inconsistent;

   private int inaccurate;

   private int source;

   private int test;

   private int operatorBlocked;

   private int derived;

   public String toString() {
      StringBuilder sb = new StringBuilder();
      sb.append(validity);
      sb.append(overflow);
      sb.append(outOfRange);
      sb.append(badReference);
      sb.append(oscillatory);
      sb.append(failure);
      sb.append(oldData);
      sb.append(inconsistent);
      sb.append(inaccurate);
      sb.append(source);
      sb.append(test);
      sb.append(operatorBlocked);
      sb.append(derived);


      return sb.toString();
```

```
    }
}
```

SvPacket

```
package org.example.packetStructureCapture;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class SvPacket {
    private String macDst;

    private String macSrs;

    private String type;

    private String timestamp;

    private String appId;

    private int length;

    private String reserved1;

    private String reserved2;

    private APDU apdu = new APDU();


}
```

APDU

```
package org.example.packetStructureSend;

public class APDU {

    private static byte[] appID = {0x40, 0x00};
    private static byte[] lgth = {0x00, 0x6c};
    private static byte[] reserved1 = {0x00, 0x00};
    private static byte[] reserved2 = {0x00, 0x00};
    private static byte savPDU_tag = 0x60;
    private static byte savPDU_lgt = 0x62;
    private static byte noASDU_tag = (byte) 0x80;
    private static byte noASDU_lgt = 0x01;
    private static byte noASDU = 0x01;
```

```
    private static byte SequenceofASDU_tag = (byte) 0xa2;
    private static byte SequenceofASDU_lgt = 0x5d;


    public byte[] convertToBytes(){
        byte[] result = new byte[15];
        int offset = 0;

        System.arraycopy(appID, 0,result,offset,2);
        offset+=2;

        System.arraycopy(lgth, 0,result,offset,2);
        offset+=2;

        System.arraycopy(reserved1, 0,result,offset,2);
        offset+=2;

        System.arraycopy(reserved2, 0,result,offset,2);
        offset+=2;

        result[offset] = savPDU_tag;
        offset+=1;

        result[offset] = savPDU_lgt;
        offset+=1;

        result[offset] = noASDU_tag;
        offset+=1;

        result[offset] = noASDU_lgt;
        offset+=1;

        result[offset] = noASDU;
        offset+=1;

        result[offset] = SequenceofASDU_tag;
        offset+=1;

        result[offset] = SequenceofASDU_lgt;
        offset+=1;

        return result;
    }
}
```

ASDU

```
package org.example.packetStructureSend;

import java.nio.charset.StandardCharsets;
import java.util.Random;

public class ASDU {
    private static byte SequenceASDU_tag = 0x30;
    private static byte SequenceASDU_lgt = 0x5b;
    private static byte svID_tag = (byte) 0x80;
    private static byte svID_lgt = 0x0A;
    private static byte smpCnt_tag = (byte) 0x82;
    private static byte smpCnt_lgt = 0x02;
    private static byte confRev_tag = (byte) 0x83;
    private static byte confRev_lgt = 0x04;
    private static int confRev = 1;
    private static byte smpSynch_tag = (byte) 0x85;
    private static byte smpSynch_lgt = 0x01;
    private static byte smpSynch = 0x01;
    private static byte SequenceofData_tag = (byte) 0x87;
    private static byte SequnceofData_lgt = 0x40;

    private String svID;
    private short smpCnt;

    private int Ia_Amp_instMag;
    private int Ia_Amp_q;
    private int Ib_Amp_instMag;
    private int Ib_Amp_q;
    private int Ic_Amp_instMag;
    private int Ic_Amp_q;
    private int Ineut_Amp_instMag;
    private int Ineut_Amp_q;
    private int Ua_Vol_instMag;
    private int Ua_Vol_q;
    private int Ub_Vol_instMag;
    private int Ub_Vol_q;
    private int Uc_Vol_instMag;
    private int Uc_Vol_q;
    private int Uneut_Vol_instMag;
    private int Uneut_Vol_q;

    String fakeID = "FAKEMU0000";


    public ASDU(String svID, short smpCnt, int[] mes) throws Exception {
        this.svID = svID;
        if (this.svID.length() < 10) {
            while (this.svID.length() < 10) {
```

```
            this.svID = this.svID + "0";
        }
    } else if (this.svID.length() > 10) {
        this.svID = this.svID.substring(0, 10);
    }
    this.smpCnt = smpCnt;

    if (mes.length != 8) {
        throw new Exception("Длинна массива =/= 8");
    }
    this.Ia_Amp_instMag = mes[0];
    this.Ib_Amp_instMag = mes[1];
    this.Ic_Amp_instMag = mes[2];
    this.Ineut_Amp_instMag = mes[3];
    this.Ua_Vol_instMag = mes[4];
    this.Ub_Vol_instMag = mes[5];
    this.Uc_Vol_instMag = mes[6];
    this.Uneut_Vol_instMag = mes[7];

//      if ((this.smpCnt > 1111) && (this.smpCnt <2500)) {
//          this.svID = fakeID;
//      }



//      if ((2100 < this.smpCnt) && (this.smpCnt < 3000)){
//          this.Ia_Amp_q = generateSubstitution();
//          this.Ib_Amp_q = generateSubstitution();
//          this.Ic_Amp_q = generateSubstitution();
//          this.Ineut_Amp_q = generateSubstitution();
//          this.Ua_Vol_q = generateSubstitution();
//          this.Ub_Vol_q = generateSubstitution();
//          this.Uc_Vol_q = generateSubstitution();
//          this.Uneut_Vol_q = generateSubstitution();
//      } else{
          this.Ia_Amp_q = 0;
          this.Ib_Amp_q = 0;
          this.Ic_Amp_q = 0;
          this.Ineut_Amp_q = 0;
          this.Ua_Vol_q = 0;
          this.Ub_Vol_q = 0;
          this.Uc_Vol_q = 0;
          this.Uneut_Vol_q = 0;
//      }



  }

  public byte[] convertToBytes() {
```

```
byte[] result = new byte[93];
int offset = 0;

result[offset] = SequenceASDU_tag;
offset += 1;

result[offset] = SequenceASDU_lgt;
offset += 1;

result[offset] = svID_tag;
offset += 1;

result[offset] = svID_lgt;
offset += 1;

System.arraycopy(convertingToByte(svID), 0, result, offset, 10);
offset += 10;

result[offset] = smpCnt_tag;
offset += 1;

result[offset] = smpCnt_lgt;
offset += 1;

System.arraycopy(convertingToByte(smpCnt), 0, result, offset, 2);
offset += 2;

result[offset] = confRev_tag;
offset += 1;

result[offset] = confRev_lgt;
offset += 1;

System.arraycopy(convertingToByte(confRev), 0, result, offset, 4);
offset += 4;

result[offset] = smpSynch_tag;
offset += 1;

result[offset] = smpSynch_lgt;
offset += 1;

result[offset] = smpSynch;
offset += 1;

result[offset] = SequenceofData_tag;
offset += 1;

result[offset] = SequnceofData_lgt;
offset += 1;
```

```
System.arraycopy(convertingToByte(Ia_Amp_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ia_Amp_q), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ib_Amp_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ib_Amp_q), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ic_Amp_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ic_Amp_q), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ineut_Amp_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ineut_Amp_q), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ua_Vol_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ua_Vol_q), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ub_Vol_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Ub_Vol_q), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Uc_Vol_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Uc_Vol_q), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Uneut_Vol_instMag), 0, result, offset, 4);
offset += 4;

System.arraycopy(convertingToByte(Uneut_Vol_q), 0, result, offset, 4);
offset += 4;

return result;
```

```
    }

    public static byte[] convertingToByte(String str) {
        return str.getBytes(StandardCharsets.UTF_8);
    }

    public static byte[] convertingToByte(short val) {
        byte[] res = new byte[2];
        res[1] = (byte) (val & 0xff);
        res[0] = (byte) ((val >> 8) & 0xff);
        return res;
    }

    public static byte[] convertingToByte(int val) {
        byte[] res = new byte[4];
        res[3] = (byte) (val & 0xff);
        res[2] = (byte) ((val >> 8) & 0xff);
        res[1] = (byte) ((val >> 16) & 0xff);
        res[0] = (byte) ((val >> 24) & 0xff);
        return res;
    }

    public static int generateSubstitution(){
        Random rand = new Random();
        int randomNumber = rand.nextInt(Short.MAX_VALUE); // генерирует рандомное число от 0 до
32767
        return randomNumber;
    }
}
```

SvParser

```
package org.example.pcapFiles;

import org.pcap4j.core.PcapPacket;
import org.example.packetStructureCapture.*;

import javax.xml.bind.DatatypeConverter;
import java.util.Optional;

import static org.reflections.Reflections.log;

public class SvParser {

    private static final int datasetSize = 64;

    public Optional<SvPacket> decode(PcapPacket packet){
        try {
            byte [] data = packet.getRawData();
```

```
        int length = data.length;

        SvPacket result = new SvPacket();

        result.setTimestamp(packet.getTimestamp().toString());

        System.out.println(packet.getTimestamp()); //packet capture time

        result.setMacDst(byteArrayToMac(data, 0));
        result.setMacSrs(byteArrayToMac(data, 6));
        result.setType(twoByteArrayToString(data, 12));

        result.setAppId(twoByteArrayToString(data, 14));
        result.setLength(byteArrayToInt(data, 16, 2));
        result.setReserved1(twoByteArrayToString(data, 18));
        result.setReserved2(twoByteArrayToString(data, 20));

        result.setApdu(byteToAPDU(data, 24)); //24 start of apdu




        return Optional.of(result);
    } catch (Exception e) {log.error("Cannot parse SV packet");}
    return Optional.empty();
}


public static String byteArrayToMac (byte[] b, int offset){
    return String.format("%02x:%02x:%02x:%02x:%02x:%02x",
            b[offset],
            b[offset + 1],
            b[offset + 2],
            b[offset + 3],
            b[offset + 4],
            b[offset + 5]
    );

}

public APDU byteToAPDU(byte[] b, int offset){
    APDU apdu = new APDU();
    int noAsdu = byteArrayToInt(b, offset + 2, 1);
    apdu.setNoASDU(noAsdu);

    for (int i = 0; i < noAsdu; i++) {
        apdu.getSeqASDU().add(byteToASDU(b, offset + 5 + i * 93)); // 29 bit = start of asdu
    }
```

```
    return apdu;
}

public ASDU byteToASDU(byte[] b, int offset){
    ASDU asdu = new ASDU();

    String hexId = byteArrayToString(b, offset + 4, 10);
    byte[] decId = DatatypeConverter.parseHexBinary(hexId); //from hex to decimal array
    String id = new String(decId); // from decimal array to text (ASCII)

    asdu.setSvID(id);
    asdu.setSmpCnt(byteArrayToInt(b, offset + 14 + 2, 2));
    asdu.setConfRev(byteArrayToInt(b, offset + 18 + 2, 4));
    asdu.setSmpSynch(byteArrayToInt(b, offset + 24 + 2, 1));  // 0 - None 1 - Local 2 - Remote
    asdu.setDataset(byteToPhsMeas(b, offset + 27 + 2));
    return asdu;
}

public PhsMeas byteToPhsMeas(byte[] b, int offset){
    PhsMeas dataset = new PhsMeas();

    dataset.setInstIa(byteArrayToInstVal(b, offset + 8 * 0));
    dataset.setQIa(byteArrayToQuality(b, offset + 4 + 8 * 0));

    dataset.setInstIb(byteArrayToInstVal(b, offset + 8 * 1));
    dataset.setQIb(byteArrayToQuality(b, offset + 4 + 8 * 1));

    dataset.setInstIc(byteArrayToInstVal(b, offset + 8 * 2));
    dataset.setQIc(byteArrayToQuality(b, offset + 4 + 8 * 2));

    dataset.setInstIn(byteArrayToInstVal(b, offset + 8 * 3));
    dataset.setQIn(byteArrayToQuality(b, offset + 4 + 8 * 3));

    dataset.setInstUa(byteArrayToInstVal(b, offset + 8 * 4));
    dataset.setQUa(byteArrayToQuality(b, offset + 4 + 8 * 4));

    dataset.setInstUb(byteArrayToInstVal(b, offset + 8 * 5));
    dataset.setQUb(byteArrayToQuality(b, offset + 4 + 8 * 5));

    dataset.setInstUc(byteArrayToInstVal(b, offset + 8 * 6));
    dataset.setQUc(byteArrayToQuality(b, offset + 4 + 8 * 6));

    dataset.setInstUn(byteArrayToInstVal(b, offset + 8 * 7));
    dataset.setQUn(byteArrayToQuality(b, offset + 4 + 8 * 7));


    return dataset;
}
```

```
private Quality byteArrayToQuality(byte[] b, int offset) {
   Quality quality = new Quality();

   int qbyte = byteArrayToInt(b, offset, 4);

   quality.setValidity((qbyte & 0b11));
   qbyte = qbyte >> 2;

   quality.setOverflow((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setOutOfRange((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setBadReference((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setOscillatory((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setFailure((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setOldData((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setInconsistent((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setInaccurate((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setSource((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setTest((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setOperatorBlocked((qbyte & 0b1));
   qbyte = qbyte >> 1;

   quality.setDerived((qbyte & 0b1));




   return quality;
}

public static String twoByteArrayToString(byte[] b, int offset){
```

```
        return String.format("0x%02x%02x",
                b[offset],
                b[offset + 1]
        );

    }

    public static String byteArrayToString(byte[] b, int offset, int len){
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < len; i++) {
            hexString.append(String.format("%02x", b[offset + i]));
        }
        String out = new String(hexString);
        return out;

    }

    private int byteArrayToInt(byte[] b, int offset, int len) {
        int value = b[offset + len - 1] & 0xFF;
        for (int i = 1; i < len; i++) {
            value = value | (b[offset + (len - i - 1)] & 0xFF) << 8 * i;
        }
        return value;
    }


    private int byteArrayToInstVal(byte[] b, int offset) {
        return b[offset + 3] & 0xff | (b[offset + 2] & 0xff) << 8 | (b[offset + 1] & 0xff) << 16 | (b[offset] & 0xff)
<< 24;
    }


}
```

SvPublisher

```
package org.example.pcapFiles;

import lombok.Getter;
import lombok.Setter;
import lombok.SneakyThrows;
import lombok.extern.slf4j.Slf4j;
import org.example.dataobjects.measurements.MV;
import org.example.packetStructureSend.APDU;
import org.example.packetStructureSend.ASDU;
import org.pcap4j.packet.EthernetPacket;
import org.pcap4j.packet.Packet;
import org.pcap4j.packet.UnknownPacket;
import org.pcap4j.packet.namednumber.EtherType;
```

```
import org.pcap4j.util.MacAddress;

import java.io.UnsupportedEncodingException;
import java.util.Random;

@Slf4j
public class SvPublisher extends SvSubscriber {

   public MV phsAInst = new MV();
   public MV phsBInst = new MV();
   public MV phsCInst = new MV();
   public MV phsNuetInst = new MV();

   public MV phsAUnst = new MV();
   public MV phsBUnst = new MV();
   public MV phsCUnst = new MV();
   public MV phsNuetUnst = new MV();

   @Getter @Setter
   private String srcMAC;
   @Getter @Setter
   private String dstMAC;
   private short count;

   String fakeMAC = "12:34:56:78:9a:bc";
   int test = 0;

   Random random = new Random();


   public SvPublisher() {
      this.phsAInst.getInstMag().getF().setValue(0d);
      this.phsBInst.getInstMag().getF().setValue(0d);
      this.phsCInst.getInstMag().getF().setValue(0d);
      this.phsNuetInst.getInstMag().getF().setValue(0d);
      this.phsAUnst.getInstMag().getF().setValue(0d);
      this.phsBUnst.getInstMag().getF().setValue(0d);
      this.phsCUnst.getInstMag().getF().setValue(0d);
      this.phsNuetUnst.getInstMag().getF().setValue(0d);
   }

   @SneakyThrows @Override
   public void process() {

      APDU apdu = new APDU();
//      PcapHandle sendHandle = .openLive(SNAPLEN,
PcapNetworkInterface.PromiscuousMode.PROMISCUOUS, READ_TIMEOUT);
      ASDU asdu = new ASDU("TESTMU0101", count, compileValues());
      byte[] payload = new byte[apdu.convertToBytes().length + asdu.convertToBytes().length];
```

```
        System.arraycopy(apdu.convertToBytes(), 0, payload, 0, apdu.convertToBytes().length);
        System.arraycopy(asdu.convertToBytes(), 0, payload, apdu.convertToBytes().length,
asdu.convertToBytes().length);

        UnknownPacket.Builder svBuilder = new UnknownPacket.Builder();
        svBuilder.rawData(payload);


        EthernetPacket.Builder etherBuilder = new EthernetPacket.Builder();
//      if ((count > 1500) && (count <3000)){
//         etherBuilder
//               .srcAddr(MacAddress.getByName(srcMAC))
//               .dstAddr(MacAddress.getByName(fakeMAC))
//               .type(new EtherType((short) 0x88ba, "SV_IEC61850"))
//               .payloadBuilder(svBuilder)
//               .paddingAtBuild(true);
//      }else{
         etherBuilder
               .srcAddr(MacAddress.getByName(srcMAC))
               .dstAddr(MacAddress.getByName(dstMAC))
               .type(new EtherType((short) 0x88ba, "SV_IEC61850"))
               .payloadBuilder(svBuilder)
               .paddingAtBuild(true);
//      }

        Packet p = etherBuilder.build();

        this.getHandle().sendPacket(p);
//      log.info("Packet sent " + test);
        test+=1;

        if (count <3999){
           count += 1;
        }else {
           count = 0;
        }
//      if ((count > 1111) && (count <3214)){
//         Thread.sleep(random.nextLong(10) + 1);
//      }else{

           Thread.sleep((long) 0.25);
//      }

        if (test == 7199){
           System.out.println(String.format("packets sent: %d", test));
           stop();
           Thread.interrupted();
        }
```

```
    }
    /*Метод фундаментально неверно работает - переводит из hex в decimal*/

//    public static byte[] convertToByte(String stringLine) throws UnsupportedEncodingException {
//        int offset = 0;
//
//        byte[] result = new byte[stringLine.length() / 2];
//        for (int i = 0; i < result.length; i++) {
//            String oneByte = stringLine.substring(0+offset, 2+offset);
//            Integer hex = Integer.parseInt(oneByte, 16);
//            result[i] = hex.byteValue();
//            offset+=2;
//        }
//
////        StringBuilder sb = new StringBuilder();
////        for (byte b : bytes) {
////            sb.append(String.format("%02X ", b));
////        }
////        result = sb.toString());
////        }
////        return result;
//        return result;
//    }

    public static byte[] convertMACtoByte(String macString) throws UnsupportedEncodingException {
        String[] macArray = macString.split(":");
        byte[] result = new byte[6];

        for (int i = 0; i < 6; i++) {
            Integer hex = Integer.parseInt(macArray[i], 16);
            result[i] = hex.byteValue();
        }
        return result;
    }

    public int[] compileValues(){
        int[] result = new int[8];
        result[0] = phsAInst.getInstMag().getF().getValue().intValue();
        result[1] = phsBInst.getInstMag().getF().getValue().intValue();
        result[2] = phsCInst.getInstMag().getF().getValue().intValue();
        result[3] = phsNuetInst.getInstMag().getF().getValue().intValue();

        result[4] = phsAUnst.getInstMag().getF().getValue().intValue();
        result[5] = phsBUnst.getInstMag().getF().getValue().intValue();
        result[6] = phsCUnst.getInstMag().getF().getValue().intValue();
        result[7] = phsNuetUnst.getInstMag().getF().getValue().intValue();
        return result;
    }
}
```

SvSubscriber

```
package org.example.pcapFiles;

import com.opencsv.CSVWriter;
import lombok.Getter;
import lombok.Setter;
import lombok.SneakyThrows;
import lombok.extern.slf4j.Slf4j;
import org.example.logiclanodes.common.LN;
import org.example.logiclanodes.measurements.MMXU;
import org.example.packetStructureCapture.SvPacket;
import org.pcap4j.core.*;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

@Slf4j
public class SvSubscriber extends LN {
    /**
     * check this.setNickName("VMware Virtual Ethernet Adapter for VMnet8");
     */

    @Getter @Setter
    private String nickName;
    @Getter @Setter
    private ArrayList<String> nicArray = new ArrayList<>();
    @Getter @Setter
    private ArrayList<String> nicMAC = new ArrayList<>();
    @Getter @Setter
    private ArrayList<String> nicIP = new ArrayList<>();
    @Getter @Setter
    private PcapHandle handle;
    @Getter @Setter
    private static final int BUFFER_SIZE = 1000;
    @Getter @Setter
    private static LinkedBlockingQueue<String> csvBuffer = new LinkedBlockingQueue<>(BUFFER_SIZE);
    HashMap<Integer, SvPacket> svPacketMap = new LinkedHashMap<>();
//    int lastVal = 0;

    MMXU mmxu = new MMXU();
```

```
    final List <PacketListener> listeners = new CopyOnWriteArrayList<>();

    private PacketListener defaultPacketListener = packet -> {
      listeners.forEach(listener -> listener.gotPacket(packet));
    };

    public void checkNic(){
      try {
        for (PcapNetworkInterface nic : Pcaps.findAllDevs()) {
//            System.out.println(nic.getAddresses());
//            System.out.println(nic.getLinkLayerAddresses());
          nicMAC.add(String.valueOf(nic.getLinkLayerAddresses().get(0)));
          nicArray.add(nic.getDescription());
          nicIP.add(parseIP(String.valueOf(nic.getAddresses())));
        }
        System.out.println("NIC written down");
      } catch (PcapNativeException e) {
        throw new RuntimeException(e);
      }
    }

    public String parseIP(String ipString){
      String IPADDRESS_PATTERN =
          "(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)";

      Pattern pattern = Pattern.compile(IPADDRESS_PATTERN);
      Matcher matcher = pattern.matcher(ipString);
      if (matcher.find()) {
        return matcher.group();
      }
      else{
        return "0.0.0.0";
      }
    }

    @SneakyThrows
    public void start(){
      if (nickName != null && !nickName.isEmpty()){
        initializeNetworkInterface();

          if (handle != null) {
            String filter = "ether proto 0x88ba";  // proto 0x88ba = IEC SV

            handle.setFilter(filter, BpfProgram.BpfCompileMode.OPTIMIZE);

            Thread captureThread = new Thread(() -> {
              try {
                log.info("Starting Packet Capturing");
                handle.loop(0, defaultPacketListener); //packetCount should be x2. 0 for unlimited
capture
```

```
                } catch (PcapNativeException e) {
                    log.info("Finished Packet Capturing");
                } catch (InterruptedException e) {
                    log.info("Finished Packet Capturing");
                } catch (NotOpenException e) {
                    log.info("Finished Packet Capturing");
                }


            });

            captureThread.start();
        }
    } else log.error("Select NIC first");
  }

  @SneakyThrows
  public void stop(){
    if (nickName != null && !nickName.isEmpty()) {
      if (handle != null) {
        handle.breakLoop();
//          handle.close();
        log.info("Packet Capturing Stopped");
      }else {
        log.error("Packet Capturing ALREADY Stopped");
      }
    } else log.error("Select NIC first");
  }

  @SneakyThrows
  public void initializeNetworkInterface() {
    Optional<PcapNetworkInterface> nic = Pcaps.findAllDevs().stream()
        .filter(i -> nickName.equals(i.getDescription()))
        .findFirst();

    if (nic.isPresent()) {
      handle = nic.get().openLive(1500, PcapNetworkInterface.PromiscuousMode.PROMISCUOUS, 10);
      log.info("Network handler: {}", nic);
    }
    else {
      log.error("Network Interface not found");
    }
  }

  public void addListener(PacketListener listener) {
    listeners.add(listener);
  }

  @SneakyThrows
  @Override
```

```
    public void process() {
//      log.debug("EtListener method");

        File resultCSV = new File("src/main/resources/TestRun.csv");
        FileWriter fileWriter = new FileWriter(resultCSV);
        CSVWriter writer = new CSVWriter(fileWriter);

        writer.writeNext(new String[]{
                "time",
                "source",
                "destination",
                "svID",
                "smpCnt",
                "Ia",
                "Ia_quality",
                "Ib",
                "Ib_quality",
                "Ic",
                "Ic_quality",
                "Ua",
                "Ua_quality",
                "Ub",
                "Ub_quality",
                "Uc",
                "Uc_quality",
        });




        this.checkNic();
        this.setNickName("VMware Virtual Ethernet Adapter for VMnet8");
        this.getNicArray();


        SvParser svParser = new SvParser();

        AtomicInteger curCnt = new AtomicInteger();

//      HashMap<String, ArrayList<SvPacket>> sourceMap = new HashMap<>();

//      ArrayList<Optional> array = new ArrayList<>();

        this.addListener(packet -> {
            Optional<SvPacket> svPacket = svParser.decode(packet);
            int noASDU = svPacket.get().getApdu().getNoASDU();
            for (int i = 0; i < noASDU; i++) {


                if (svPacket.isPresent()
```

```
                && curCnt.get() != svPacket.get().getApdu().getSeqASDU().get(i).getSmpCnt()
//                  && lastVal != svPacket.get().getApdu().getSeqASDU().get(i).getDataset().getInstIa()
            ) {


                svPacketMap.put(svPacketMap.size(), svPacket.get());
                mmxu.process(svPacket.get().getApdu().getSeqASDU().get(0).getDataset());
                System.out.println(svPacketMap.size());

//                if (svPacketMap.size() == 7199){
//                    try {
//                        Thread.sleep(1000);
//                    } catch (InterruptedException e) {
//                        throw new RuntimeException(e);
//                    }
//                    System.out.println(String.format("Captured packets: %d", svPacketMap.size()));
//                    for (int j = 0; j < svPacketMap.size(); j++) {
//
//                        writer.writeNext(new String[]{
//                                String.valueOf(svPacketMap.get(j).getTimestamp()),
//                                String.valueOf(svPacketMap.get(j).getMacSrs()),
//                                String.valueOf(svPacketMap.get(j).getMacDst()),
//                                String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSvID()),
//                                String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSmpCnt()),
//                                String.valueOf(mmxu.IphsA.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIa().toString()),
//                                String.valueOf(mmxu.IphsB.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIb().toString()),
//                                String.valueOf(mmxu.IphsC.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIc().toString()),
//                                String.valueOf(mmxu.UphsA.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUa().toString()),
//                                String.valueOf(mmxu.UphsB.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUb().toString()),
//                                String.valueOf(mmxu.UphsC.get(j)[0]),
//
String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUc().toString())
//                        });
//                    }
//                    try {
//                        writer.close();
//                        stop();
//                        Thread.interrupted();
//
//                    } catch (IOException e) {
```

```
//                    throw new RuntimeException(e);
//                }
//            }

            if (svPacketMap.size() == 7199){
                System.out.println("DONE");
                for (int j = 0; j < svPacketMap.size(); j++) {

                    writer.writeNext(new String[]{
                            String.valueOf(svPacketMap.get(j).getTimestamp()),
                            String.valueOf(svPacketMap.get(j).getMacSrs()),
                            String.valueOf(svPacketMap.get(j).getMacDst()),
                            String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSvID()),
                            String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getSmpCnt()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstIa()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIa().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstIb()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIb().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstIc()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQIc().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstUa()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUa().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstUb()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUb().toString()),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getInstUc()/1000d),

String.valueOf(svPacketMap.get(j).getApdu().getSeqASDU().get(0).getDataset().getQUc().toString())
                    });
                }
                try {
                    writer.close();
                    stop();
                    Thread.interrupted();

                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
```

```
//                if (setSvPckt.size() >= 37000) {
//                    System.out.println(setSvPckt);
//                }

                curCnt.set(svPacket.get().getApdu().getSeqASDU().get(i).getSmpCnt());  //update counter else writes packet twice
            }

        }

    });

    this.start();
  }



  public static synchronized void addToBuffer(String data) {
    try {
      csvBuffer.put(data);
    } catch (InterruptedException e) {
      System.err.println("Error adding data to buffer: " + e.getMessage());
    }
  }

  public static void handlePacket(Optional<SvPacket> svPacket) {
    // Extract the relevant information from the packet

    // Format the data as a CSV string


    // Add the data to the buffer
//    addToBuffer(csvData);
  }

//  public static void startWriterThread(String fileName) {
//    Thread writerThread = new Thread(() -> {
//      try (FileWriter fileWriter = new FileWriter(fileName, true)) {
//        while (true) {
//          String data = csvBuffer.take();
//          fileWriter.append(data);
//          fileWriter.append("\n");
//          fileWriter.flush();
//        }
//      } catch (IOException | InterruptedException e) {
//        System.err.println("Error writing data to CSV file: " + e.getMessage());
//      }
//    });
//    writerThread.start();
```

```
//   }


}
```

ComtradeParser

```
package org.example;

import lombok.SneakyThrows;
import org.example.logiclanodes.common.LN;
import org.example.logiclanodes.input.LCOM;
import org.example.pcapFiles.SvSubscriber;
import org.example.pcapFiles.SvPublisher;

import java.util.*;

public class ComtradeParser {
   /**
    * Основной класс программы, по вызову метода CreateCSV()
    * происходит цикличное
    * 1)чтение строки данных .dat файла объектом класса LCOM
    * 2)отправка SV пакета по данным LCOM (замер 3 токов и 3 напряжений в один момент
времени) объектом класса SvPublisher
    * 3)захват SV пакета объектом класса SvSubscriber и запись данных в файл группами по 7200
пакетов
    * **/
   private static List<LN> inList = new ArrayList<>();


   @SneakyThrows
   public void CreateCSV(){


      Thread.sleep(20000); // задержка для выполнения модели ПСКАД

      // создание и запуск SV подписчика
      SvSubscriber SvSubscriber = new SvSubscriber();
      SvSubscriber.process();


      // создание узла для чтения и
      LCOM lcom = new LCOM();
      lcom.setFilePath(

"E:\\DZ\\11sem\\AI_Enregy\\KP\\pythonProject\\PSCAD_files\\testGrid.gf42\\Rank_00001\\Run_0000
1\\",
            "ABC_W1");
      inList.add(lcom);
```

```
        SvPublisher svPublisher = new SvPublisher();
        svPublisher.checkNic();
//      System.out.println(EtListen.getNicArray());


        System.out.println("Pick a Src");
        System.out.println(svPublisher.getNicArray());

        int nicSRC = 5;
        String srcMAC = svPublisher.getNicMAC().get(nicSRC);
        System.out.println(
            "Source" +
                "\nName: " + svPublisher.getNicArray().get(nicSRC) +
                "\nIP: " + svPublisher.getNicIP().get(nicSRC) +
                "\nMAC: " + srcMAC
        );


        System.out.println("Pick a Dst");
        System.out.println(svPublisher.getNicArray());

        int nicDST = 6;
        String dstMAC = svPublisher.getNicMAC().get(nicDST);
        System.out.println(
            "Destination" +
            "\nName: " + svPublisher.getNicArray().get(nicDST) +
            "\nIP: " + svPublisher.getNicIP().get(nicDST) +
            "\nMAC: " + dstMAC
            );

        String broadMAC = "ff:ff:ff:ff:ff:ff";


        svPublisher.setNickName(svPublisher.getNicArray().get(nicSRC));
        svPublisher.initializeNetworkInterface();

        svPublisher.setSrcMAC(srcMAC);
        svPublisher.setDstMAC(broadMAC);


        inList.add(svPublisher);

        svPublisher.phsAInst = lcom.OUT.get(0);
        svPublisher.phsBInst = lcom.OUT.get(1);
        svPublisher.phsCInst = lcom.OUT.get(2);

        svPublisher.phsAUnst = lcom.OUT.get(3);
        svPublisher.phsBUnst = lcom.OUT.get(4);
```

```
        svPublisher.phsCUnst = lcom.OUT.get(5);


        while (lcom.hasNextData()) {
            inList.forEach(LN::process);
        }
        System.out.println("end");
    }
}
```

MU

```
package org.example;

import java.nio.file.*;


public class MU {
    public static void main(String[] args) throws Exception {
        // Получаем объект WatchService
        WatchService watcher = FileSystems.getDefault().newWatchService();

        boolean isProcessingEvent = false;

        ComtradeParser comParser = new ComtradeParser();

        // Регистрируем изменения в директории
        Path dir =
Paths.get("E:\\DZ\\11sem\\AI_Enregy\\KP\\pythonProject\\PSCAD_files\\testGrid.gf42\\Rank_00001\\
Run_00001");
        dir.register(watcher, StandardWatchEventKinds.ENTRY_MODIFY);

        // Бесконечный цикл для мониторинга событий
        while (true) {
            WatchKey key;
            try {
                // Ожидаем получение событий
                key = watcher.take();
//                Thread.sleep( 50 );
            } catch (InterruptedException ex) {
                return;
            }

//            Thread.sleep( 50 );
            // Обрабатываем события
            for (WatchEvent<?> event : key.pollEvents()) {
                WatchEvent.Kind<?> kind = event.kind();

                Thread.sleep( 150 );
```

```
            // Если произошло изменение файла
            if (kind == StandardWatchEventKinds.ENTRY_MODIFY) {
                @SuppressWarnings("unchecked")
                WatchEvent<Path> ev = (WatchEvent<Path>) event;
                Path filename = ev.context();

                // Проверяем, что изменение произошло именно в нужном файле
                if (filename.toString().equals("ABC_W1.dat")) {
                    isProcessingEvent = true;
                    System.out.println("File was modified: " + filename);
                    comParser.CreateCSV();
                    isProcessingEvent = false;


                }
            }
        }

        // Сбрасываем ключ, чтобы можно было продолжать получать события
        boolean valid = key.reset();
        if (!valid) {
            break;
        }
    }
  }
}
```

Приложение Б

КОД ОСНОВНОЙ ПРОГРАММЫ

```
import numpy as np
import pandas as pd
from keras import Sequential
from keras.src.layers import LSTM, Dense
from keras.src.regularizers import regularizers
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import accuracy_score, f1_score, mean_squared_error, r2_score, roc_auc_score



# print(file_name, path)
def generate_dataframe(path_csv, name):

    print('\nForming dataframe [time, interval]')
    data = pd.read_csv(path_csv, encoding ="ISO-8859-1")
    print(data.info())


    data['time'] = pd.to_datetime(data['time'])

    print(data)

    time = pd.DataFrame(data, columns=['time'])

    print(time)

    time['interval'] = time['time'].diff()

    time.loc[0, 'interval'] = time.loc[1, 'interval']

    time['interval'] = time['interval'].dt.total_seconds() * 1000

    print(time)
    time.to_csv(name, index=False)

    return time


def prepare_data(data):
    windows = data['interval'].rolling(window=5)
    print(windows)

    mean_values = windows.mean()
    std_values = windows.std()
```

```
    # вычисляем среднее значение и стандартное отклонение для каждого окна
    data['mean_interval'] = mean_values
    data['std_interval'] = std_values
    data.dropna(inplace=True)

    print(data)

    mean_threshold = 0.5

    # определяем метки для каждого окна данных
    data['label'] = np.where((data['mean_interval'] <= mean_threshold),
                             0, 1)

    # print(data.head())

    # определяем размер окна
    window_size = 32

    num_features = 1

    # разбиваем DataFrame на окна фиксированного размера
    X = []
    y = []
    time = []

    # проходимся по всем измерениям, кроме последних window_size
    for i in range(len(data) - window_size):
        # добавляем временную метку
        time.append(data.iloc[i]['time'])

        # добавляем входные данные
        X.append(data.iloc[i:i + window_size]['interval'].values.reshape(window_size, num_features))

        # добавляем выходные данные
        y.append(data.iloc[i + window_size - 1]['label'])

    # преобразуем списки в массивы NumPy
    X_done = np.array(X)
    y_done = np.array(y)
    time_done = np.array(time)
    return X_done, y_done, time_done, window_size, num_features

path_normal =
"E:/DZ/11sem/AI_Enregy/KP/pythonProject/SVcreateAndParse/src/main/resources/TestRun(ALLDATA_
NOFAULT_7200_80perPeriod).csv"
path_switched =
"E:/DZ/11sem/AI_Enregy/KP/pythonProject/SVcreateAndParse/src/main/resources/TestRun(randInt).cs
v"
data_clean = generate_dataframe(path_normal, "normal")
```

```
data_messed = generate_dataframe(path_switched, "fault")

print("done")

interval_normal = np.array(data_clean['interval'].values)
interval_messed = np.array(data_messed['interval'].values)

# Создаем график исходных данных
plt.plot(np.arange(len(interval_normal)), interval_normal)
plt.title("Интеравал между получением SV пакетами", fontsize=10)
plt.ylabel('Интервал, ms', fontsize=8)
plt.xlabel('№ packet ', fontsize=8)
plt.show()

plt.plot(np.arange(len(interval_messed)), interval_messed)
plt.title("Интеравал между получением SV пакетами с подменой", fontsize=10)
plt.ylabel('Интервал, ms', fontsize=8)
plt.xlabel('№ packet ', fontsize=8)
plt.show()

# препподготовка данных
X_messed, y_messed, time_messed, window_size_messed, num_features_messed =
prepare_data(data_messed)
X_clean, y_clean, time_clean, window_size_clean, num_features_clean = prepare_data(data_clean)



# определяем архитектуру сети
model = Sequential()
model.add(LSTM(32, input_shape=(window_size_messed, num_features_messed),
kernel_regularizer=regularizers.L2(0.01), dropout = 0.2) )
# model.add(LSTM(50, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy', 'MeanSquaredError'])

model.summary
# разбиваем данные на обучающую и тестовую выборки
train_size = int(0.8 * len(X_messed))
train_X = X_messed[:train_size]
train_y = y_messed[:train_size]
test_X = X_messed[train_size:]
test_y = y_messed[train_size:]

epochs = 100
epoch_range = np.arange(1,epochs+1,1)
# обучаем модель
history = model.fit(train_X, train_y, epochs=epochs, batch_size=50, validation_data=(test_X, test_y))
```

```
accuracy_list = []
MSE_list = []
accuracy_list.append(history.history['accuracy'])
MSE_list.append(history.history['MeanSquaredError'])


print("_____ТОЧНОСТЬ_____", accuracy_list)

# оцениваем модель на тестовой выборке
# loss, accuracy = model.evaluate(test_X, test_y)
# print('Точность на тестовой выборке:', accuracy, loss)

# предсказываем значения для тестовых данных
y_pred = model.predict(test_X)

#точность
plt.plot(epoch_range, accuracy_list[0])
plt.title("Accuracy", fontsize=10)
plt.ylabel('Accuracy rate', fontsize=8)
plt.xlabel('epoch', fontsize=8)
plt.show()

#mse
plt.plot(epoch_range, MSE_list[0])
plt.title("MSE", fontsize=10)
plt.ylabel('rate', fontsize=8)
plt.xlabel('epoch', fontsize=8)
plt.show()


# Создаем график
plt.plot(np.arange(len(y_pred)), y_pred)
plt.title("Проверка модели на тестовой выборке (0-в норме, 1-обнаружена подмена)", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)
plt.show()

plt.subplot(1, 2, 1)
plt.plot(np.arange(len(y_pred)), y_pred)
plt.title("predict", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)

plt.subplot(1, 2, 2)
plt.plot(np.arange(len(test_y)), test_y)
plt.title("test", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)

plt.show()
```

```
# X_test_scaled = scaler.transform(test_X[:, :-1].reshape(-1, test_X.shape[-1])).reshape(test_X.shape)
# y_pred = model.predict(X_test_scaled)
y_pred_binary = [1 if y >= 0.5 else 0 for y in y_pred]

# вычисляем метрики качества
accuracy = accuracy_score(test_y, y_pred_binary)
f1 = f1_score(test_y, y_pred_binary)
rmse = mean_squared_error(test_y, y_pred, squared=False)
r2 = r2_score(test_y, y_pred)
# auc = roc_auc_score(test_y, y_pred)


# сравниваем предсказанные значения с фактическими значениями
accuracy = np.mean(y_pred_binary == test_y)

# print('Точность модели на тестовых данных:', accuracy, loss)
print('Точность:', accuracy)
print('F1-мера:', f1)
print('RMSE:', rmse)

plt.plot(np.arange(len(y_pred_binary)), y_pred_binary)
plt.title("Проверка модели на тестовой выборке (0-в норме, 1-обнаружена подмена)", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)
plt.show()

train_size = int(0.8 * len(X_messed))
train_X_new = X_clean[:train_size]
train_y_new = y_clean[:train_size]
test_X_new = X_clean[train_size:]
test_y_new = y_clean[train_size:]

print("s")

y_pred_new = model.predict(test_X_new)


plt.plot(np.arange(len(y_pred_new)), y_pred_new)
plt.title("Проверка модели на новой выборке (0-в норме, 1-обнаружена подмена)", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)
plt.show()




# X_test_scaled = scaler.transform(test_X[:, :-1].reshape(-1, test_X.shape[-1])).reshape(test_X.shape)
# y_pred = model.predict(X_test_scaled)

# преобразуем предсказанные значения в бинарные метки
y_pred_new_binary = [1 if y >= 0.5 else 0 for y in y_pred_new]
```

```
plt.plot(np.arange(len(y_pred_new_binary)), y_pred_new_binary)
plt.title("Проверка модели на новой выборке (0-в норме, 1-обнаружена подмена)", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)
plt.show()

# предсказываем значения для тестовых данных
y_pred_train = model.predict(train_X)



# # Создаем график
# plt.plot(np.arange(len(y_pred)), y_pred)
# plt.title("Проверка модели на тестовой выборке (0-в норме, 1-обнаружена подмена)", fontsize=10)
# plt.ylabel('Предсказанное значение', fontsize=8)
# plt.show()

plt.subplot(1, 2, 1)
plt.plot(np.arange(len(y_pred_train)), y_pred_train)
plt.title("predict_on_train", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)

plt.subplot(1, 2, 2)
plt.plot(np.arange(len(train_y)), train_y)
plt.title("train", fontsize=10)
plt.ylabel('Предсказанное значение', fontsize=8)

plt.show()
```

Приложение В

МОДЕЛЬ СЕТИ, ВЫПОЛНЕННАЯ В PSCAD