Сергей В. Запечников[1], Андрей Ю. Щербаков[2]
[1]Национальный исследовательский ядерный университет «МИФИ»,
Каширское ш., 31, Москва, 115409, Россия
[2]Центр развития криптовалют и цифровых финансовых активов ВИНИТИ РАН
Усиевича ул., 20, Москва, 125190, Россия
[1]e-mail: SVZapechnikov@mephi.ru, http://orcid.org/0000-0002-7975-6040
[2]e-mail: x509@ras.ru, http://orcid.org/0000-0002-1593-6704

КОНФИДЕНЦИАЛЬНОЕ МАШИННОЕ ОБУЧЕНИЕ НА ОСНОВЕ ДВУСТОРОННИХ ПРОТОКОЛОВ БЕЗОПАСНЫХ ВЫЧИСЛЕНИЙ*

DOI: http://dx.doi.org/10.26583/bit.2021.4.03

Аннотация. Статья посвящена анализу систем конфиденциального машинного обучения, основанных на концепции безопасных двусторонних вычислений. Приводятся общие сведения о конфиденциальном машинном обучении, анализируются цели и задачи его применения. Предлагается обобщенная модель архитектуры конфиденциального машинного обучения, отражающая основные функциональные блоки систем. Рассматривается постановка задачи безопасных многосторонних вычислений. Приводятся описания криптографических примитивов и протоколов, используемых для реализации двусторонних протоколов безопасных вычислений, включая гомоморфное шифрование, схемы разделения секрета, протоколы забывающей передачи, GC-схемы. Анализируются существующие системы конфиденциального машинного обучения на основе двусторонних протоколов безопасных вычислений. Основное внимание уделяется алгоритмическим аспектам организации систем, методам и протоколам защиты данных в них. Рассматриваются системы, стойкие к получестному и активному противнику, как основанные на универсальных модулях безопасных двусторонних вычислений, так и специализированные, предназначенные для обеспечения конфиденциальности конкретных технологий машинного обучения, таких как свёрточные нейронные сети. Подробно рассматриваются реализованные прототипы таких систем. Основываясь на результатах проведенного анализа, формулируются выводы о перспективах развития систем конфиденциального машинного обучения.
Ключевые слова: конфиденциальное машинное обучение, безопасные многосторонние вычисления, GC-схемы, забывающая передача, схемы разделения секрета, гомоморфное шифрование.
Для цитирования: ЗАПЕЧНИКОВ, Сергей В.; ЩЕРБАКОВ, Андрей Ю. КОНФИДЕНЦИАЛЬНОЕ МАШИННОЕ ОБУЧЕНИЕ НА ОСНОВЕ ДВУСТОРОННИХ ПРОТОКОЛОВ БЕЗОПАСНЫХ ВЫЧИСЛЕНИЙ. Безопасность информационных технологий, [S.l.], т. 28, № 4, с. 39–51, 2021. ISSN 2074-7136. URL: https://bit.mephi.ru/index.php/bit/article/view/1374. DOI: http://dx.doi.org/10.26583/bit.2021.4.03.

***Благодарности.** Работа выполнена при поддержке Министерства науки и высшего образования РФ (проект государственного задания № 0723-2020-0036).*

Sergey V. Zapechnikov[1], Andrey Yu. Shcherbakov[2]
[1]National Research Nuclear University MEPhI (Moscow Engineering Physics Institute),
Kashirskoe shosse, 31, Moscow, 115409, Russia
[2]All-Russian Institute for Scientific and Technical Information of Russian Academy of Sciences (VINITI RAS),
Usievicha str., 20, Moscow, 125190, Russia
[1]e-mail: SVZapechnikov@mephi.ru, http://orcid.org/0000-0002-7975-6040
[2]e-mail: x509@ras.ru, http://orcid.org/0000-0002-1593-6704

Privacy-preserving machine learning based on secure two-party computations*

DOI: http://dx.doi.org/10.26583/bit.2021.4.03

Abstract. The paper is devoted to the analysis of privacy-preserving machine learning systems based on secure two-party computations. The paper provides introductory information about privacy-preserving machine learning systems, analyses the goals and objectives of its application. A generalized model of privacy-preserving machine learning architecture is proposed, reflecting the main functional blocks of the systems. The formulation of the problem of secure multi-party computation is considered. The descriptions of cryptographic primitives and protocols used to implement two-party secure computation protocols, including homomorphic encryption, secret sharing schemes, oblivious transfer and garbled circuits, are given. The current privacy-preserving machine learning systems based on two-party secure computations are analyzed. The main attention is paid to algorithmic aspects of systems, methods and protocols of information security in them. Systems resistant to semi-honest and active adversaries are considered, both based on universal modules for secure two-party computations, and specialized ones designed to ensure the privacy of specific machine learning technologies, such as convolutional neural networks. Implemented prototypes of several such systems are considered in detail. Based on the results of the analysis, conclusions are formulated about the features of the future privacy-preserving machine learning systems.

Keywords: privacy-preserving machine learning, secure multi-party computations, garbled circuits, oblivious transfer, secret sharing scheme, homomorphic encryption.

For citation: ZAPECHNIKOV, Sergey V.; SHCHERBAKOV, Andrey Yu. Privacy-preserving machine learning based on secure two-party computations. IT Security (Russia), [S.l.], v. 28, n. 4, p. 39–51, 2021. ISSN 2074-7136. URL: https://bit.mephi.ru/index.php/bit/article/view/1374. DOI: http://dx.doi.org/10.26583/bit.2021.4.03.

***Acknowledgement.** This work was supported by the Ministry of Science and Higher Education of the Russian Federation (state assignment project No. 0723-2020-0036).*

Введение

Одна из самых значительных тенденций современных компьютерных наук и информационных технологий – бурное развитие технологий и систем искусственного интеллекта (ИИ). Немало говорится о том, что ИИ формирует ядро нового технологического уклада. Как любой комплекс новых технологий, он проходит путь от теоретической разработки, экспериментов и прототипов к широкому внедрению во многие сферы деловой деятельности. Одним из важных критериев принятия обществом новых технологий, безусловно, является доверие к ним. Один из важных аспектов доверия к информационным технологиям – информационная безопасность.

Технологии ИИ представляют собою конгломерат самых разнообразных методов и алгоритмов: логических выводов, поиска, оптимального управления и др. Однако ни у кого не вызывает сомнения, что ядро технологий ИИ составляют методы и модели машинного обучения, в первую очередь, глубокого обучения. В связи с этим первостепенный интерес представляют механизмы обеспечения информационной безопасности систем машинного обучения.

1. Конфиденциальное машинное обучение

Задачу конфиденциального машинного обучения (КМО) можно определить как обеспечение гарантий конфиденциальности данных каждого из участников системы машинного обучения в условиях, когда лица, предоставляющие обучающую выборку на этапе обучения модели (training) либо предоставляющие запросы к модели на этапе её эксплуатации (inference) и ожидающие получения ответов на свои запросы (далее будем называть их клиентами), дистанционно взаимодействуют с обладателем модели, способным выполнять вычисления с помощью этой модели (далее будем называть его сервером). При этом клиенты заинтересованы в неразглашении своих данных (обучающей выборки, запросов к модели, ответов на них) как друг другу, так и обладателю модели. В то же время обладатель модели заинтересован в неразглашении параметров своей модели

клиентам. Конкретные ситуации, в которых возникает такая заинтересованность, могут различаться: например, при обработке его персональных данных, а также данных, составляющих врачебную, налоговую или банковскую тайну. Заинтересованность обладателя модели может возникать при предоставлении платных услуг (например, прогнозирования) с использованием модели. Такого рода услуги получили обобщенное наименование «машинное обучение как сервис» (MLaaS – Machine Learning as a Service).

В оригинале концепция КМО называется privacy-preserving machine learning, т.е. буквально «машинное обучение, сохраняющее приватность». Как известно, конфиденциальность информации неформально можно определить как гарантии того, что она не станет известна лицам, которым она не предназначена. Однако применительно к системам машинного обучения, как и в ряде других случаев, в зарубежной литературе чаще всего используется понятие приватности (privacy). Под ним понимается обеспечение тайны частной жизни, гарантии того, что никто не сможет узнать о владельце (источнике) какой-либо информации больше того, что он сам желает. Каждый владелец вправе регулировать, кто и какие сведения о нём получает. Это свойство применимо не только к конфиденциальным данным. Приватность включает в себя такие составляющие как конфиденциальность, разграничение доступа, анонимность, несвязываемость действий или событий, неразличимость инициатора события и ряд других в зависимости от конкретной ситуации. Современные технологии предоставляют много способов узнать о человеке больше, чем он сам того пожелает: прямые утечки данных, восстановление информации по косвенным признакам, интеллектуальный анализ данных с применением машинного обучения (выявление «тонких» и скрытых закономерностей в данных) и многие другие. В широком смысле слова приватность подразумевает защищенность от «излишне любопытного» нарушителя.

Конфигурацию, в которой осуществляется КМО, можно представить архитектурной моделью, показанной на рис. 1.

Рис. 1. Обобщенная модель архитектуры систем КМО
Fig. 1. The generalized architectural model of privacy-preserving machine learning system

Модель включает в себя три основных функциональных блока:

- блок генерации задачи, передаваемой на аутсорсинг, и получения результата решения задачи;

- блок преобразования (трансформации) задачи и проверки результата решения задачи;
- блок решения задачи.

Физическое соответствие функциональных блоков участникам вычислительного процесса может быть неоднозначным. Наиболее предпочтительной является схема, когда первые два блока расположены на доверенной системе-клиенте, третий блок – у недоверенной серверной стороны, которая может быть представлена одним или несколькими физически выделенными серверами либо виртуальными машинами. Однако такая конфигурация может приводить к высокой вычислительной нагрузке на клиента. Второй возможной конфигурацией является размещение первого блока на доверенной системе-клиенте, второго блока – на доверенной системе-шлюзе, третьего блока – на недоверенных серверах. Такая модель позволяет удобно разместить по блокам всю основную функциональность, обеспечивающую безопасность информации, но потенциально связана с большим количеством уязвимостей.

Исполнитель вычислений (серверная сторона) может быть представлен единственным физическим вычислителем либо кластером взаимосвязанных устройств, которые в этом случае должны выполнять между собой многосторонние протоколы безопасных вычислений.

Блоки подготовки задачи, а также обратного преобразования и проверки решения задачи могут включать в себя разную функциональность в зависимости от метода решения задачи. Так, при использовании гомоморфного шифрования первый из этих блоков будет выполнять преобразование зашифрования входных данных, второй – расшифрования результата решения задачи. В случае использования безопасных многосторонних вычислений на основе схем разделения секрета первый блок будет разделять конфиденциальные входные данные на доли для последующей пересылки долей вычислителям, второй блок – собирать результат решения задачи из долей, переданных вычислителями. В ряде случаев к обратному преобразованию может добавляться проверка корректности решения задачи, например, с помощью доказательств с нулевым разглашением, прилагаемых вычислителями к своим долям решения.

В качестве промежуточной модели рассматривается также возможность участия в вычислениях клиента (при наличии у него достаточных вычислительных ресурсов) наряду с серверами (это характерно для протоколов на основе схем разделения секрета).

От недоверенных компонентов модели требуется следующая функциональность:

- выполнение алгоритмов обработки данных без раскрытия открытого текста данных;
- генерация доказательств корректного выполнения требуемых алгоритмов обработки данных.

От доверенных компонентов модели требуется следующая функциональность:

- подготовка исходных данных решаемой задачи для передачи недоверенным компонентам;
- прием результатов решения задачи от недоверенных компонентов и преобразование их в формат, пригодный для восприятия заказчиком вычислений;
- при необходимости – проверка корректности решения задачи недоверенным компонентом.

В связи с этим наиболее подходящими инструментами реализации функций, требуемых от недоверенных компонентов, выглядят следующие криптографические методы:

- полностью и частично гомоморфные схемы шифрования (в части алгоритмов выполнения арифметических операций над зашифрованными данными);
- GC-схемы (garbled circuits);

- схемы разделения секрета (операции над долями разделенных секретов);
- протоколы безопасных многосторонних вычислений (SMPC – secure multi-party computations);
- доказательства с нулевым разглашением (алгоритмы генерации доказательств).

Наиболее подходящими инструментами реализации функций, требуемых от доверенных компонентов, представляются следующие криптографические методы:

- полностью и частично гомоморфные схемы шифрования (в части алгоритмов зашифрования и расшифрования данных);
- схемы разделения секрета (операции по разделению секретов на доли и восстановлению секретов из долей);
- доказательства с нулевым разглашением (алгоритмы проверки доказательств).

2. Безопасные многосторонние вычисления

Напомним постановку задачи безопасных многосторонних вычислений [1]. Рассматривается многосторонний протокол, в котором каждый из участников имеет свой индивидуальный секрет. Нужно вычислить некоторую функцию от этих секретов, так чтобы результат вычислений был известен всем участникам группы, но сами секреты не были разглашены участниками протокола друг другу или какой-либо третьей стороне. Формально: пусть участники криптографического протокола $P_1, P_2, \dots, P_n$ имеют конфиденциальные входные данные $x_1, x_2, \dots, x_n$ соответственно. В результате выполнения протокола ими совместно должна быть вычислена функция вида $y = f(x_1, x_2, \dots, x_n)$ при выполнении следующих свойств:

- *корректности*: каждый из участников $P_1, P_2, \dots, P_n$ получает y;
- *приватности* (конфиденциальности): никому из участников и посторонних лиц не разглашается никакая дополнительная информация, кроме той, которую они знали до начала выполнения протокола, в том числе и промежуточные результаты вычислений.

Частный случай безопасных многосторонних вычислений – безопасные двусторонние вычисления. Пусть участники криптографического протокола P_1, P_2 имеют конфиденциальные входные данные x_1, x_2 соответственно. В результате выполнения протокола ими совместно должна быть вычислена функция вида $y = f(x_1, x_2)$ при выполнении аналогичных свойств корректности и приватности.

Важный криптографический примитив для реализации безопасных многосторонних протоколов – *протокол (1,2)-«забывающей передачи»* (oblivious transfer), обозначаемый (1,2)-OT. В протоколе участвуют отправитель S и получатель R. Входные данные: S имеет секреты $x_0, x_1 \in \{0,1\}^n$, R – бит выбора $b \in \{0,1\}$. В результате выполнения протокола R должен получить бит x_b, S не получает никакой новой информации (т.е. для него выбор, сделанный R, остаётся неизвестным).

В качестве примера реализации протокола (1,2)-OT рассмотрим протокол Bellare – Micali [2]. Пусть $\mathbb{G}$ – группа простого порядка p с образующим элементом g. Пусть H – хэш-функция, отображающая $\mathbb{G} \to \{0,1\}^l$ (она моделирует случайного оракула). Пусть m_0, m_1 – сообщения отправителя, пусть $b \in \{0,1\}$ – бит выбора получателя. Протокол выполняется следующим образом.

1. Отправитель S выбирает случайным образом элемент группы $c \leftarrow^R \mathbb{G}$ и пересылает c участнику R.

2. Получатель R выбирает случайный ключ $k \leftarrow^R \mathbb{Z}_p$, вычисляет два открытых ключа схемы Эль-Гамаля: $y_b \leftarrow g^k$, $y_{1-b} \leftarrow \frac{c}{g^k}$ и пересылает y_0, y_1 участнику S.

3. Если $y_0 \cdot y_1 \neq c$, то отправитель S прерывает участие в протоколе. В противном случае S выбирает $r_0, r_1 \leftarrow^R \mathbb{Z}_p$, вычисляет шифртексты согласно схеме Эль-Гамаля: $c_0 \leftarrow \left(g^{r_0}, H\left(y_0^{r_0}\right)\right) \oplus m_0,\ c_1 \leftarrow \left(g^{r_1}, H\left(y_1^{r_1}\right)\right) \oplus m_1$ – и пересылает c_0, c_1 участнику R.

4. Получатель R разбирает шифртекст $c_b = (v_0, v_1)$, расшифровывает его, используя знание ключа k: $m_b = H\left(v_0^k\right) \oplus v_1$ и выдает m_b в качестве ответа.

Протокол (1,2)-OT используется в качестве примитива в специальной криптосхеме, называемой *GC-схемой* (англ. garbled circuits), предложенной Яо [3]. Криптосхема Яо включает в себя алгоритм генерации GC-схемы и протокол вычисления GC-схемы. Рассмотрим их подробнее.

Алгоритм генерации GC-схемы использует следующие параметры: булеву схему C, реализующую функцию F, а также параметр безопасности κ.

Алгоритм состоит из трёх шагов.

1. *Генерация меток проводников.* Для каждого проводника w_i схемы C случайным образом выбираются метки проводников $w_i^b = \left(k_i^b \in_R \{0,1\}^\kappa, p_i^b \in_R \{0,1\}\right)$ такие, что $p_i^b = 1 - p_i^{1-b}$.

2. *Конструирование GC-схемы.* Для каждого логического вентиля G_i схемы C в топологическом порядке выполняются следующие действия:

а) предполагается, что G_i – элементарная булева схема с двумя входами, реализующая функцию g_i: $w_c = g_i(w_a, w_b)$, где входные метки – $w_a^0 = (k_a^0, p_a^0)$, $w_a^1 = (k_a^1, p_a^1)$, $w_b^0 = (k_b^0, p_b^0)$, $w_b^1 = (k_b^1, p_b^1)$,а выходные метки – $w_c^0 = (k_c^0, p_c^0)$, $w_c^1 = (k_c^1, p_c^1)$;

б) создаётся запутанная таблица для G_i. Для каждой из четырёх возможных комбинаций входов $v_a, v_b \in \{0,1\}$ схемы G_i устанавливается $e_{v_a,v_b} = H\left(k_a^{v_a} \| k_b^{v_b} \| i\right) \oplus w_c^{g_i(v_a,v_b)}$. Записи таблицы значений e_{v_a,v_b} сортируются по входным указателям, помещая запись e_{v_a,v_b} в позицию $\langle p_a^{v_a}, p_b^{v_b} \rangle$.

3. *Создание выходной таблицы декодирования.* Для каждого выходного проводника w_i – выхода вентиля G_j с метками $w_i^0 = (k_i^0, p_i^0)$, $w_i^1 = (k_i^1, p_i^1)$ создаётся запутанная таблица выходов для обеих возможных значений $v \in \{0,1\}$. Пусть $e_v = H(k_i^v \| "out" \| j) \oplus v$ (поскольку вычисляется XOR-сумма с единственным битом, используется самый младший бит выхода H). Записи таблицы значений e сортируются по входным битам-указателям, помещая запись e_v в позицию p_i^v (конфликта не будет, так как $p_i^1 = p_i^0 \oplus 1$).

Протокол вычисления GC-схемы использует следующие параметры: $x \in \{0,1\}^n$ и $y \in \{0,1\}^n$ – входные двоичные векторы участников P_1 и P_2 соответственно, C – булева схема, реализующая функцию F.

Протокол состоит из пяти шагов.

1. *Генерация и пересылка GC-схемы.* P_1 играет роль генератора GC-схемы, выполняя для этого предыдущий алгоритм. Затем P_1 пересылает P_2 полученную GC-схему $\hat{C}$, включая выходную таблицу декодирования.

2. *Пересылка активных меток проводников для входа x.* P_1 пересылает P_2 активные метки проводников для тех проводников, которые соответствуют входу, предоставляемому P_1.

3. *Пересылка активных меток проводников для входа y.* Для каждого проводника w_i, для которого вход предоставляет P_2, участники P_1 и P_2 выполняют протокол «забывающей передачи», в котором P_1 играет роль отправителя, P_2 – получателя. При этом:

- два входных секрета P_1 – это две метки проводников, бит выбора P_2 – это его входной бит на этом проводнике;
- после завершения протокола «забывающей передачи» P_2 получает активную метку на этом проводнике.

4. *Вычисления по GC-схеме.* P_2 вычисляет полученную схему $\hat{C}$ вентиль за вентилем, начиная с активных меток на входных проводниках, а именно, для каждого вентиля G_i с запутанной таблицей $T = (e_{0,0}, \dots, e_{1,1})$ и активными метками входных проводников $w_a = (k_a, p_a)$, $w_b = (k_b, p_b)$ участник P_2 вычисляет активную метку выходного проводника $w_c = (k_c, p_c)$: $w_c = H(k_a \| k_b \| i) \oplus e_{p_a, p_b}$.

5. Получение выхода, используя выходную таблицу декодирования. Когда все вентили схемы $\hat{C}$ вычислены, P_2 получает биты выходного двоичного вектора, используя строку «*out*» в качестве второго ключа для декодирования выхода с выходных вентилей. Участник P_2 пересылает полученный выходной вектор участнику P_1, и оба они заканчивают протокол с этим результатом.

3. Системы КМО на основе двусторонних протоколов безопасных вычислений

Проанализируем подходы к построению систем КМО на основе двусторонних протоколов безопасных вычислений. Основное внимание будем уделять алгоритмическим аспектам организации систем, методам и протоколам защиты данных в них,

Модули ABY и ABY 2.0. Модуль ABY (аббревиатура означает «Arithmetic, Boolean, Yao») [4] – это программный бэкенд-компонент систем КМО для безопасных двусторонних вычислений, основанный на идее сочетания арифметических, булевых и GC-схем. Для этого используются три формы разделения секрета: арифметическое, булево и Яо-разделение. Это необходимо для того, чтобы выполнять каждую операцию над разделенными секретами наиболее быстрым образом.

Для каждой формы разделения секрета используется своя семантика, доступен свой набор стандартных операций и свои характерные приемы оптимизации при выполнении протоколов.

Арифметическое разделение секрета. Арифметическое разделение $\langle x \rangle^A$ l-битного числа x – это доли $\langle x \rangle_0^A + \langle x \rangle_1^A \equiv x (mod\ 2^l)$, где $\langle x \rangle_0^A, \langle x \rangle_1^A \in \mathbb{Z}_{2^l}$.

Для разделения долей выполняется протокол $Shr_i^A(x)$: участник P_i выбирает число $r \in_R \mathbb{Z}_{2^l}$, полагает $\langle x \rangle_i^A = x - r$, посылает r участнику P_{1-i}, который полагает $\langle x \rangle_{1-i}^A = r$.

Для восстановления долей выполняется протокол $Rec_i^A(x)$: участник P_{1-i} посылает свою долю $\langle x \rangle_{1-i}^A$ участнику P_i, который вычисляет $x = \langle x \rangle_0^A + \langle x \rangle_1^A$.

Каждая арифметическая схема – это последовательность элементарных операций сложения и умножения над кольцом целых чисел, которые выполняются следующим образом.

Сложение $\langle z \rangle^A = \langle x \rangle^A + \langle y \rangle^A$: участник P_i локально вычисляет $\langle z \rangle_i^A = \langle x \rangle_i^A + \langle y \rangle_i^A$.

Умножение $\langle z \rangle^A = \langle x \rangle^A \cdot \langle y \rangle^A$: предварительно вычисляются по специальному протоколу «арифметические тройки» вида $\langle c \rangle^A = \langle a \rangle^A \cdot \langle b \rangle^A$. Далее участник P_i полагает $\langle e \rangle_i^A = \langle x \rangle_i^A - \langle a \rangle_i^A$, $\langle f \rangle_i^A = \langle y \rangle_i^A - \langle b \rangle_i^A$, оба участника выполняют протоколы $Rec^A\langle e \rangle$, $Rec^A\langle f \rangle$, затем каждый P_i вычисляет $\langle z \rangle_i^A = i \cdot e \cdot f + f \cdot \langle a \rangle_i^A + e \cdot \langle b \rangle_i^A + \langle c \rangle_i^A$.

Булево разделение секрета. Булево разделение $\langle x \rangle^B$ бита x между двумя участниками определяется как $\langle x \rangle_0^B \oplus \langle x \rangle_1^B = x$, где доли $\langle x \rangle_0^B, \langle x \rangle_1^B \in \mathbb{Z}_2$.

Для разделения бита на доли выполняется протокол $Shr_i^B(x)$: участник P_i выбирает $r \in_R \{0,1\}$, вычисляет $\langle x \rangle_i^B = x \oplus r$, посылает r участнику P_{1-i}, который устанавливает у себя $\langle x \rangle_{1-i}^B = r$.

Для восстановления бита *x* выполняется протокол $Rec_i^B(x)$: участник P_{1-i} посылает свою долю $\langle x\rangle_{1-i}^B$ участнику P_i, который вычисляет $x = \langle x\rangle_0^B \oplus \langle x\rangle_1^B$.

Любая вычислимая функция может быть представлена в виде булевой схемы, состоящей из элементарных логических операций XOR и AND.

Для выполнения операции XOR над разделенными битами $\langle z\rangle^B = \langle x\rangle^B \oplus \langle y\rangle^B$ каждый участник P_i локально вычисляет $\langle z\rangle_i^B = \langle x\rangle_i^B + \langle y\rangle_i^B$.

Для выполнения операции AND над разделенными битами $\langle z\rangle^B = \langle x\rangle^B \wedge \langle y\rangle^B$: используются предварительно вычисленные по специальному протоколу «булевы мультипликативные тройки» вида $\langle c\rangle^B = \langle a\rangle^B \wedge \langle b\rangle^B$. Далее участник P_i вычисляет $\langle e\rangle_i^B = \langle a\rangle_i^B \oplus \langle x\rangle_i^B$, $\langle f\rangle_i^B = \langle b\rangle_i^B \oplus \langle y\rangle_i^B$, оба участника выполняют протоколы $Rec^B\langle e\rangle$, $Rec^B\langle f\rangle$, после чего каждый P_i вычисляет $\langle z\rangle_i^B = i \cdot e \cdot f \oplus f \cdot \langle a\rangle_i^B \oplus e \cdot \langle b\rangle_i^B \oplus \langle c\rangle_i^B$.

Модуль ABY поддерживает ряд других логических операций, оптимизированная форма выполнения которых описана в [5, 6].

Разделение Яо. Пусть P_0 – обладатель схемы, которую нужно вычислить (garbler), P_1 – участник, который выполняет вычисления по этой схеме. P_0 имеет для каждого проводника схемы *w* по два ключа: k_0^w и k_1^w. P_1 обладает одним из этих ключей, не зная, какому из двух возможных открытых текстов (нулевой либо единичный бит) он соответствует.

Яо-разделение $\langle x\rangle^Y$ величины *x* определяется как $\langle x\rangle_0^Y = k_0$ и $\langle x\rangle_1^Y = k_x = k_0 + xR$, где R – κ-битная случайная двоичная строка с $R[0] = 1$, выбранным P_0, а $k_0^w \in_R \{0,1\}^\kappa$, $k_1^w = k_0^w \oplus R$.

Для разделения бита *x* выполняются протоколы $Shr_0^Y(x)$ и $Shr_1^Y(x)$. Протокол $Shr_0^Y(x)$: участник P_0 выбирает $\langle x\rangle_0^Y = k_0 \in_R \{0,1\}^\kappa$ и посылает $k_x = k_0 + xR$ участнику P_1. Протокол $Shr_1^Y(x)$: оба участника выполняют протокол C-OT$_\kappa^1$, где P_0 выступает в роли отправителя, вводит функцию корреляции $f_R(x) = x \oplus R$ и получает $(k_0, k_1 = k_0 \oplus R)$ с $k_0 \in_R \{0,1\}^\kappa$, а P_1 – в роли получателя с битом выбора *x* и получает $\langle x\rangle_1^Y = k_x$.

Для восстановления бита *x* выполняется протокол $Rec_i^Y(x)$: участник P_{1-i} посылает свой так называемый перестановочный бит $\pi = \langle x\rangle_{1-i}^Y[0]$ участнику P_i, который вычисляет $x = \pi \oplus \langle x\rangle_i^Y[0]$.

Если нужно разделить не один бит, а *l*-битную двоичную величину, то каждая из перечисленных выше операций выполняется *l* раз параллельно.

Для выполнения операции XOR над разделенными битами $\langle z\rangle^Y = \langle x\rangle^Y \oplus \langle y\rangle^Y$ каждый участник P_i локально вычисляет $\langle z\rangle_i^Y = \langle x\rangle_i^Y + \langle y\rangle_i^Y$.

Для выполнения операции AND над разделенными битами $\langle z\rangle^Y = \langle x\rangle^Y \wedge \langle y\rangle^Y$ участник P_0 создает запутанную таблицу $Gb_{\langle z\rangle_0^Y}(\langle x\rangle_0^Y, \langle y\rangle_0^Y)$ [7]. Участник P_0 отсылает таблицу участнику P_1, который расшифровывает ее, используя ключи $\langle x\rangle_1^Y$ и $\langle y\rangle_1^Y$.

Модуль ABY поддерживает ряд других операций с GC-схемами, описанных в [8].

Указанный набор базовых операций формирует своеобразный виртуальный процессор для выполнения безопасных двусторонних вычислений.

Кроме этого, модуль ABY реализует встроенный набор функций, алгоритмов и протоколов для всех возможных конвертаций битовых строк между открытым текстом, арифметическим, булевым и Яо-разделениями.

Модуль ABY 2.0 [5] представляет собой существенно модернизированную версию модуля ABY, что позволило значительно ускорить прежде всего выполнение операций умножения разделенных секретов и конвертацию между различными формами их представления. Оптимизации касаются в основном арифметической формы

представления. Булева форма теперь трактуется как частный случай арифметической для кольца $\mathbb{Z}_2$, а Яо-разделения секрета остались практически без изменений.

Система EzPC. Система EzPC [9] концептуально представляет собой компилятор универсального назначения для преобразования программ, написанных на высокоуровневых языках программирования для одного вычислителя, в программы, реализующие ту же функциональность в виде комплекса протоколов безопасных двусторонних вычислений. В качестве бэкенда используется модуль ABY [4]. В качестве промежуточного представления алгоритмов используется код на языке C++ с вызовами функций интерфейса прикладного программирования модуля ABY.

EzPC протестирован на различных алгоритмах машинного обучения, таких как линейная регрессия, наивный байесовский классификатор, решающие деревья и пр.

В качестве направления дальнейшей работы авторы рассматривают разработку фронтенда для EzPC, который позволил бы автоматически транслировать модели машинного обучения, созданные при помощи библиотеки TensorFlow, в комплекс протоколов безопасных двусторонних вычислений. Отметим, что эта идея позднее была реализована в системе CrypTFlow [10].

Модуль CrypTFlow2. CrypTFlow2 представляет собой программный модуль, предназначенный для интеграции в систему КМО, предназначенный для использования на стадии применения обученных глубоких нейронных сетей для получения прогнозных ответов на запросы пользователей [11].

Модуль CrypTFlow2 встраивается в качестве бэкенда в систему CrypTFlow [10], заменяя собой первоначально использовавшийся в ней модуль ABY для реализации двусторонних протоколов безопасных вычислений. В качестве фронтенда используется модуль Athos. Цель замены модуля ABY на CrypTFlow2 – в том, чтобы повысить производительность системы, что достигается отказом от использования GC-схем.

Все вычисляемые функции представляются в виде арифметических либо булевых схем (в зависимости от того, какая форма удобнее). Обычно арифметическое представление удобно для целочисленных операций (скалярное произведение векторов, матричное умножение), булево представление – для логических операций (сравнение, вычисление старшего бита разделенного числа и т.п.).

В основе математического обеспечения модуля CrypTFlow2 лежат (2,2)-пороговые СРС над кольцом $\mathbb{Z}_L$, где $L = 2^l$ (обычно l=32) и над кольцом $\mathbb{Z}_n$, где n – произвольное положительное целое число.

Алгоритм разделения секрета $Share^L(x)$ берет элемент $x \in \mathbb{Z}_L$ и создает доли $\langle x \rangle_0^L$ и $\langle x \rangle_1^L$, которые также являются элементами кольца $\mathbb{Z}_L$. Доли генерируются путем выбора случайных элементов $\langle x \rangle_0^L$ и $\langle x \rangle_1^L$ таких, что $\langle x \rangle_0^L + \langle x \rangle_1^L = x$, где «+» – сложение в кольце $\mathbb{Z}_L$.

Алгоритм восстановления секрета $Reconst^L(\langle x \rangle_0^L, \langle x \rangle_1^L)$ берет на входе две доли секрета и возвращает на выходе $x = \langle x \rangle_0^L + \langle x \rangle_1^L$.

Другие используемые в модуле CrypTFlow2 криптографические примитивы – это протоколы 1-из-2 и 1-из-k-забывающей передачи (1,2)-ОТ и (1,k)-ОТ, функции-мультиплексоры MUX (на вход подаются от участников P_1, P_2 арифметические доли числа $a \in \mathbb{Z}_n$ и булевы доли бита выбора c, а возвращаются доли числа a, если c=1, и доли числа 0 в противном случае), протоколы B2A преобразования булева представления долей в арифметическое, схема гомоморфного шифрования BFV.

Авторы отмечают, что одну и ту же функциональность модуля CrypTFlow2 в принципе можно реализовать двумя способами: либо на основе протоколов ОТ, либо на основе гомоморфного шифрования.

В модуле CrypTFlow2 на базе перечисленных и ряда других примитивов реализован новый протокол MILL решения так называемой задачи Яо о миллионерах (участник P_0 имеет на входе число x, участник P_1 – число y, им необходимо выполнить безопасный двусторонний протокол, который даст им на выходе доли бита 1, если $x < y$, и доли бита 0 в противном случае. На основе этого протокола реализуются протоколы DReLU вычисления производной функции ReLU для l-битных целых чисел и для произвольного $\mathbb{Z}_n$, которые необходимы для безопасного вычисления нелинейных функций активации слоев нейросети.

Другой комплект протоколов предназначен для безопасного двустороннего выполнения операции целочисленного деления (в частном случае – усечения числа, если делитель – степень 2). В качестве примитивов в этих протоколах используются ранее упомянутые протоколы DReLU, MILL, B2A, (1,k)-OT.

Оба комплекта протоколов применяются для вычисления нелинейных слоев нейросетей – функций ReLU, Maxpool, Argmax.

При вычислении полной нейронной сети чередуются линейные и нелинейные слои. Для вычисления линейных слоев с характерными для них операциями матричного умножения и свертки используются представления операндов в виде элементов кольца $\mathbb{Z}_L$, $L = 2^l$, для чего более выгодным по критерию времени выполнения оказывается использование комплекта протоколов на базе (1,2)-OT и (1,k)-OT.

Для вычисления нелинейных слоев более выгодно представлять операнды в виде элементов кольца $\mathbb{Z}_n$, где n – простое число, при котором такое кольцо становится полем. Поэтому более выгодным по критерию времени выполнение оказывается применение комплекта протоколов на базе гомоморфного шифрования. Кроме того, выбор может зависеть от конфигурации сети, в которой выполняются вычисления. Как показали эксперименты авторов, в глобальной сети протоколы на основе гомоморфного шифрования всегда работают быстрее, в локальной сети возможен выбор в зависимости от дополнительных требований и критериев.

Система Gazelle. Gazelle – система КМО для сверточных нейронных сетей, основанная на линейных гомоморфных схемах шифрования и GC-схемах [12]. В ней реализована идея предобработки (предварительных вычислений) перед выполнением онлайн-фазы протокола. Гомоморфные схемы шифрования используются для выполнения линейных операций над шифртекстами и используются при вычислении линейных слоев – слоев свертки. GC-схемы используются для нелинейных операций. Сложность вычислений и объем передаваемых данных распределяются примерно поровну между предварительной и интерактивной фазами протокола. В настоящее время система Gazelle представляет в основном исторический и познавательный интерес, так как более новые решения превосходят ее по показателям производительности и стойкости. Идеи системы Gazelle использованы при разработке систем Delphi и Muse.

Система Delphi. Delphi – система КМО, основанная на безопасных двусторонних вычислениях, предназначенная для получения прогнозных ответов на запросы клиентов к обученной сверточной нейросети, расположенной на стороне сервера. Предполагается, что не более, чем один из участников вычислений может быть получестным нарушителем, второй является честным [13].

Система Delphi основана на идеях системы Gazelle [12], однако основной целью разработчиков было снижение довольно высокой вычислительной и коммуникационной нагрузки на участников протокола на предварительной и интерактивной фазах протокола.

Оптимизация вычислений на линейных слоях нейросетях основана на том, что синаптические веса обученной нейросети уже известны к моменту выполнения запроса

клиента, поэтому вычисления долей этих параметров с использованием линейных гомоморфных схем шифрования можно вынести в фазу предварительных вычислений. Интерактивная фаза при этом значительно облегчается, так как операции выполняются только над долями секретов без необходимости интерактивного взаимодействия.

Оптимизация вычислений на нелинейных слоях (поддерживается только одна нелинейная функция активации – ReLU) достигается путем выборочной замены функций ReLU в некоторых слоях их квадратичными приближениями. Полная замена на всех слоях не применяется, так как она приводит к неприемлемому снижению точности прогнозирования. Для поиска тех слоев, в которых возможна такая замена, система Delphi включает в себя специальный планировщик. Он оптимизирует архитектуру нейросети, исходя из заданных потребителем требований производительности и точности. Таким образом, особенностью применения системы Delphi является необходимость частичного изменения архитектуры нейросети, что неизбежно приводит к некоторому снижению точности прогнозирования по сравнению с исходной нейросетью.

Система Muse. Muse – система КМО, основанная на безопасных двусторонних вычислениях, предназначенная для получения прогнозных ответов на запросы клиентов к обученной сверточной нейросети, расположенной на стороне сервера, стойкая к атакам злоумышленных клиентов [14]. Основная идея, заложенная в основу Muse, – новый криптографический протокол условного раскрытия секретов (CDS – conditional disclosure of secrets), предназначенный для переключения между аутентифицированными аддитивными долями секретов и метками GC-схем, а также улучшенная процедура генерации троек Бивера (Beaver’s triples). Эти идеи позволили сконцентрировать значительную часть вычислений протокола на предварительной фазе.

Система Muse основывается на идеях системы Delphi [13]: для вычислений на линейных слоях нейросети используется аддитивное разделение секрета, на нелинейных слоях – GC-схемы.

Чтобы предотвратить попытки клиентов исказить свои доли, пересылаемые в протоколе серверу, предложено использовать теоретико-информационные гомоморфные коды аутентификации сообщений. Однако их применение сталкивается с рядом технических сложностей. Чтобы преодолеть их, в системе Muse применено более сложное техническое решение.

При вычислении линейных слоев аутентификация обеспечивается посредством гомоморфных кодов аутентификации, а для нелинейных слоев клиенты получают только те метки для GC-схем, которые соответствуют их аутентифицированным долям секрета, что обеспечивается посредством протокола CDS. Авторы системы разработали методику высокопроизводительного исполнения такого протокола и перенесли его на этап предварительных вычислений, что позволило максимизировать производительность системы.

Заключение

В ходе работы выполнено поисковое исследование и проведен обзор систем КМО на основе двусторонних протоколов безопасных вычислений. Выделена неформальная постановка задачи и сформулирована обобщенная модель систем КМО. Выполнен обзор принципов организации и функционирования, архитектуры и математического обеспечения наиболее известных из существующих систем КМО, основанных на двусторонних протоколах безопасных вычислений.

Основным алгоритмическим инструментом безопасных вычислений в системах КМО являются схемы разделения секрета. В системах КМО используются три вида разделения секрета: арифметическое, булево и Яо-разделение, каждое из которых позволяет выполнять безопасные вычисления с разделенными секретами по

соответствующему виду схем: арифметическим, булевым либо GC-схемам (garbled circuits). Наблюдается тенденция постепенного отказа от GC-схем в пользу все более широкого применения арифметических и булевых схем. Сферой применения GC-схем остаются преимущественно сложные нелинейные функции. Наиболее вероятная причина этого в высокой сложности проектирования и реализации таких протоколов.

Анализ систем КМО на основе двусторонних протоколов безопасных вычислений позволяет выделить две ведущих линии их развития:

- системы с ядром на основе универсальных программных модулей с набором базовых операций, позволяющих построить протокол безопасных вычислений, реализующий произвольную функциональность, ограниченную лишь сложностью выполнения протокола (например, модули ABY, ABY 2.0 и надстройки над ними, такие как EzPC);
- специализированные КМО с комплектом криптографических примитивов, оптимизированным для достижения высоких целевых показателей (производительности, точности прогнозирования) для определенных методов машинного обучения или моделей нарушителя (например, системы CrypTFlow2, Muse).

Продолжением исследования будет анализ систем КМО на основе трехсторонних протоколов безопасных вычислений.

СПИСОК ЛИТЕРАТУРЫ:

1. Evans D., Kolesnikov V., Rosulek M. A pragmatic introduction to secure multi-party computation. – 182 p. URL: https://securecomputation.org/docs/pragmaticmpc.pdf (дата обращения: 14.09.2021).
2. Bellare M., Micali S. Non-interactive oblivious transfer and applications. In Advances in Cryptology – CRYPTO'89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. P. 547–557. DOI: http://dx.doi.org/10.1007/0-387-348-5-0_48.
3. Yao, C. How to generate and exchange secrets. Proc. of 27th Annual Symposium on Foundations of Computer Science, 1986. P. 162–167. DOI: http://dx.doi.org/10.1145/266420.266424.
4. Demmler, D. ABY – a framework for efficient mixed-protocol secure two-party computation. D. Demmler, T. Schneider, M. Zohner. 22nd Network and Distributed System Security Symposium (NDSS'15), Internet Society, San Diego, CA, USA, February 8-11, 2015. URL: https://encrypto.de/papers/DSZ15.pdf (дата обращения: 14.09.2021).
5. Patra A. ABY2.0: Improved mixed-protocol secure two-party computation. A. Patra, T. Schneider, A. Suresh et al. URL: https://ia.cr/2020/1225 (дата обращения: 14.09.2021).
6. Mohassel P. How to hide circuits in MPC an efficient framework for private function evaluation. P. Mohassel, S. Sadeghian. Proceedings of EUROCRYPT'13, ser. LNCS, vol. 7881. Springer, 2013. P. 557–574. DOI: http://dx.doi.org/10.1007/978-3-642-38348-9_33.
7. Schneider T. GMW vs. Yao? Efficient secure two-party computation with low depth circuits. T. Schneider and M. Zohner. Financial Cryptography and Data Security (FC'13), ser. LNCS, vol. 7859. Springer, 2013. P. 275–292. DOI: http://dx.doi.org/10.1007/978-3-642-39884-1_23.
8. Bellare M. Efficient garbling from a fixed-key blockcipher. M. Bellare, V. Hoang, S. Keelveedhi, et al. Symposium on Security and Privacy (S&P'13). IEEE, 2013. P. 478–492. DOI: http://dx.doi.org/10.1109/SP.2013.39.
9. Kolesnikov V. Improved garbled circuit building blocks and applications to auctions and computing minima. V. Kolesnikov, A.-R. Sadeghi, T. Schneider. Cryptology And Network Security (CANS'09), ser. LNCS, vol. 5888. Springer, 2009. P. 1–20. DOI: http://dx.doi.org/10.1007/978-3-642-10433-6_1.
10. Chandran N., Gupta D., Rastogi A., Sharma R., Tripathi S. EzPC: Programmable and Efficient Secure Two-Party Computation for Machine Learning. 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 2019. P. 496–511. DOI: http://dx.doi.org/10.1109/EuroSP.2019.00043.
11. Kumar E. et al. CrypTFlow: Secure TensorFlow Inference. arXiv preprint. 2020. – 18 p. URL: https://arxiv.org/pdf/1909.07814v2.pdf (дата обращения: 14.09.2021).
12. Rathee D. et al. CrypTFlow2: Practical 2-Party Secure Inference. arXiv preprint. 2020. – 18 p. URL: https://arxiv.org/pdf/2010.06457.pdf (дата обращения: 14.09.2021).

13. Juvekar C. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. C. Juvekar, V. Vaikuntanathan, A. Chandrakasan. Cryptology ePrint Archive. 2021. – 17 p. URL: https://eprint.iacr.org/2018/073.pdf (дата обращения: 14.09.2021).
14. Mishra P. Delphi: A Cryptographic Inference Service for Neural Networks. P. Mishra, R. Lehmkuhl, A. Srinivasan et al. Proc. of USENIX Security 2020 (USENIX Security Symposium). URL: https://www.usenix.org/system/files/sec20spring_mishra_prepub.pdf (дата обращения: 14.09.2021).
15. Lehmkuhl R. Muse: Secure Inference Resilient to Malicious Clients. R. Lehmkuhl, P. Mishra, A. Srinivasan et al. Proc. of USENIX Security 2021 (USENIX Security Symposium). URL: https://people.eecs.berkeley.edu/~raluca/MUSEcamera.pdf (дата обращения: 14.09.2021).

REFERENCES:

[1] Evans D., Kolesnikov V., Rosulek M. A pragmatic introduction to secure multi-party computation. – 182 p. URL: https://securecomputation.org/docs/pragmaticmpc.pdf (accessed: 14.09.2021).
[2] Bellare M., Micali S. Non-interactive oblivious transfer and applications. In Advances in Cryptology – CRYPTO ’89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. P. 547–557. DOI: http://dx.doi.org/10.1007/0-387-348-5-0_48.
[3] Yao, C. How to generate and exchange secrets. Proc. of 27th Annual Symposium on Foundations of Computer Science, 1986. P. 162–167. DOI: http://dx.doi.org/10.1145/266420.266424.
[4] Demmler, D. ABY – a framework for efficient mixed-protocol secure two-party computation. D. Demmler, T. Schneider, M. Zohner. 22nd Network and Distributed System Security Symposium (NDSS'15), Internet Society, San Diego, CA, USA, February 8-11, 2015. URL: https://encrypto.de/papers/DSZ15.pdf (accessed: 14.09.2021).
[5] Patra A. ABY2.0: Improved mixed-protocol secure two-party computation. A. Patra, T. Schneider, A. Suresh et al. URL: https://ia.cr/2020/1225 (accessed: 14.09.2021)
[6] Mohassel P. How to hide circuits in MPC an efficient framework for private function evaluation. P. Mohassel, S. Sadeghian. Proceedings of EUROCRYPT’13, ser. LNCS, vol. 7881. Springer, 2013. P. 557–574. DOI: http://dx.doi.org/10.1007/978-3-642-38348-9_33.
[7] Schneider T. GMW vs. Yao? Efficient secure two-party computation with low depth circuits. T. Schneider and M. Zohner. Financial Cryptography and Data Security (FC’13), ser. LNCS, vol. 7859. Springer, 2013. P. 275–292. DOI: http://dx.doi.org/10.1007/978-3-642-39884-1_23.
[8] Bellare M. Efficient garbling from a fixed-key blockcipher. M. Bellare, V. Hoang, S. Keelveedhi, et al. Symposium on Security and Privacy (S&P’13). IEEE, 2013. P. 478–492. DOI: http://dx.doi.org/10.1109/SP.2013.39.
[9] Kolesnikov V. Improved garbled circuit building blocks and applications to auctions and computing minima. V. Kolesnikov, A.-R. Sadeghi, T. Schneider. Cryptology And Network Security (CANS’09), ser. LNCS, vol. 5888. Springer, 2009. P. 1–20. DOI: http://dx.doi.org/10.1007/978-3-642-10433-6_1.
[10] Chandran N., Gupta D., Rastogi A., Sharma R., Tripathi S. EzPC: Programmable and Efficient Secure Two-Party Computation for Machine Learning. 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 2019. P. 496–511. DOI: http://dx.doi.org/10.1109/EuroSP.2019.00043.
[11] Kumar E. et al. CrypTFlow: Secure TensorFlow Inference. arXiv preprint. 2020. – 18 p. URL: https://arxiv.org/pdf/1909.07814v2.pdf (accessed: 14.09.2021).
[12] Rathee D. et al. CrypTFlow2: Practical 2-Party Secure Inference. arXiv preprint. 2020. – 18 p. URL: https://arxiv.org/pdf/2010.06457.pdf (accessed: 14.09.2021).
[13] Juvekar C. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. C. Juvekar, V. Vaikuntanathan, A. Chandrakasan. Cryptology ePrint Archive. 2021. – 17 p. URL: https://eprint.iacr.org/2018/073.pdf (accessed: 14.09.2021).
[14] Mishra P. Delphi: A Cryptographic Inference Service for Neural Networks. P. Mishra, R. Lehmkuhl, A. Srinivasan et al. Proc. of USENIX Security 2020 (USENIX Security Symposium). URL: https://www.usenix.org/system/files/sec20spring_mishra_prepub.pdf (accessed: 14.09.2021).
[15] Lehmkuhl R. Muse: Secure Inference Resilient to Malicious Clients. R. Lehmkuhl, P. Mishra, A. Srinivasan et al. Proc. of USENIX Security 2021 (USENIX Security Symposium). URL: https://people.eecs.berkeley.edu/~raluca/MUSEcamera.pdf (accessed: 14.09.2021).

Поступила в редакцию – 14 сентября 2021 г. Окончательный вариант – 25 ноября 2021 г.
Received – September 14, 2021. The final version – November 25, 2021.