БЕЗОПАСНОСТЬ ЦИФРОВЫХ ТЕХНОЛОГИЙ. – 2024. – № 3 (114). – 34–52

МЕТОДЫ И СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.8 DOI: 10.17212/2782-2230-2024-3-34-52

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ПОСТРОЕНИЯ СИСТЕМ ОБРАБОТКИ ЗАШИФРОВАННЫХ ДАННЫХ И ИХ СРАВНЕНИЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ МАШИННОГО ОБУЧЕНИЯ*

ЛАПИНА М.А.[1], АРДЕЕВ Д.Ю.[2], ЛАПИН В.Г.[3]

[1] *355017, РФ, г. Ставрополь, ул. Пушкина, 1, ФГАОУ ВО «Северо-Кавказский федеральный университет», доцент кафедры информационной безопасности автоматизированных систем. E-mail: mlapina@ncfu.ru*
[2] *355017, РФ, г. Ставрополь, ул. Пушкина, 1, ФГАОУ ВО «Северо-Кавказский федеральный университет», ассистент кафедры информационная безопасность автоматизированных систем. E-mail: ardeev.dima345@gmail.com*
[3] *355017, РФ, г. Ставрополь, ул. Пушкина, 1, ФГАОУ ВО «Северо-Кавказский федеральный университет», кандидат физико-математических наук, доцент кафедры вычислительной математики и кибернетики. E-mail: vitlx@yandex.ru*

В статье рассмотрены два метода построения систем обработки зашифрованных данных, основанные на гомоморфном шифровании и разделительных вычислениях. Каждый метод отличается представленным алгоритмом и моделью обработки данных. Рассмотрены базовые операции над зашифрованными данными, а именно умножение и сложение. Для исследования использовались следующие библиотеки: TenSEAL, TensorFlow и PyTorch. Библиотека TenSEAL является инструментом полного гомоморфного шифрования, созданного для языка программирования C++, но адаптированного под использующийся в исследовании язык программирования Python. Эта библиотека позволяет использовать метод полного гомоморфного шифрования в построении вычислительной модели, задачей которой будет обработка зашифрованных данных. Для реализации метода разделительных вычислений, более известного как multy-party computation, будет использоваться библиотека TensorFlow, позволяющая создавать несколько тензоров и одновременно обучать их, что, в свою очередь, дает возможность реализовать принцип разделительных вычислений.

Ключевые слова: шифрование, конфиденциальность, гомоморфное шифрование, разделительные вычисления, тензоры, шифротексты

* Статья получена 13 августа 2024 г.

ВВЕДЕНИЕ

В современном мире информационные технологии стали неотъемлемой частью человеческой жизни. Причиной этого является технологический скачок в области компьютерных технологий, а также распространение широкополосных информационных систем, таких как интернет. Этот процесс способствует широкой автоматизации объектов социальной и критической инфраструктуры современного государства, а также связан с увеличением объема потока данных, необходимого для их корректной работы. Из этого следует, что информация – крайне важный ресурс, требующий обеспечения безопасности и конфиденциальности данных.

В настоящее временя актуальны два вида методов построения систем обработки зашифрованных данных – это гомоморфное шифрование и разделительные вычисления. Построение моделей систем для работы с зашифрованными данными проведем с использованием соответствующих библиотек языка программирования Python.

1. ОПИСАНИЕ РАБОТЫ

1.1. МЕТОДЫ ОБРАБОТКИ ЗАШИФРОВАННЫХ ДАННЫХ

Гомоморфное шифрование [3] – это метод шифрования данных, позволяющий проводить различные математические операции, не выполняя дешифрования информации, тем самым сохраняя конфиденциальность сведений. Для верного выполнения гомоморфного шифрования данных должны быть выполнены два условия: условие корректности и условие конфиденциальности. Условие корректности – это условие, при котором результат математических вычислений над исходными данными возможно получить, если будет проведена замена математической операции на последовательный алгоритм, проводящий вычислительные операции над зашифрованными или расшифрованными данными. Условие конфиденциальности – это условие, при котором операция, проводимая над данными, и промежуточные результаты не должны быть расшифрованы и не должны предоставлять дополнительные сведения, которые могли бы способствовать преждевременному дешифрованию информации, а следовательно, и потери данных. Этот метод позволяет проводить над зашифрованными переменными как алгебраические операции, так и булевы. Исходя из этого имеются следующие типы гомоморфного шифрования.

Частичное гомоморфное шифрование (англ. Somewhat homomorphic encryption, SHE) – метод, поддерживающий операции одного конкретного типа над зашифрованными данными.

Полностью гомоморфное шифрование (англ. Full Homomorphic Encryption, FHE) – метод, поддерживающий операции различных типов над зашифрованными данными.

Для построения систем обработки зашифрованных данных будет использоваться метод полного гомоморфного шифрования FHE. Он имеет ряд определенных достоинств:

- гибкость вычислений – поддержка выполнения любых вычислительных операций над зашифрованными данными;
- минимизация потери конфиденциальности данных. Поскольку все операции происходят в зашифрованном виде, открытость вычислительных узлов минимизируется. Даже при утечке данные остаются скрытыми от мошенников из-за шифрования;
- обеспечение приватности в облачных вычислениях. Подходит для выполнения вычислений на удаленных серверах либо в закрытых компьютерных сетях, минимизирует риски потери данных.

Недостатки данного метода:

- высокие вычислительные затраты. Операции умножения и сложения зашифрованных элементов зачастую проходят намного дольше в сравнении с незашифрованными. Также на производительность влияет непосредственно объем данных (чем он больше, тем медленнее проходит операция);
- большие размеры шифротекстов. Шифрованные тексты, созданные с применением гомоморфного шифрования, по размеру превышают исходные данные;
- ограниченное количество операций. Несмотря на то что полное гомоморфное шифрование FHE поддерживает вычисления при помощи любых математических операций, подавляющее большинство задач решается ограниченным числом операций (например, сложением). Это свойство накладывает ограничения для реализации многоуровневых вычислительных задач, требующих разделения на отдельные подоперации.

Эти недостатки делают метод гомоморфного шифрования сложным для широкого практического применения в области вычислений над зашифрованными данными.

Безопасные разделительные вычисления (англ. Secure Multy-Party Computation, MPC) – это криптографический метод, обеспечивающий возможность выполнения вычислений одновременно несколькими операторами без необходимости раскрытия зашифрованной информации друг другу. Каждая сторона имеет фрагменты сокрытых данных, при объединении которых можно получить результат без раскрытия конфиденциальной информации.

Главными аспектами МРС являются следующие понятия:

- конфиденциальность. Оператор не имеет никакой информации о входных данных других операторов, кроме той, которую он подает на ввод и может получить при завершении вычислений;
- корректность. Результат вычислительных операций должен быть корректным по отношению к тому результату, который возможно было бы получить без использования метода разделительных вычислений;
- надежность. Отказоустойчивость системы и способность купировать попытки перехвата данных злоумышленниками. Она должна обеспечить корректность вычислений и верность конечного результата, несмотря на возникшие проблемы;
- согласованность. Операторы одновременно получают результат для предотвращения несанкционированного использования конечного результата злоумышленником либо же недобросовестным оператором для исключения возможности утечки информации;
- анонимность. Сокрытие информации об операторах, участвующих в предоставлении данных для разделительных вычислений.

Перечисленные понятия можно отнести к преимуществам данного метода, однако у него также имеется и ряд недостатков, с которыми сталкиваются большинство разработчиков, применяющих разделительные вычисления в своих проектах.

Перечень основных недостатков:

- зависимость от операторов. Надежность МРС зависит от поведения операторов, участвующих в вычислениях. При любых отклонениях от нормы ресурсоемкость вычислений повышается, также сбой у одного из участников вычислений может нарушить весь процесс;
- ограниченная масштабируемость. Методы МРС трудномасштабируемы для большого количества операторов, участвующих в операциях, что увеличивает время работы программного кода, а также ресурсоемкость, негативно сказывается на скорости вычислений, а также повышает риски выхода из строя алгоритма, так как возрастает возможность системной неисправности, способной нарушить работоспособность алгоритма.

Несмотря на потенциал разделительных вычислений в области обеспечения безопасности и конфиденциальности вычислений, проводимых над зашифрованными данными, для реализации этого метода на практике необходимо учитывать всевозможные угрозы и ограничения, которые могут способствовать нарушению работоспособности системы, построенной при использовании метода.

1.2. ПЕРЕЧЕНЬ БИБЛИОТЕК

Для создания моделей машинного обучения, на основе которых будут применяться методы, используются две наиболее распространенные библиотеки – TensorFlow и PyTorch. Остальные библиотеки используют поверх указанных библиотек, тем самым обеспечивая оптимизацию ресурсов системы, на которой запускается модель. Рассмотрим каждую библиотеку в отдельности.

TensorFlow – это открытая библиотека, созданная для машинного обучения и адаптированная компанией Google. Она обеспечивает широкий функционал модели машинного обучения. Этот метод для проведения расчетов использует структуру статических графов, узлы которых представляют собой математические операции, а ребра – это многомерные массивы (тензоры). Такой метод позволяет проводить эффективные вычисления между устройствами электронной вычислительной машины, такими как центральный процессор и графический процессор. Также немаловажным отличием является ее гибкость. TensorFlow поддерживает различные уровни абстракции: от низкоуровневых API для сложных математических задач и до высокоуровневых API, таких как Keras, которые упрощают создание и обучение моделей.

Главным преимуществом библиотеки TensorFlow для такого исследования является его дополнительный функционал, предназначенный для работы с информационными моделями, работающими методами разделительных вычислений.

PyTorch – это библиотека, созданная для машинного обучения компанией Facebook и имеющая открытый код. Для вычислений использует динамический граф, позволяющий изменять структуру данных при выполнении обучения модели – это делает библиотеку удобной для работы с различными типами данных и различными объемами, однако это может сказаться негативно на качестве получаемых вычислений. Из преимуществ можно назвать открытый доступ к уже готовым и обученным моделям нейронных сетей, что позволяет сразу приступать к различным тестам и экспериментам над ними.

Из обеих представленных библиотек для построения моделей вычислительных систем наиболее подходящей для текущего исследования является TensorFlow. Эта библиотека предоставляет расширенный функционал создания информационной модели, а также имеет богатую экосистему для развертывания моделей.

TenSEAL – это библиотека, изначально написанная на языке C++, позже была адаптирована под развертку языка Python. Она специализируется на реализации алгоритмов гомоморфного шифрования. Главной особенностью биб-

лиотеки является то, что она работает с открытой библиотекой TensorFlow. Основные преимущества библиотеки:

- реализация функций гомоморфного шифрования на более быстром языке программирования, так как язык Python не является строго типизированным языком программирования;
- производительность. Библиотека TenSEAL имеет высокое быстродействие при работе над зашифрованными данными, поскольку основная часть функционала написана на языке C++, имеющем строгую типизацию данных и тем самым обеспечивающим быстрые вычисления;
- масштабируемость. Библиотека TenSEAL предназначена для работы с большими объемами данных, что позволяет использовать ее в областях, сопряженных с выполнением многоуровневых задач.

2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

2.1. РЕАЛИЗАЦИЯ МЕТОДОВ

Алгоритм работы с инструментами, необходимыми для проведения вычислительных операций над зашифрованными данными при помощи методов гомоморфного шифрования и разделительных вычислений, представлен ниже.

1. Для установки библиотек TenSEAL и TensorFlow необходимо запустить среду разработки для языка программирования Python.

2. В консоли среды разработки прописать команду **pip install** и указать названия библиотек.

3. После установки библиотеки импортировать модули библиотек непосредственно в код при помощи команды **from** (название библиотеки) **import** (название модулей).

После выполнения предыдущих шагов появляется возможность начать исследование операций над зашифрованными данными.

Для выполнения построения модели нейронной сети, использующей метод разделительных вычислений, как было описано выше, будем использовать библиотеку TensorFlow [1]. Эта библиотека является очень удобным инструментом и имеет самый широкий перечень функций в сравнении с аналогами.

В качестве операционной системы, применяемой в исследовании нейронной сети, была выбрана Windows 10, среда разработки – google colabrory [12],

язык программирования – Python [13], для обучения модели будет использоваться датасет MNIST на 1000 изображений.

Ниже приведен алгоритм построения системы, основанной на разделительных вычислениях, для работы с зашифрованными данными.

1. Производится установка основных библиотек, необходимых для построения модели вычислительной системы, через команду **pip** в области программирования.

2. Далее импортируем в код модули библиотек при помощи команды **import**.

3. Для обучения созданной модели необходимо загрузить данные из датасета MNIST, после чего полученные переменные преобразуются путем деления на 255, тем самым придут к диапазону от нуля до единицы. Эта операция продемонстрирована на рис. 1.

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Рис. 1. Преобразование данных

Как показано на рис. 2, данные разделяются на две подвыборки при помощи функции **train_test_split**, полученной из библиотеки **sklearn**.

```
x_train_1, x_train_2, y_train_1, y_train_2 = train_test_split(x_train, y_train, test_size=0.5, random_state=42)
x_test_1, x_test_2, y_test_1, y_test_2 = train_test_split(x_test, y_test, test_size=0.5, random_state=42)
```

Рис. 2. Разделение данных

Выполняется формирование базовой модели с помощью функции **create_model** и компиляция ее с использованием **tensorflow.keras.Sequential**, после чего компилируется с использованием оптимизатора Adam и функции потерь **categorical_crossentropy** (рис. 3).

На рис. 4 показано, как создается первая модель с помощью функции **create_model()** и обучается при помощи валидации **validation_data(x_test_1, y_test_1)**.

```
def create_model():
    model = Sequential([
        Flatten(input_shape=(28, 28)),
        Dense(512, activation='relu'),
        Dropout(0.2),
        Dense(10, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Рис. 3. Формирование базовой модели

```
model_1 = create_model()
history_1 = model_1.fit(x_train_1, y_train_1, epochs=5, validation_data=(x_test_1, y_test_1))
```

Рис. 4. Создание первой модели

Затем создается вторая модель по тому же принципу, что и предыдущая (рис. 5).

При помощи функции **ensemble_predictions** происходит объединение примерной точности обеих моделей.

```
def ensemble_predictions(models, data):
    predictions = [model.predict(data) for model in models]
    averaged_predictions = np.mean(predictions, axis=0)
    return np.argmax(averaged_predictions, axis=1)
```

Рис. 5. Объединение примерной точности моделей

После проведения всех вышеперечисленных операций производится создание списка моделей (рис. 6), куда добавляются предыдущие модели.

```
models = [model_1, model_2]
```

Рис. 6. Список моделей

Оценивается точность операции шифрования и расшифровывания на тестовом наборе данных.

Выполняется вывод данных формата: точность, скорость работы кода, затраты оперативной памяти для проведения вычислений методом разделительных вычислений.

В табл. 1 представлен полученный результат.

Таблица 1

Полученные данные в результате исследования модели MPC

Точность в зашифрованной модели	0.9792
Время работы	174.1779 с
Использовано ОЗУ	1863.18 Мб

Модель показывает высокую скорость работы алгоритма шифрования, построенного по принципу MPC. За счет распределения итоговой вычислительной нагрузки на две подмодели основной алгоритм достигает повышенной точности при минимально затраченном времени. Из этого можно сделать вывод, что метод разделительных вычислений – один из самых мощных вычислительный инструментов в области булевых и алгебраических операций над зашифрованными данными.

Исследование метода гомоморфного шифрования будет основываться на запатентованной готовой модели. Для реализации поставленной задачи будет изменен объем данных, поступающих на вход, что позволит выполнить более точный анализ.

Для выполнения построения модели, основанной на методе гомоморфного шифрования, будет использован тот же язык программирования, что и для прошлой модели, – Python. Среда программирования – google colab, операционная система персонального компьютера – Windows 10, для чистоты эксперимента в данной модели также будет использоваться датасет MNIST на 1000 изображений.

В ходе проведенного анализа методов гомоморфного шифрования для исследования был выбран метод полного гомоморфного шифрования (англ. Full Homomorphic Encryption, FHE). Наиболее подходящая библиотека, реализующая принцип FHE, – это библиотека TenSEAL, так как она изначально создавалась на языке программирования C++, но имеет развертку для языка Python, что существенно облегчает ее применение в текущем исследовании.

Ниже представлен алгоритм построения системы.

Производится установка основных библиотек, необходимых для построения модели вычислительной системы через команду **pip** в среде программирования.

Далее импортируем в код модули библиотек при помощи команды **import**.

Загрузка тренировочных и тестовых данных через команду **datasets.MNIST** и их преобразование в тензоры при помощи команды **transforms.ToTensor()** показана на рис. 7.

```
# Загрузка и подготовка данных MNIST
train_data = datasets.MNIST('data', train=True, download=True, transform=transforms.ToTensor())
test_data = datasets.MNIST('data', train=False, download=True, transform=transforms.ToTensor())
```

Рис. 7. Загрузка тренировочного и тестового датасета

Создание загрузчика данных, необходимого для тренировочного набора данных, выполняется через команду **train_loader** с размером блока 64, а подвыборка набора тестовых данных – через команду **test_loader** с размером блока в 10 элементов. Участок кода продемонстрирован на рис. 8.

```
# Создание загрузчиков данных
train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size, shuffle=True)
K = 10  # Размер тестовой выборки
subsample_test_indices = torch.randperm(len(test_data))[:K]
test_loader = torch.utils.data.DataLoader(test_data, batch_size=batch_size, sampler=torch.u
```

Рис. 8. Создание загрузчика

Создание вычислительной модели и ее определение через команду **class ConvNet**, а также ее инициализация представлены на рис. 9.

```
# Определение модели нейронной сети
class ConvNet(torch.nn.Module):
    def __init__(self, hidden=64, output=10):
        super(ConvNet, self).__init__()
        self.conv1 = torch.nn.Conv2d(1, 4, kernel_size=7, padding=0, stride=3)
        self.fc1 = torch.nn.Linear(256, hidden)
        self.fc2 = torch.nn.Linear(hidden, output)
```

Рис. 9. Инициализация модели

Обучение модели благодаря функции **Train** для будущего выполнения алгебраических операций над зашифрованными тензорами показано на рис. 10.

```
# Функция для обучения модели
def train(model, train_loader, criterion, optimizer, n_epochs=10):
    model.train()
    for epoch in range(1, n_epochs + 1):
        train_loss = 0.0
        for data, target in train_loader:
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, target)
            loss.backward()
            optimizer.step()
            train_loss += loss.item()
        train_loss = train_loss / len(train_loader)
        print('Epoch: {} \tTraining Loss: {:.6f}'.format(epoch, train_loss))
    model.eval()
    return model
```

Рис. 10. Функция обучения модели

Функция тестирования модели для проверки работоспособности, а также вычисление средней точности показаны на рис. 11.

```
# Функция для тестирования модели
def test(model, test_loader, criterion):
    test_loss = 0.0
    class_correct = list(0. for i in range(10))
    class_total = list(0. for i in range(10))
    model.eval()
    for data, target in test_loader:
        output = model(data)
        loss = criterion(output, target)
        test_loss += loss.item()
        _, pred = torch.max(output, 1)
        correct = np.squeeze(pred.eq(target.data.view_as(pred)))
        for i in range(len(target)):
            label = target.data[i]
            class_correct[label] += correct[i].item()
            class_total[label] += 1
    test_loss = test_loss / len(test_loader)
    print(f'Test Loss: {test_loss:.6f}\n')
    for label in range(10):
        print(f'Test Accuracy of {label}: {int(100 * class_correct[label] / class_total[label])}% ({int(np.sum(class_correct[label]))}/{int(np
    print(f'\nTest Accuracy (Overall): {int(100 * np.sum(class_correct) / np.sum(class_total))}% ({int(np.sum(class_correct))}/{int(np.sum(cla
```

Рис. 11. Функция тестирования модели

Создание зашифрованной модели, содержащей преобразованные в шифротекст тензоры через команду **EncConvNet** и копирование весов из обученной модели Conv, показаны на рис. 12.

```
# Класс для зашифрованной нейронной сети
class EncConvNet:
    def __init__(self, torch_nn):
        # Копирование весов и смещений из обученной модели ConvNet
        self.conv1_weight = torch_nn.conv1.weight.data.view(
            torch_nn.conv1.out_channels, torch_nn.conv1.kernel_size[0],
            torch_nn.conv1.kernel_size[1]
        ).tolist()
        self.conv1_bias = torch_nn.conv1.bias.data.tolist()

        self.fc1_weight = torch_nn.fc1.weight.T.data.tolist()
        self.fc1_bias = torch_nn.fc1.bias.data.tolist()

        self.fc2_weight = torch_nn.fc2.weight.T.data.tolist()
        self.fc2_bias = torch_nn.fc2.bias.data.tolist()
```

Рис. 12. Создание зашифрованной модели

Эта модель имеет большее количество функций, необходимых для верного проведения шифрования данных и последующих операций над ними. Далее представлена таблица результатов работы системы обработки зашифрованных данных (табл. 2).

Таблица 2

Результат работы модели, основанной на методе гомоморфного шифрования

Точность в зашифрованной модели	0.9894
Время работы	650.1342 с
Использовано ОЗУ	2363.18 Мб

Как показывает практическое исследование модели, основанной на методе гомоморфного шифрования, быстродействие падает из-за специфики принципа работы. Данные поступают на вход единым пакетом, а не частями, поэтому сначала создается тензор, состоящий полностью из незашифрованной информации, а затем он преобразуется в зашифрованный блок, что создает дополнительную нагрузку на алгоритм и требует больше оперативной

памяти. Однако стоит отметить, что, несмотря на ресурсоемкость операции, конечный показатель точности довольно высок.

2.2. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ

Оба метода позиционируются как инструменты для работы с зашифрованной информацией, а также предназначены для сохранения конфиденциальности данных. Наглядно сравним показатели быстродействия, ресурсоемкости и точности при помощи табл. 3.

Таблица 3

Сравнение моделей

Название метода	Точность	Быстродействие	Ресурсоемкость
МРС	0.9792	174.1779 с	1863.18 Мб
ГШ	0.9994	550.1342 с	2363.25 Мб
Разница	0.202	375,9563 с	500.07 Мб

Из вышеприведенных данных можем утверждать следующее: скорость быстродействия выше у модели, построенной на разделительных вычислениях, чем у модели, построенной на гомоморфном шифровании, примерно на 376 секунд при обработке датасета на 1000 изображений. Достигается это за счет того, что перед обработкой полученных данных методом разделительных вычислений сет элементов делится надвое и одновременно шифруется, в то время как алгоритм, построенный на гомоморфном шифровании, обрабатывает данные целиком, превращает полученный датасет в тензор, а уже потом начинает процесс шифрования. Также по этой причине алгоритму на МРС требуется меньше оперативной памяти, чем алгоритму на FHE. В то же время показатель точности у разделительных вычислений немного ниже, чем у FHE. Вызвано это потерями «крупиц» при раздельном шифровании датасета и разделительных вычислений. Для малых объемов информации потеря одного-трех процентов данных не столь критична, однако при работе с большими данными это может негативно сказаться на точности измерений.

В статье проведено исследование моделей нейронных сетей, построенных на основе методов гомоморфного шифрования, а именно метода полного гомоморфного шифрования и метода разделительных вычислений. Рассмот-

рены две модели: первая модель использует метод MPC, а вторая – метод FHE.

Анализ показывает, что каждая из моделей имеет свои преимущества и недостатки. Модель на основе MPC демонстрирует высокую скорость работы и меньшую ресурсоемкость благодаря параллельной обработке данных. Этот метод лучше подходит для задач, требующих быстрого выполнения и работы с небольшими объемами данных. Однако точность вычислений у этой модели несколько ниже из-за возможных потерь данных при параллельной обработке зашифрованных данных.

С другой стороны, модель на основе FHE отличается высокой точностью вычислений, что делает ее предпочтительной для задач, требующих обработки большого количества данных и не имеющих приоритета по времени выполнения. Однако этот метод требует значительных вычислительных ресурсов для обработки данных. Связано это с тем, что при использовании этого метода датасет поступает целиком, что требует большего времени для обработки данных на каждом этапе вычислений.

ЗАКЛЮЧЕНИЕ

В результате сравнения двух моделей установлено, что ни один из использующихся в исследовании методов не является универсальным. Выбор метода зависит от конкретных задач и требований к системе. Если приоритетом является быстродействие и обработка небольших объемов данных, лучше использовать метод MPC. Если же важна точность результатов, предпочтительнее использовать метод FHE, несмотря на его высокую ресурсоемкость.

Таким образом, в ходе исследования продемонстрировано, что оба метода имеют свои области применения и могут эффективно использоваться для решения различных задач машинного обучения с обработкой зашифрованных данных.

Подводя итоги, можно сказать, что с современным уровнем цифровизации и информатизации всех отраслей жизни человека всё более насущным становится вопрос сохранения конфиденциальности данных и работы с ними в зашифрованном виде. Основными методами для таких операций являются разделительные вычисления и гомоморфное шифрование, так как обеспечивают безопасность личной информации человека, но имеют ряд особенностей, которые порождают вопросы о том, какой метод эффективнее.

Относительно недавно человечество открыло для себя нейронные технологии, которые способны облегчить ряд задач, встающих перед ним.

Главным отличием таких технологий является то, что они не программируются в привычном понимании, а обучаются. Стоит отметить, что любое обучение – это работа с данными. Задача методов гомоморфного шифрования и разделительных вычислений состоит в том, чтобы сохранить безопасность и конфиденциальность информации, которая будет использоваться для машинного обучения.

По завершении исследования можно сделать вывод, что нет универсального метода работы с зашифрованными данными. Опираясь на полученные результаты, можно сказать, что разделительные вычисления больше подходят для решения задач, требующих быстродействия и работы с небольшим объемом данных. Гомоморфное шифрование больше подходит для получения не столько быстрого, сколько более точного результата. Обработка больших данных – трудоемкая операция с высокой точностью результатов.

СПИСОК ЛИТЕРАТУРЫ

1. TenSeal – Python FHE Library: сайт. – URL: https://pypi.org/project/tenseal/ (дата обращения: 28.08.2024).

2. Tensorflow – Python MPC Library: сайт. – URL: https://www.tensorflow.org/?hl=ru (дата обращения: 28.08.2024).

3. Полностью гомоморфное шифрование (обзор) / Л.К. Бабенко, Ф.Б. Буртыка, О.Б. Макаревич, А.В. Трепачева // Вопросы защиты информации. – 2015. – № 3 (110). – С. 3–26. – URL: https://elibrary.ru/item.asp?id=24833959 (дата обращения: 28.08.2024).

4. What is Secure Multiparty Computation // Inpher. Technology. – URL: https://inpher.io/technology/what-is-secure-multiparty-computation/ (accessed: 28.08.2024).

5. Multy-Party Computation method / GitHub. – URL: https://github.com/rdragos/awesome-mpc (accessed: 28.08.2024).

6. Multy-Party Computation encrypted / GitHub. – URL: https://github.com/tf-encrypted/tf-encrypted (accessed: 28.08.2024).

7. Full homomorphic encryption in python / GitHub. – URL: https://github.com/ViktorAxelsen/TFE-GNN (accessed: 28.08.2024).

8. PyTorch framework: website. – URL: https://pytorch.org/ecosystem/ (accessed: 28.08.2024).

9. *Nitin Kendre*. 7 Essential techniques for data preprocessing using python: a guide for data scientists / DEV Community. – URL: https://dev.to/newbie_coder/

7-essential-techniques-for-data-preprocessing-using-python-a-guide-for-data-scientists-3ijk (accessed: 28.08.2024).

10. AI Confidential: How can machine learning on encrypted data improve privacy protection? / Ericsson blog. – URL: https://www.ericsson.com/en/blog/2021/9/machine-learning-on-encrypted-data (accessed: 28.08.2024).

11. *Monteiro Tiago Capelo*. How homomorphic encryption works / FreeCodeCamp. – URL: https://www.freecodecamp.org/news/homomorphic-encryption-in-plain-english/ (accessed: 28.08.2024).

12. Google Colaboratory: website. – URL: https://colab.google/ (accessed: 29.08.2024).

13. Python 3.12.4 // Python: website. – URL: https://www.python.org/downloads/ (accessed: 29.08.2024).

14. *Crockett E.* Building machine learning models with encrypted data / Amazon Science Blog. – 2021. – January 5. – URL: https://www.amazon.science/blog/building-machine-learning-models-with-encrypted-data (accessed: 29.08.2024).

15. Federated Learning / Papers with Code: website. – URL: https://paperswithcode.com/task/federated-learning (accessed: 29.08.2024).

16. Dataset MNIST / Kaggel: website. – URL: https://www.kaggle.com/datasets/hojjatk/mnist-dataset (accessed: 29.08.2024).

17. Encrypted key-value database using homomorphic encryption / ZAMA. – URL: https://www.zama.ai/post/encrypted-key-value-database-using-homomorphic-encryption (accessed: 29.08.2024).

18. Программный комплекс для моделирования сверточных нейронных сетей, сохраняющих конфиденциальность, в условиях ограниченных вычислительных ресурсов: свидетельство о гос. регистрации программы для ЭВМ № 2024614441 / Лапина М.А., Фисенко Н.С., Бабенко М.Г. – URL: https://www1.fips.ru/ofpstorage/Doc/PrEVM/RUNWPR/000/002/024/614/441/2024614441-00001/document.pdf (дата обращения: 29.08.2024).

Лапина Мария Анатольевна, кандидат физико-математических наук, доцент, доцент кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. E-mail: mlapina@ncfu.ru

Ардеев Дмитрий Юрьевич, ассистент кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. E-mail: ardeev.dima345@gmail.com

Лапин Виталий Геннадьевич, кандидат физико-математических наук, доцент кафедры вычислительной математики и кибернетики Северо-Кавказского федерального университета. E-mail: vitlx@yandex.ru

DOI: 10.17212/2782-2230-2024-3-34-52

Comparative analysis of methods for building encrypted data processing systems and their comparison for solving machine learning problems*

Lapina M.A.[1], Ardeev D.Yu.[2], Lapin V.G.[3]

[1] *Federal State Autonomous Educational Institution of Higher Education "North Caucasus Federal University", 1 Pushkina Street, Stavropol, 355017, Russian Federation, associate professor of the department of information security of automated systems. E-mail: mlapina@ncfu.ru*

[2] *Federal State Autonomous Educational Institution of Higher Education "North Caucasus Federal University", 1 Pushkina Street, Stavropol, 355017, Russian Federation, assistant of the department information security of automated systems. E-mail: ardeev.dima345@gmail.com.*

[3] *Federal State Autonomous Educational Institution of Higher Education "North Caucasus Federal University", 1 Pushkina Street, Stavropol, 355017, Russian Federation, associate professor of the department of computational mathematics and cybernetics. E-mail: vitlx@yandex.ru*

The article discusses two methods for constructing encrypted data processing systems based on homomorphic encryption and separation calculations. Each method differs from each other in the presented algorithm and data processing model. Basic operations on encrypted data are considered, namely, multiplication and addition. The following libraries were used for the study: tenSEAL, tensorflow and PyTorch. The tenSEAL library is a fork of the full homomorphic encryption tool created for the C++ programming language, but adapted for the Python programming language used in the study. This library allows you to use the method of full homomorphic encryption in constructing a computational model, the task of which will be to process encrypted data. To implement the method of separation calculations, better known as multi-party computation, the tensorflow library will be used, which allows you to create several tensors and train them simultaneously, which in turn makes it possible to implement the principle of separation calculations.

Keywords: encryption, confidentiality, homomorphic encryption, separation computing, tensors, ciphertexts

* *Received 13 August 2024.*

REFERENCES

1. *TenSeal – Python FHE Library*. Website. Available at: https://pypi.org/project/tenseal/ (accessed 28.08.2024).

2. Tensorflow – Python MPC Library. Website. Available at: https://www.tensorflow.org/?hl=ru (accessed 28.08.2024).

3. Babenko L.K., Burtyka F.B., Makarevich O.B., Trepacheva A.V. Polnost'yu gomomorfnoe shifrovanie (obzor) [Fully homomorphic encryption (Review)]. *Voprosy zashchity informatsii = Information Security Questions*, 2015, no. 3 (110), pp. 3–26. Available at: https://elibrary.ru/item.asp?id=24833959 (accessed 28.08.2024).

4. Inpher. Technology. *What is Secure Multiparty Computation*. Available at: https://inpher.io/technology/what-is-secure-multiparty-computation/ (accessed 28.08.2024).

5. GitHub. *Multy-Party Computation method*. Available at: https://github.com/rdragos/awesome-mpc (accessed 28.08.2024).

6. GitHub. *Multy-Party Computation encrypted*. Available at: https://github.com/tf-encrypted/tf-encrypted (accessed 28.08.2024).

7. GitHub. *Full homomorphic encryption in python*. Available at: https://github.com/ViktorAxelsen/TFE-GNN (accessed 28.08.2024).

8. *PyTorch framework*. Website. Available at: https://pytorch.org/ecosystem/ (accessed 28.08.2024).

9. Nitin Kendre. *7 Essential techniques for data preprocessing using python: a guide for data scientists*. DEV Community. Available at: https://dev.to/newbie_coder/7-essential-techniques-for-data-preprocessing-using-python-a-guide-for-data-scientists-3ijk (accessed 28.08.2024).

10. Ericsson blog. *AI Confidential: How can machine learning on encrypted data improve privacy protection?* Available at: https://www.ericsson.com/en/blog/2021/9/machine-learning-on-encrypted-data (accessed 28.08.2024).

11. Monteiro Tiago Capelo. *How homomorphic encryption works*. FreeCodeCamp. Available at: https://www.freecodecamp.org/news/homomorphic-encryption-in-plain-english/ (accessed 28.08.2024).

12. *Google Colaboratory*. Website. Available at: https://colab.google/ (accessed 29.08.2024).

13. Python 3.12.4. Available at: https://www.python.org/downloads/ (accessed 29.08.2024).

14. Crockett E. Building machine learning models with encrypted data. *Amazon Science Blog*, 2021, January 5. Available at: https://www.amazon.science/blog/building-machine-learning-models-with-encrypted-data (accessed 29.08.2024).

15. Papers with Code. *Federated Learning*. Available at: https://papers-withcode.com/task/federated-learning (accessed 29.08.2024).

16. Kaggel. *Dataset MNIST*. Available at: https://www.kaggle.com/datasets/hojjatk/mnist-dataset (accessed 29.08.2024).

17. ZAMA. *Encrypted key-value database using homomorphic encryption*. Available at: https://www.zama.ai/post/encrypted-key-value-database-using-homomorphic-encryption (accessed 29.08.2024).

18. Lapina M.A., Fisenko N.S., Babenko M.G. *Programmnyi kompleks dlya modelirovaniya svertochnykh neironnykh setei, so-khranyayushchikh konfidentsial'nost', v usloviyakh ogranichennykh vychislitel'nykh* resursov [Software package for modeling privacy-preserving convolutional neural networks under limited computing conditions resources]. The Certificate on official registration of the computer program. No. 2024614441, 2024. Available at: https://www1.fips.ru/ofpstorage/Doc/PrEVM/RUNWPR/000/002/024/614/441/2024614441-00001/document.pdf (accessed 29.08.2024).

Для цитирования:

Лапина М.А., *Ардеев Д.Ю.*, *Лапин В.Г.* Сравнительный анализ методов построения систем обработки зашифрованных данных и их сравнение для решения задач машинного обучения // Безопасность цифровых технологий. – 2024. – № 3 (114). – С. 34–52. – DOI: 10.17212/2782-2230-2024-3-34-52.

For citation:

Lapina M.A., Ardeev D.Yu., Lapin V.G. Sravnitel'nyi analiz metodov postroeniya sistem obrabotki zashifrovannykh dannykh i ikh sravnenie dlya resheniya zadach mashinnogo obucheniya [Comparative analysis of methods for building encrypted data processing systems and their comparison for solving machine learning problems]. *Bezopasnost' tsifrovykh tekhnologii = Digital Technology Security*, 2024, no. 3 (114), pp. 34–52. DOI: 10.17212/2782-2230-2024-3-34-52.