ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № ?, 2014, с. ??–??

УДК 004.021:51

Л. К. Бабенко, Ф. Б. Буртыка, О. Б. Макаревич,
А. В. Трепачева

Защищенные вычисления и гомоморфное шифрование

Аннотация. Рассматривается задача построения систем защищенных криптографических вычислений и её приложения. Анализируются достижения, сделанные исследователями в данной области за последние годы и сложности, с которыми они столкнулись. Рассматривается авторский подход к решению поставленных задач. Дается прогноз перспектив развития отрасли на последующие годы.

Ключевые слова и фразы: полностью гомоморфное шифрование, гомоморфное шифрование, обфускация, облачные вычисления, смарт-карты, децентрализованные сети, машинное обучение.

Введение

Криптография уже давно зарекомендовала себя как надежное средство для гарантированного сохранения конфиденциальности информации. Однако, в связи с ростом и развитием средств передачи информации, телекоммуникационных и компьютерных сетей, и особенно Интернета перед криптографией встают новые задачи. Одной из таких задач является *обеспечение вычислений над зашифрованными данными.*

Суть этой задачи состоит в том, чтобы проводить вычисления над зашифрованными данными без их расшифрования. Впервые такая задача была поставлена в статье [1] изобретателями знаменитой криптосистемы RSA. Уже сама криптосистема RSA обеспечивала мультипликативный гомоморфизм, т.е. позволяла осуществлять перемножение шифртекстов, и после расшифрования извлекать из

Работа поддержана грантом РФФИ .

полученного в результате шифртекста произведение исходных открытых текстов, т.е. выполнялось соотношение

$$D(E(m_1) \cdot E(m_2)) = m_1 \cdot m_2 \ , \tag{1}$$

где $D(\cdot)$ – функция расшифрования, а $E(\cdot)$ – функция зашифрования (таким образом $E(m_1)$ и $E(m_2)$ – шифртексты сообщений m_1 и m_2 соответственно. Особый же интерес представляла возможность построения *полностью* гомоморфного шифрования, т.е. шифрования, позволяющего проводить над шифртекстами *любые необходимые* вычисления. К примеру, такую криптосистему можно было бы получить в случае, если бы она была гомоморфна одновременно и по операции сложение и по операции умножение,

$$\begin{cases} D(E(m_1) \odot E(m_2)) = m_1 \cdot m_2, \\ D(E(m_1) \oplus E(m_2)) = m_1 + m_2. \end{cases} \tag{2}$$

где $\odot$ и $\oplus$ – операции над шифртекстами, соответствующие операциям $\cdot$ и $+$ над открытыми текстами.

Если криптосистема с такими свойствами сможет надежно зашифровать два бита, то поскольку над битами операции сложения и умножения формируют полный по Тьюрингу базис, становится возможным вычислить любую булеву (а следовательно и вообще любую вычислимую) функцию. К примеру, возможно осуществить гомоморфную сортировку (!) набора чисел в зашифрованном виде без расшифрования следующим образом [2] :

```
void bubblesort(int* arr, unsigned n)
{
  for(unsigned i=0;i<n-1;i++)
  {
    for(unsigned j=1;i<n-i;j++)
   {
      arr[j-1] = cmpmin(arr[j-1],arr[j]);
      arr[j] = cmpmax(arr[j-1],arr[j]);
    }
  }
}
```

где cmpmin и cmpmax – функции, возвращающие зашифрованный результат сравнения чисел a и b (соответственно, минимум и максимум). В разделе 3 будет рассмотрен способ реализации таких функций с помощью булевых полиномов.

В течении долгого времени не было известно о том, возможно ли построение криптостойкого полностью гомоморфного шифрования (достаточно полный обзор достижений того времени можно найти в [3] и [4], а небольшой обзор на русском языке – в [5]), пока в работе [6] исследователь из IBM Крейг Джентри представил систему, теоретически позволяющую осуществить такое шифрование.

Вслед за этим отрасль гомоморфной криптографии начала бурно развиваться [7–10] и сейчас уже имеется сотни публикаций посвященных исследованиям в этой области (один из наиболее свежих обзоров можно найти в [11]).

Несмотря на большое количество исследований по гомоморфному и полностью гомоморфному шифрованию (ПГШ), пока нет примеров внедренных в практику и полноценно эксплуатирующихся систем с их применением. В чем тут дело? Коренной недостаток, присущий гомоморфным криптосистемам состоит в том, что в атаках на них может использоваться их дополнительная структура. К примеру, при использовании исходного варианта RSA [12] для цифровой подписи, произведение двух подписей будет давать корректную подпись для произведения двух соответствующх сообщений. Хотя есть много способов избежать такой атаки, к примеру применяя хэш-функции,или используя избыточность вероятностных криптосистем, есть также более сложные атаки (см например [13], где показывается нестойкость криптосхем из [14, 15]). Для криптосхем с открытым ключом желание повысить криптостойкость приводит к снижению эффективности.

Данный недостаток можно преодолеть снижением требований к криптосхеме, а именно позволив ей быть *симметричной*, но компактной (данный термин обозначает отсутствие увеличения шифртекстов в процессе гомоморфных вычислений) и *криптостойкой к атаке на основе известных открытых текстов* [16, 17] что как мы увидим в дальнейшем вполне достаточно для большинства приложений.

Размер ключа симметричного шифра, как правило, в несколько, а то и в десятки раз меньше размера ключа шифра с открытым ключом. В криптосистеме RSA для обеспечения надлежащей надежности и долгосрочной секретности рекомендуется использовать ключ

длиной не менее 4096 бит. Для сравнения, длина ключа в алгоритме ГОСТ 28147-89 равна 256 бит, что в 16 раз меньше, чем длина ключа в криптосистеме RSA. По скорости шифрования симметричная криптосистема DES примерно в тысячу раз лучше, чем RSA.

Требование криптостойкости к атаке на основе известных открытых текстов существенно, поскольку получить шифртекст ПГШ, соответствующий известному открытому тексту можно легко (например вычислив выражение $x^p - x$, где p – максимальный из возможных открытых текстов).

Сложность обеспечения гомоморфных криптосхем необходимой эффективностью и криптостойкостью приводит к вопросу о том, зачем использовать гомоморфное шифрование вместо обычного. Главный интерес изучения гомоморфного шифрования состоит в широком диапазоне приложений как теоретических так и практических.

В данной работе мы подробнее рассмотрим практическую сторону исследований – приложения полностью гомоморфного шифрования: защищенные облачные вычисления, программную и аппаратную обфускацию, защиту децентрализованных беспроводных сетей, машинное обучение на зашифрованных данных.

1. Защищенные облачные вычисления

Одним из самых естественных приложений гомоморфного шифрования является проведение защищенных вычислений над данными клиента, хранящимися на удаленном недоверенном «облачном» сервере. Поясним более подробно, что мы здесь под этим подразумеваем.

Допустим, клиент зашифровал данные стандартным шифром и разместил их на удалённом сервере. В случае необходимости их изменения клиент может *доверить серверу свой секретный ключ*, чтобы сервер расшифровал данные и внёс необходимые изменения, а затем зашифровал снова. Однако в этом случае сервер сможет прочитать все данные клиента. А также возможен перехват ключа при передаче его серверу по каналу, в результате чего данные клиента могут стать доступными некому третьему лицу. Также клиент может скачать свои данные из облака на свой компьютер, расшифровать, провести над ними нужное вычисление и при необходимости зашифровать этот результат и загрузить его на облако. Однако это потребует много времени и вычислительных ресурсов, которыми клиент возможно не располагает, коль скоро он прибегнул к использованию облачного сервиса.

Из вышеописанного вытекают несколько конкретных требований к безопасному (защищенному) облачному сервису. Во-первых, данные клиента должны храниться в таком виде, что при их чтении невозможно было бы понять, что это за данные. То есть данные необходимо шифровать. Причем понятно, что данные должны поступать на сервер уже зашированными. Это означает, что шифрование должно проводиться ещё на стороне клиента. Во-вторых, должна быть возможность обрабатывать эти данные не расшифровывая. Иначе облачный сервер становится всего лишь безопасным хранилищем. А для каждой операции над данными потребуется пересылать их на сторону клиента.

В настоящее время существующие облачные сервисы не являются полностью защищенными. В лучшем случае есть возможность лишь зашифровать данные на стороне пользователя, но это бывает крайне редко. Довольно часто данные зашифрованы ключом, который хранится самим же облаком.

Для создания защищенного облачного сервиса необходимо шифровать данные с помощью гомоморфной схемы шифрования. В этом случае окажется возможным проводить вычисления над данными непосредственно в зашифрованном виде на стороне сервера. При этом шифрование данных будет проводиться на стороне клиента. И таким образом необходимые требования к защищенному облачному сервису будут выполнены.

Перейдем теперь к рассмотрению конкретного примера защищенного облачного сервиса. Речь пойдет об организациии защищенной облачной базы данных (БД). Постановка задачи будет следующей: *Некоторая организация не обладает достаточными ресурсами для хранения и обработки своей БД. Поэтому ей нужен сервис, который позволил бы разместить БД на облачном сервере и затем работать с ней удаленно. Облачному серверу организация, однако, не доверяет. Поэтому все данные шифруются перед отправкой на сервер. Клиенты, обращающиеся к БД, являются программами, запущенными на машинах, находящимися во владении организации. Все клиенты друг другу доверяют и ключ расшифрования у них является общим.* Ниже мы объясним, как данную задачу можно решить с помощью симметричного гомоморфного шифрования.

1.1. Модель БД

Современная реляционная БД является набором прямоугольных таблиц. Для простоты будем далее считать, что таблица только одна. Таблица имеет m атрибутов $a_1, ..., a_m$ и по сути представляет из себя набор записей $\{R_i\}_{i=1}^{w}$, где $R_i = \{v_{i,j}\}_{j=1}^{m}$, $v_{i,j}$ – значение записи R_i для атрибута a_j. Также еще договоримся, что a_j, $v_{i,j}$ и номера записей $i \in \{1, ..., w\}$ являются элементами некоторого большого конечного поля $\mathbb{F}_q$. Клиенту необходимо делать простые SQL запросы к БД, такие как:

(3) SELECT $*$ FROM db WHERE $(a_{z_1} = v_1^*)$ OR... OR $(a_{z_t} = v_t^*)$

(4) SELECT $*$ FROM db WHERE $(a_{z_1} = v_1^*)$ AND... AND $(a_{z_t} = v_t^*)$

Запросы типа (3) будем называть дизъюнктивными, а запросы типа (4) – конъюнктивными.

Обратимся теперь к модели Клиент-Сервер. Облачный сервер S хранит $db = \{R_i\}_{i=1}^{w}$, принадлежащую клиенту K. Периодически K обращается с запросами к S. В результате запроса K должен получить все записи в db, удовлетворяющие условию WHERE. При этом S ничего не должен узнать о значениях $v_i^*, 1 \le i \le t$, участвующих в запросе, а также о том, какие записи в db соответствуют запросу.

Существующий подход к решению данной задачи заключается в том, что обработка запроса проходит в две стадии:

- Сначала K получает номера $i_1, ..., i_n \in \{1, ..., w\}$ записей, который соответствуют условиям в SQL запросе. Этот этап нужно организовать так, чтобы S не узнал $v_i^*, 1 \le i \le t$ и $i_1, ..., i_n$.
- K по очереди извлекает из db записи с индексами $i_1, ..., i_n$. При этом S не должен узнать $i_1, ..., i_n$.

Такая последовательность действий использовалась например в [18]. Однако отметим, что в [18] рассматривалась немного другая задача, т.к. предполагалось, что БД принадлежит S и как следствие она хранилась на S в открытом виде. Мы же рассматриваем случай, когда БД принадлежит K. Поэтому на самом деле S хранит не $db = \{v_{i,j}\}_{i=1,j=1}^{w,m}$, а $db = \{E_{sk}(v_{i,j})\}_{i=1,j=1}^{w,m}$, где E – отображение шифрования, sk – секретный ключ, принадлежащий K.

Рассмотрим теперь по очереди как указанные два этапа обработки запроса можно организовать при условии, что для шифрования $v_{i,j}$ используется криптосистема с гомоморфными свойствами.

1.2. Защищенное получение индексов

Поскольку K хочет скрыть $v_i^*, 1 \leq i \leq t$, ему необходимо их зашифровать. На S он передает пары $(a_{z_k}, E(v_k^*)), 1 \leq i \leq t$. В свою очередь S должен для каждой записи $R_i = \{E(v_{i,j})\}_{j=1}^m, i = 1..w$ провести следующее вычисление:

- Для $\forall z_k = 1..t$ S должен вычислить $e_k = f(E(v_{i,z_k}, E(v_k^*))$, где f - функция, осуществляющая проверку на равенство v_{i,z_k} и v_k^* гомоморфно. То есть $e_k = E(0)$ если $v_{i,z_k} \neq v_k^*$ and $e_k = E(1)$ если $v_{i,z_k} = v_k^*$.
- Далее в зависимости от типа запроса S должен сделать следующее:
  - *Конъюнктивный запрос:* Нужно вычислить
  $$e'_i = Hom_{AND}(e_1, ..., e_t),$$
  где Hom_{AND} - функция, осуществляющая вычисление побитовой конъюнкции нижележащих открытых текстов. Ясно, что если $e'_i = E(1)$, то R_i соответвует запросу, а если $e'_i = E(0)$, то не соответствует. Отметим, что по сути дела выполняется $Hom_{AND}(e_1, ..., e_t) = e_1 \cdot ... \cdot e_t$.
  - *Дизъюнктиный запрос:* Нужно вычислить
  $$e'_i = Hom_{XOR}(e_1, ..., e_t),$$
  где Hom_{XOR} - функция, осуществляющая вычисляющая побитовый XOR нижележащих открытых текстов.
- S отправляет вектор шифртекстов $\vec{Res} = (e'_1, ..., e'_w)$ клиенту K.

Расшифровав компоненты вектора $\vec{Res}$ K узнает индексы $i_1, ..., i_n$ записей, соответствующих его SQL-запросу. Отметим, что в общем представленное решение пока не очень эффективно, так как для того, чтоб получить индексы K должен обработать объем информации в размере по сути одного столбца таблицы *db*. Вопрос о том, как его сократить требует дальнейшей проработки.

Ясно, что для реализации описанного подхода можно использовать симметричную гомоморфную криптосистему.

1.3. Защищенное извлечение записей из БД по их индексам

Для извлечения записей из БД по их индексам K может воспользоваться стандартным протоколом секретного получения информации (СПИ, англ. Private Information Retrieval, PIR) Разработка про-

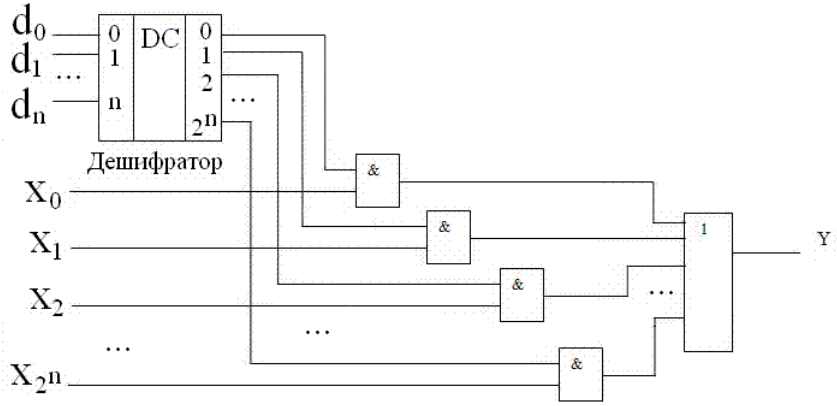

Рис. 1. Мультиплексор

токолов СПИ представляет собой самостоятельный интерес и пользуется большой популярностью у исследователей уже более десяти лет [19] с момента опубликования [20] где эта задача была впервые рассмотрена. На данный момент было предложено большое количество различных протоколов СПИ [21–24]. Наиболее недавние из них [25, 26] основаны на использовании криптосистем с мультипликативными и аддитивными гомоморфными свойствами. Вкратце опишем их принцип работы.

Механизм получения данных из БД по индексу может быть представлен в виде схемы из функциональных элементов (СФЭ) (или логической схемы) [27] называемой *мультиплексор*. Мультиплексор имеет несколько информационных входов, один или более управляющих входов и один выход. Мультиплексор позволяет передавать сигнал с одного из информационных входов на выход. Выбор желаемого входа осуществляется подачей соответствующей комбинации управляющих сигналов на управляющие входы.

Как выглядит полиномиальная функция, реализующая мультиплексор? Пусть $d_0, d_1, ..., d_n \in \{0, 1\}$ – управляющие сигналы, на который подается номер записи, который нужно извлечь. Также пусть $x_0, x_1, ..., x_{2^n}$ – информационные сигналы (т.е. в нашем случае все проиндексированные записи из БД). Тогда для того, чтобы выбрать один из , подав в битовом представлении на входы , нужно вычислить следующую функцию

$$
\begin{aligned}
y = Mul(d_0, ..., d_n, x_0, ..., x_{2^n}) = \\
= x_0 \cdot ((d_0 \oplus 1) \times (d_1 \oplus 1) \times ... \times (d_n \oplus 1)) + \\
+ x_1 \cdot (d_0 \times (d_1 \oplus 1) \times ... \times (d_n \oplus 1)) + \\
+ x_2 \cdot ((d_0 \oplus 1) \times d_1 \times ... \times (d_n \oplus 1)) + ... + \\
+ x_{2^n} \cdot (d_0 \times d_1 \times ... \times d_n)
\end{aligned} \tag{5}
$$

Отметим, что x_i в формуле 1.3 не обязательно являются битами. Они могут, например, являться элементами некоторого конечного поля $\mathbb{F}_q$. Поэтому операции $\{+, \cdot\}$ в формуле 1.3 – это сложение и умножение по модулю q.

Если все x_i зашифрованы с помощью гомоморфной криптосистемы, то $Mul(d_0, ..., d_n, x_0, ..., x_{2^n})$ можно вычислить гомоморфно над шифртекстами. Для этого клиенту достаточно зашифровать двоичные разряды $d_0, d_1, ..., d_n$ индекса нужной ему записи, и отправить эти шифртексты на сервер. Результат гомоморфного вычисления Mul над шифртекстами будет шифртекстом значения x_i, запрашиваемого клиентом. Ясно, что для того, чтобы все работало корректно, используемая криптосистема должна поддерживать, как гомоморфное сложение, так и умножение, однако нет необходимости в том, чтобы она была с открытым ключом.

2. Программная и аппаратная обфускация

Обфускация (от лат. obfuscare – затенять, затемнять; и англ. obfuscate – делать неочевидным, запутанным, сбивать с толку) или запутывание кода — приведение алгоритма или функции к виду, сохраняющему их функциональность, но затрудняющему анализ, понимание принципов работы и модификацию. В идеале, обфусцированная функция (или алгоритм) должна представлять из себя "виртуальный черный ящик в том смысле, что что-нибудь о принципе ее работы можно понять, только анализируя входные данные и результаты работы.

Традиционно западные исследователи уделяют много внимание программной обфускации [28] (даже в тех случаях когда анализируется обфускация логических схем по сути это всего лишь другая модель для программной обфускации). Однако, для России в связи с её уникальной ситуацией задача обфускации возникает в совершенно уникальной постановке.

Далее мы рассмотрим сначала классическую постановку задачи программной обфускации и её применение для защиты программных агентов, а затем перейдем к аппаратой обфускации для защиты проектных решений микроэлектронных схем.

2.1. Защита программных агентов

Как известно, программный агент – это компьютерная программа, которая вступает в отношение посредничества с пользователем или другой программой, при этом выступая от имени кого-то и имея некоторую степень автономности. При этом часто возникают ситуации, в которых необходима их информационная защита.

Рассмотрим следующий пример: специализированный программный агент отправляется посетить серверы нескольких авиакомпаний, узнать из их баз данных, есть ли у них билет, удовлетворяющий заданным критериям и, определив наилучшее предложение из подходящих, забронировать его (активный агент) либо отправить информацию клиенту (пассивный агент). Простейшая атака состоит в том, чтобы вмешаться в работу этого агента и злонамеренно навязать ему предложение какой-либо авиакомпании.

Другой пример: программный агент должен подобрать комплектующие, чтобы собрать наилучшую конфигурацию компьютера (автомобиля) по соотношению цена/эффективность для решения заданной задачи.

Как уже было сказано во введении, ПГШ над битами может вычислить любую функцию, поскольку все традиционные компьютерные архитектуры используют двоичные строки. Таким образом возможно зашифровать целиком всю программу так, что она останется работоспособной. Это может быть использовано для защиты программных агентов от недоверенных хостов с помощью шифрования [29].

Для дальнейшего рассмотрения необходимо формализовать понятие «программа», чтобы дать определение обфускации. Существует много моделей вычислений, однако для нашей цели наиболее удобными являются т.н. нашина Тьюринга (МТ) и схема из функциональных элементов (СФЭ). В рамках этих моделей приведем ниже общепринятые определения обфускации программ.

Определение 1. *Вероятностный алгоритм $\mathcal{O}$ называется обфускатором МТ для множества машин Тьюринга $\mathfrak{F}$, если*

- *(Функциональность) Для каждой МТ $M \in \mathfrak{F}$ строка $\mathcal{O}(M)$ описывает МТ, которая вычисляет ту же функцию, что и M.*
- *(Полиномиальное замедление) Длина описания и время выполнения $\mathcal{O}(M)$ не более чем в полиномиальное число раз больше, чем M. Другими словами, существует полином p такой, что для каждой МТ $M \in \mathfrak{F}$ выполняется $|\mathcal{O}(M)| \leq p(|M|)$ и если M завершается через t шагов, то $\mathcal{O}(M)$ завершается через $p(t)$ шагов.*
- *(Скрытность устройства) для любого PPT A существует PPT S и бесконечно малая функция α такие что*

$$\left| Pr[A(\mathcal{O}(M)) = 1] - Pr[S^{\langle M \rangle}(1^{|M|}) = 1] \right| \leq \alpha(|M|)$$

Определение обфускации СФЭ отличается от приведенного выше по сути только отсутствием условия на завершимость, поскольку оно является следствием конечности размера СФЭ.

Определение 2. *Вероятностный алгоритм $\mathcal{O}$ называется обфускатором СФЭ для множества СФЭ $\mathfrak{F}$, если*

- *(Функциональность) Для каждой СФЭ $C \in \mathfrak{F}$ строка $\mathcal{O}(C)$ описывает СФЭ, которая вычисляет ту же функцию, что и C.*
- *(Полиномиальное замедление) Существует полином p такой, что для каждой СФЭ $C \in \mathfrak{F}$ выполняется $|\mathcal{O}(C)| \leq p(|C|)$.*
- *(Скрытность устройства) для любого PPT A существует PPT S и бесконечно малая функция α такие что для любой СФЭ C выполняется*

$$\left| Pr[A(\mathcal{O}(C)) = 1] - Pr[S^{C}(1^{|C|}) = 1] \right| \leq \alpha(|C|)$$

Доказано [28] что в рамках этих определений универсальная криптостойкая обфускация невозможна, однако можно дать вполне естественное альтернативное определение программной обфускации, в рамках которого возможна универсальная криптостойкая обфускация и которая хорошо подходит для защиты программных агентов от недоверенных хостов.

Определение 3. *Вероятностный алгоритм $\mathcal{O}$ называется обфускатором МТ для множества МТ $\mathfrak{F}$, если*

- *(Функциональность) Для каждой МТ $M \in \mathfrak{F}$ строка $\mathcal{O}(M)$ описывает МТ, которая вычисляет ту же функцию, что и M, а результат возвращается в зашифрованном виде (но так, что все шифртексты имеют размер, который не зависит от M).*

- *(Полиномиальное замедление) Длина описания и время выполнения $\mathcal{O}(M)$ не более чем в полиномиальное число раз больше, чем M. Другими словами, существует полином p такой, что для каждой МТ $M \in \mathfrak{F}$ выполняется $|\mathcal{O}(M)| \leq p(|M|)$ и если M завершается через t шагов, то $\mathcal{O}(M)$ завершается через $p(t)$ шагов.*
- *(Скрытность устройства) для любого PPT A существует PPT S и бесконечно малая функция α такие что*

$$\left| Pr[A(\mathcal{O}(M)) = 1] - Pr[S^{\langle M \rangle}(1^{|M|}) = 1] \right| \leq \alpha(|M|)$$

Определение 4. *Вероятностный алгоритм $\mathcal{O}$ называется обфускатором СФЭ для множества СФЭ $\mathfrak{F}$, если*

- *(Функциональность) Для каждой СФЭ $C \in \mathfrak{F}$ строка $\mathcal{O}(C)$ описывает СФЭ, которая вычисляет ту же функцию, что и C, но результат возвращается в зашифрованном виде (но количество выходов СФЭ увеличивается лишь в фиксированное число раз).*
- *(Полиномиальное замедление) Существует полином p такой, что для каждой СФЭ $C \in \mathfrak{F}$ выполняется $|\mathcal{O}(C)| \leq p(|C|)$.*
- *(Скрытность устройства) для любого PPT A существует PPT S и бесконечно малая функция α такие что*

$$|Pr[A(\mathcal{O}(C)) = 1] - Pr[S^{C}(1^{|1|}) = 1]| \leq \alpha(|C|)$$

При таком определении обфускации можно выполнить следующий протокол:

(1) представить программного агента в зашифрованном виде;
(2) запустить его на выполнение, причем так что входные данные кодировались бы в фломат криптосхемы без ключа;
(3) результаты работы клиент отсылает заказчику в зашифрованном виде;
(4) в случае активного агента (т.е. когда клиент должен произвести какое-то действие) сервер заказчика должен принять шифртекст, убедиться что его отправил клиент и затем отправить клиенту расшифрованное сообщение, позволяющее ему осуществить действие.

Необходимо отметить, что большинство предложенных гомоморфных криптосхем обладают свойством однородности, которое означает что открытый текст m_1 можно представить с помощью некоторого

кодирования без использования ключа в такой форме $C(m_1)$, что будет выполняться

$$D(C(m_1)\ op\ E(m_2)) = m_1\ op\ m_2\ ,$$

для всех операций *op* по которым данная криптосхема гомоморфна. Например, у криптосхемы из такое кодирование тривиально – любой открытый текст является шифртекстом самого себя, а у криптосхем достаточно в качестве шифртекста сообщения m положить матрицу $I \cdot m$, где I – единичная матрица необходимой размерности.

Очевидно, что при такой организации наличие криптосхемы с открытым ключом не требуется.

2.2. Защита проектных решений микроэлектронных схем

В связи со специфической ситуацией, сложившейся в России в сфере производства микроэлектронных схем [30], задача аппаратной обфускации приобретает особую актуальность. Отечественная материально-техническая база не позволяет проводить полный цикл производства интегральных схем с высоким уровнем миниатюризации, а передача некоторых этапов производства стороннему исполнителю порождает проблемы с информационной безопасностью проектных решений. Интересно отметить, что поскольку такого рода задача специфична для России, в иностранной литературе она практически не освещается.

Существует много методов аппаратной обфускации, однако каждый из них обладает своими недостатками [31]. Основные классы таких методов – это введение дополнительных управляющих входов и введение избыточных схемных элементов. Недостатком первого класса методов является то, что для надежной защиты всей схемы необходимо вводить столько дополнительных управляющих входов, сколько на схеме присутствует схемных элементов, что очевидно, неудобно. Недостатком же второго является то, что излишние, фактически нефункциональные (или даже нефункционирующие) схемные элементы могут быть довольно легко автоматически «вычищены» из схемы, как например способом из [28], приводя схему к изначальному виду и тем самым сводя на нет все усилия по её обфускации. Хотелось бы располагать таким методом обфускации, который имел бы гарантированную стойкость против деобфускации, сравнимую с криптографической и в то же время не создавал бы необходимости в большом количестве дополнительных управляющих сигналов.

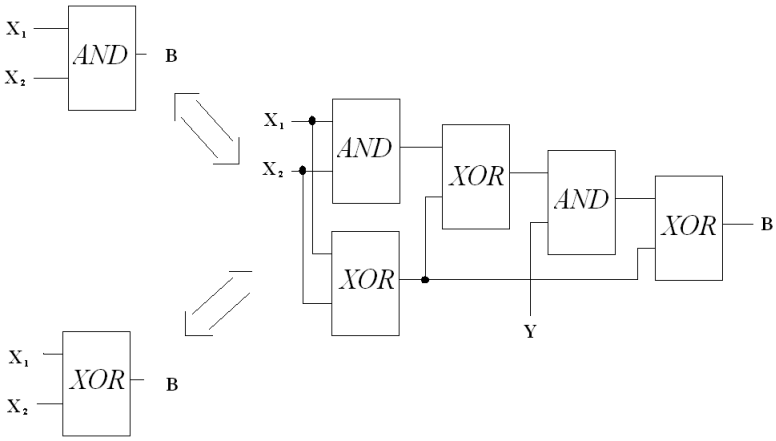

Рис. 2. Комбинированный «универсальный элемент» работающий по-разному в зависимости от управляющего сигнала Y

Можно предложить следующий метод, которым можно обфусцировать любую программу:

(1) представить микроэлектронную схему в базисе СФЭ $\{XOR, AND\}$;
(2) ввести управляющие входы;
(3) заменить логические операции на арифметические;
(4) "встроить"на управляющие входы шифртексты.

Снова воспользовавшись свойством однородности можно избежать необходимости в криптосхеме с открытым ключом.

3. Защита беспроводных децентрализованных сетей связи

Беспроводные децентрализованные самоорганизующиеся сети (англ. Mobile Ad hoc Network, MANET) – сети, состоящие из мобильных устройств. Каждое такое устройство может независимо передвигаться в любых направлениях, и, как следствие, часто разрывать и устанавливать соединения с соседями. Одной из основных проблем при построении MANET является обеспечения безопасности передаваемых данных. Для решения этой проблемы может применяться гомоморфное шифрование [**32**–**36**].

Гомоморфная схема шифрования может быть встроена в протоколы маршрутизации для повышения безопасности. С использованием гомоморфного шифрования операции могут быть выполнены промежуточными узлами над шифртекстами, что повышает безопасность протокола поскольку промежуточные узлы-нарушители не смогут определить открытого текста.

Первая задача, решаемая промежуточными узлами состоит в нахождении кратчайшего пути между двумя узлами. Расстояние между двумя соседними узлами известно.

Алгоритм нахождения оптимального пути включает операции Для нахождения оптимального пути гомоморфно необходимо применить динамическое программирование над зашифрованными данными [37]. Для этого необходимо уметь складывать, перемножать и сравнивать шифртексты чисел без их расшифрования.

Сравнение двух элементов можно представить в виде логической схемы *компаратора.*

Рассмотрим два n-битных числа A и B в двоичном представлении:

$$A = A_{n-1}...A_1A_0$$

$$B = B_{n-1}...B_1B_0$$

Эти числа будут равны в том и только том случае, когда все соответствующие разряды равны, т.е. для *всех* i выполняется $A_i = B_i$. Поскольку все цифры разрядов – двоичные, то булева функция для проверки равенства двух разрядов A_i и B_i может быть выражена следующим образом:

$$x_i = A_i \cdot B_i \oplus (A_i \oplus 1) \cdot (B_i \oplus 1)$$

где x_i равно 1 только в том случае, если $A_i = B_i$. В таком случае условие равенства A_i и B_i может быть выражено с помощью следующей булевой формулы:

$$(A = B) := x_{n-1} \cdot ... \cdot x_1 \cdot x_0$$

При вычислении неравенства, если $_i = B_i$ $\forall i = 0, ..., k-1$, при этом $_k = 1$ и $B = 0$, то мы приходим к выводу, что A$>$ B.

Это последовательное сравнение может быть выражено логически через булевы полиномы так:

$$(A > B) = A_{n-1} \cdot \overline{B_{n-1}} + x_{n-1}A_{n-2}\overline{B_{n-2}} + ... + x_{n-1}...x_2A_1\overline{B_1} + x_{n-1}...x_1A_0\overline{B_0}$$

$$(A < B) = \overline{A_{n-1}} \cdot B_{n-1} + x_{n-1}\overline{A_{n-2}}B_{n-2} + \ldots + x_{n-1}\ldots x_2\overline{A_1}B_1 + x_{n-1}\ldots x_1\overline{A_0}B_0$$

где $(A > B)$ и $(A < B)$ – это двоичные переменные результата, которые равны 1 когда A>B или A<B соответственно.

При этом результат возвращается в зашифрованном виде, что позволяет избежать проблемы, описанной в [38].

4. Криптографическая защита смарт-карт

В будущем смарт-карты больше не будет предназначены для устройств, обслуживающих единственное целевое приложение. Вместо этого наблюдается тенденция к проектированию универсальных карт с собственной операционной системой, которая может выполнять разнообразные функции [39]; таким образом, единственная индивидуальная карта будет иметь возможность взаимодействовать с несколькими поставщиками услуг (используя данные, сохраненные на карте).

Для того чтобы карта была в состоянии справиться с несколькими приложениями, было высказано предположение, что карта должна содержать данные владельца и что операционная система карты должна импортировать с сервера код функций (методов), которые будут выполняться над данными [40]. В статье [41] предлагается дополнительное решение для ослабления требования того, что все конфиденциальные данные и обработка будет происходить на карте. Идея состоит в том, что некоторые приложения могут работать вне карты на гомоморфно зашифрованных данных. Для таких приложений, карта ведут себя не просто как пассивное устройство, поскольку операционная система карты отвечает за шифрование и расшифрование данных, а также за управление доступом внешних приложений к данным (см раздел 3). Запуск некоторых приложения полностью вне карты имеет следующие преимущества:

- Формирует многопроцессорную среду, образованную из процессора карты и одного или более внешних процессоров. Таким образом, обеспечивается истинный параллелизм, хотя есть некоторая асимметрия когда внешний процессор пытается получить доступ к данным, хранимым в карте: внешние процессоры должны взаимодействовать с процессором карты в соответствии с нижеописанным Протоколом 2.

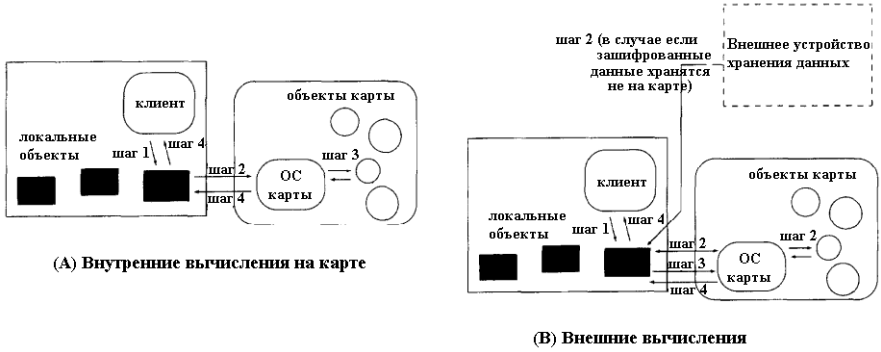

Рис. 3. Внешние и внутренние вычисления на смарт-карте

- Особенно ресурсоемкие приложения могут воспользоваться внешними устройствами хранения и внешними процессорами более мощными, чем на карте. Любое приложение, оперирующее с данными, сохраненными в карте может извлечь выгоду из подхода, представленного здесь. Это включает в себя приложения сервис-провайдеров а также приложения карт-инициатором, такие как биометрическая проверка (распознавание голоса, отпечатков пальцев или почерка), которые, как правило, требуется значительного объема хранения и большое количество сравнительно простых операций.

Приведем далее для сравнения два протокола работы универсальной смарт-карты: один основанный на традиционных вычислениях внутри карты, а другой, на использовании делегирования вычисления с помощью гомоморфного шифрования.

Протокол 1 (Вычисления внутри карты)

(1) Программа-клиент указывает на локальный объект (также называемый *прокси* или *суррогатный объект*).
(2) Методы прокси-объекта клиента выполняют процедурные вызовы методов в объекте карты. При принятии процедурного вызова методом карты, прокси-объект предоставляет карте сертифицированный код, соответствующий вызываемому методу.
(3) После проверки целостности сертификата метода, операционная карта система выполняет код метода над данными объектов карты.
(4) Операционная система карты возвращает ответ программе-клиенту.

Проблема с подходом, описанным в Протоколе 1 (объектно-ориентированный или агент на основе), заключается в том, что смарт-карте приходится самостоятельно, в конечном счете хранить и обрабатывать все конфиденциальные данные. Таким образом, ограниченное пространство для хранения и мощность обработки карты может быть узким местом когда будет обслуживаться очень ресурсоемкое приложение клиента или просто, когда несколько приложений клиента будут работать параллельно. Для таких приложений, мы предлагаем следующий Протокол 2, изображенный на Рисунке 3 (B), который предполагает, что данные объекта карты могут быть обработаны за пределами карты в зашифрованном виде.

Протокол 2 (Внешние вычисления)

(1) Программа-клиент указывает на локальный объект.
(2) Методы локального объекта клиента запрашивают с карты зашифрованную версию данных объекта карты. Карта предоставляет необходимые данные после надлежащего контроля безопасности.
(3) Методы локального объекта клиента проводят вычисления над зашифрованными данными, находят нужный (в зашифрованном виде) результат, и отправляют его в операционную систему карты.
(4) Операционная система карты запускает метод в объекте карты, который расшифровывает результат, полученный от клиента; расшифрованный ответ затем возвращается программе-клиенту.

Протокол 2 является дополнением к Протоколу 1. Таким образом, изменения должны лишь в небольшой степени влиять на жизненный цикл карты как он понимается в Протоколе 1. Объект карты o используемый в обоих протоколах имеет следующую структуру:

$$o = (\{d_i\}, \{I_i\}, \{[E^i, D^i]\}, \{A_i\}),$$

где $\{d_i\}$ – поля данных (открытые или зашифрованные), $\{I_i\}$ представляет собой набор методов интерфейсов, используется в протоколе 1, $\{[E', D']\}$ представляет собой набор алгоритмов шифрования/расшифрования ПГШ, имеющихся для этого объекта, и $\{A_i\}$ представляет собой набор методов контроля доступа, которые будут использоваться на шагах 2 и 4 Пртокола 2.

Методы шифрования, расшифрования и управления доступом являются полными методами, то есть, они содержат свои реализации. В случае если o используется только с Протоколом 1, очевидно

потребуются лишь $\{d_i\}$ и $\{I_i\}$. И наоборот, если o должен быть использован только с протоколом 2, то $\{I_i\}$ не требуется.

И как мы видим, для этого приложения вполне достаточно симметричного полностью гомоморфного шифрования.

5. Машинное обучение на зашифрованных данных

Впервые идея применения алгоритмов машинного обучения к зашифрованным данным возникла, по видимому, в [42], получила развитие в [43] однако практически реализована только в [44] в виде программной библиотеки.

Классификаторы являются бесценным инструментом во многих местах сегодня, например, медицинских или геномных предсказаниях, обнаружении спама, распознавании лиц, и финансовых вычислениях. Многие из этих приложений обрабатывают конфиденциальные данные [45–47], поэтому важно чтобы данные и классификатор оставались в секрете.

Рассмотрим типичную ситуацию обучения с учителем, изображенную на рисунке 4. Алгоритмы обучения с учителем состоят из двух фаз:

(1) фазы обучения, в течение которого алгоритм строит модель w по набору пар «объект, ответ», называемом обучающей выборкой (или прецедентами), и
(2) фазы распознавания на которой классификатор C запускается над ранее неизвестным вектором признаков x, используя модель w для вывода ответа-результата $C(x, w)$.

В приложениях, обрабатывающих конфиденциальные данные, важно, чтобы вектор признаков x и модель w оставались в секрете для одного или нескольких участников. Рассмотрим пример медицинского исследования или больницы, имеющей модель, построенную с помощью конфиденциальных медицинских данных некоторых пациентов; модель конфиденциальна, поскольку содержит информацию о пациентах, и её использование должно быть совместимым с ФЗ «О защите персональных данных»[1]. Клиент хочет использовать модель, чтобы сделать прогноз о своем здоровье (например, насколько вероятно быстрое излечение, или, будет ли успешным лечение в стационаре), но не хочет раскрывать свои конфиденциальные медицинские параметры. В идеале, больница и клиент запускают протокол после

[1] закон № 152-ФЗ «О персональных данных»

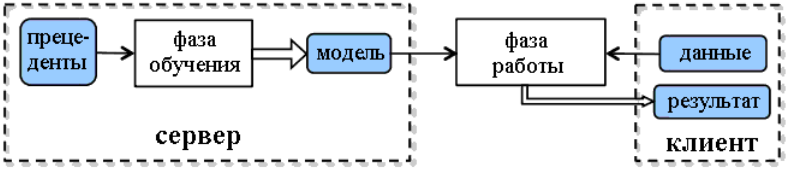

Рис. 4. Общая модель. Каждый заштрихованный квадрат указывает секретные данные, которые должны быть доступны только одному участнику: прецеденты и модель серверу, а данные и результат – клиенту. Каждый прямой без пунктирный прямоугольник указывает алгоритм, одиночные стрелки указывают входные данные алгоритма, а двойные стрелки указывают результаты.

которого клиент узнает один бит («да / нет»), и ни одна из сторон не узнает что-нибудь еще о исходных данных другой. Подобная же ситуация возникает в финансовом учреждении (например, страховой компании), которая держит в секрете модель, а заказчик хочет оценить цены или качество обслуживания на основе своей личной информации.

5.1. Гомоморфная обработка человеческой речи

Одной из наиболее заманчивых (почти фантастичных) возможностей эффективной гомоморфной обработки данных является гомоморфная обработка человеческой речи. Впервые такая возможность была предложена в [48] но пока она так и остается нереализованной.

Реализация возможности гомоморфнорй обработки речи позволит осуществить проект полностью роботизированного call-центра, который будет полностью осуществлять свою работу так что никакая информация гарантированно не будет появляться в открытом виде. Однако, для обработки речи гомоморфно в реальном времени необходимо существенно улучшить производительность как существующих алгоритмов гомоморфного шифрования, так и алгоритмов машинного машинного обучения.

6. Заключение

Отрасль гомоморфного шифрования богата как глубокими теоретическими результатами так и востребованными практикой приложениями. Как мы увидели, компактное симметричное полностью

гомоморфное, криптостойкое против атаки по известным открытым текстам подходит для большинства из них. В ближайшем будущем инструменты гомоморфной криптографии окажут свое влияние на рынок облачных услуг, и в той или иной степени на облик современных информационных технологий.

Список литературы

[1] R. L Rivest, L. Adleman, M. L Dertouzos. *On data banks and privacy homomorphisms* // Foundations of secure computation, 1978. Vol. **32**, no. 4, p. 169–178. ↑1

[2] S. Fau, R. Sirdey, C. Fontaine, C. Aguilar-Melchor, G. Gogniat. *Towards practical program execution over fully homomorphic encryption schemes* // P2p, parallel, grid, cloud and internet computing (3pgcic), 2013 eighth international conference on IEEE, 2013, p. 284–290. ↑2

[3] D. K. Rappe. Homomorphic cryptosystems and their applications, 2006. http://eprint.iacr.org/. ↑3

[4] C. Fontaine, F. Galand. *A survey of homomorphic encryption for nonspecialists* // EURASIP Journal on Information Security, 2007. Vol. **2007**. ↑3

[5] Н. Варновский, А. Шокуров. *Гомоморфное шифрование* // Российская Академия наук Институт Системного Программирования, 2006, с. 27. ↑3

[6] C. Gentry. *A fully homomorphic encryption scheme*, Ph.D. Thesis, (2009). ↑3

[7] N. P. Smart, F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes, 2009. http://eprint.iacr.org/. ↑3

[8] M. Van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan. *Fully homomorphic encryption over the integers* // Advances in cryptology–eurocrypt 2010: Springer, 2010, p. 24–43. ↑3

[9] N. P Smart, F. Vercauteren. *Fully homomorphic encryption with relatively small key and ciphertext sizes* // Public key cryptography–pkc 2010: Springer, 2010, p. 420–443. ↑3

[10] V. Vaikuntanathan. *Computing blindfolded: New developments in fully homomorphic encryption* // Foundations of computer science (focs), 2011 ieee 52nd annual symposium on IEEE, 2011, p. 5–16. ↑3

[11] P. V Parmar, S. B Padhar, S. N Patel, N. I Bhatt, R. H Jhaveri. *Survey of various homomorphic encryption algorithms and schemes* // International Journal of Computer Applications, 2014. Vol. **91**, no. 8, p. 26–32. ↑3

[12] R. L Rivest, A. Shamir, L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems* // Communications of the ACM, 1978. Vol. **21**, no. 2, p. 120–126. ↑3

[13] L. L Babenko, A. V Trepacheva. *Known plaintexts attack on polynomial based homomorphic encryption* // Proceedings of the 7nd international conference on security of information and networks ACM, 2014, p. 67–70. ↑3

[14] А. О. Жиров, О. В. Жирова, С. Ф. Кренделев. *Безопасные облачные вычисления с помощью гомоморфной криптографии* // журнал БИТ (безопасность информационных технологий), 2013. Т. **1**, с. 6–12 (russian). ↑3

[15] А. Ростовцев, А. Богданов, М. Михайлов. *МЕТОД БЕЗОПАСНОГО ВЫЧИСЛЕНИЯ ПОЛИНОМА В НЕДОВЕРЕННОЙ СРЕДЕ С ПОМОЩЬЮ ГОМОМОРФИЗМОВ КОЛЕЦ* // Проблемы информационной безопасности. Компьютерные системы, 2011. Т. **2**, с. 76–85 (russian). ↑3

[16] P. B Burtyka, O. B Makarevich. *Symmetric fully homomorphic encryption using decidable matrix equations* // Proceedings of the 7nd international conference on security of information and networks ACM, 2014, p. 67–70. ↑3

[17] Ф Б Буртыка. *СИММЕТРИЧНОЕ ПОЛНОСТЬЮ ГОМОМОРФНОЕ ШИФРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ НЕПРИВОДИМЫХ МАТРИЧНЫХ ПОЛИНОМОВ* // Известия ЮФУ. Технические науки, 2014. Т. **158**, № 9, с. 107–122. ↑3

[18] D. Boneh, C. Gentry, S. Halevi, F. Wang, D. J Wu. *Private database queries using somewhat homomorphic encryption* // Applied cryptography and network security Springer, 2013, p. 102–118. ↑6

[19] W. Gasarch. *A survey on private information retrieval* // Bulletin of the eatcs Citeseer, 2004. ↑8

[20] B. Chor, E. Kushilevitz, O. Goldreich, M. Sudan. *Private information retrieval* // Journal of the ACM (JACM), 1998. Vol. **45**, no. 6, p. 965–981. ↑8

[21] S. Yekhanin. *Private information retrieval* // Communications of the ACM, 2010. Vol. **53**, no. 4, p. 68–73. ↑8

[22] R. Ostrovsky, W. E Skeith III. *A survey of single-database private information retrieval: Techniques and applications* // Public key cryptography–pkc 2007: Springer, 2007, p. 393–411. ↑8

[23] D. Woodruff, S. Yekhanin. *A geometric approach to information-theoretic private information retrieval* // Computational complexity, 2005. proceedings. twentieth annual ieee conference on IEEE, 2005, p. 275–284. ↑8

[24] Y. Gertner, Y. Ishai, E. Kushilevitz, T. Malkin. *Protecting data privacy in private information retrieval schemes* // Proceedings of the thirtieth annual acm symposium on theory of computing ACM, 1998, p. 151–160. ↑8

[25] Z. Brakerski, V. Vaikuntanathan. *Efficient fully homomorphic encryption from (standard) lwe* // Foundations of computer science (focs), 2011 ieee 52nd annual symposium on IEEE, 2011, p. 97–106. ↑8

[26] X. Yi, M. G. Kaosar, R. Paulet, E. Bertino. *Single-database private information retrieval from fully homomorphic encryption* // Knowledge and Data Engineering, IEEE Transactions on, 2013. Vol. **25**, no. 5, p. 1125–1134. ↑8

[27] О. Лупанов. *Об асимптотических оценках сложности управляющих систем*, 1984 (russian), http://www.inf.u-szeged.hu/actacybernetica/edb/vol04n4/pdf/Lupanov_2_1980_ActaCybernetica.pdf. ↑8

[28] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang. *On the (im) possibility of obfuscating programs* // Journal of the ACM (JACM), 2012. Vol. **59**, no. 2, p. 6. ↑9, 11, 13

[29] T. Sander, C. F Tschudin. *Protecting mobile agents against malicious hosts* // Mobile agents and security: Springer, 1998, p. 44–60. ↑10

[30] В. Иванников, Н. Варновский, В. Захаров, Н. Кузюрин, А. Шокуров, А. Кононов, А. Калинин. *Методы информационной защиты проектных решений при изготовлении микроэлектронных схем* // Известия ТРТУ, 2005. Т. **4**, с. 112–119. ↑13

[31] Ш. Курмангалеев, В. САВЧЕНКО, В. КОРЧАГИН. *О методах деобфускации программ* // Труды Института системного программирования РАН, 2013. Т. **24**. ↑13

[32] G. V. S. Rao, G. Uma. *An efficient secure message transmission in mobile ad hoc networks using enhanced homomorphic encryption scheme* // GJCST-E: Network, Web & Security, 2013. Vol. **13**, no. 9. ↑14

[33] G. V. Rao, V. Kakulapati, M. Purushoththaman. *Privacy homomorphism in mobile ad hoc networks.* // International Journal of Research & Reviews in Computer Science, 2011. Vol. **2**, no. 1. ↑14

[34] L Ertaul, W Lu. *Ecc based threshold cryptography for secure data forwarding and secure key exchange in mobile ad hoc networks (manet) i* // Proc. of networking 2005 international conference, 2005. ↑14

[35] E Vaidehi. *Computing aggregation function minimum/maximum using homomorphic encryption schemes in wireless sensor networks (wsns)* // California State University, East Bay Hayward, CA, USA, 2007. ↑14

[36] V. Levent Ertaul. *Implementation of homomorphic encryption schemes for secure packet forwarding in mobile ad hoc networks (manets).* ↑14

[37] M. Yokoo, K. Suzuki. *Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions* // Proceedings of the first international joint conference on autonomous agents and multiagent systems: part 1 ACM, 2002, p. 112–119. ↑15

[38] М. ХРАПЧЕНКО, С. МАРТИШИН, В. НП. *МЕТОДЫ ПОРОГОВОЙ КРИПТОГРАФИИ ДЛЯ ЗАЩИТЫ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ* // Труды Института системного программирования РАН, 2014. Т. **26**, № 2. ↑16

[39] L. C. Guillou, M. Ugon, J.-J. Quisquater. *The smart card: A standardized security device dedicated to public cryptology* // Contemporary cryptology: The science of information integrity. IEEE Press, Piscataway, NJ, 1992. ↑16

[40] A Gamache, P. Paradinas, J.-J. Vandewalle. *World-wide smart card services (cardis'94)*, 1994, p. 141–148, (Lille,France,1994). ↑16

[41] J. Domingo-Ferrer. *Multi-application smart cards and encrypted data, processing* // Future Generation Computer Systems, 1997. Vol. **13**, no. 1, p. 65–74. ↑16

[42] S. Laur, H. Lipmaa, T. Mielikäinen. *Cryptographically private support vector machines* // Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining ACM, 2006, p. 618–624. ↑19

[43] T. Graepel, K. Lauter, M. Naehrig. *Ml confidential: Machine learning on encrypted data* // Information security and cryptology–icisc 2012: Springer, 2013, p. 1–21. ↑19

[44] R. Bost, R. A. Popa, S. Tu, S. Goldwasser. Machine learning classification over encrypted data, 2014. http://eprint.iacr.org/. ↑19

[45] J. Wiens, J. Guttag, E. Horvitz. *Learning evolving patient risk processes for c. diff colonization* // Icml workshop on machine learning from clinical data, 2012. ↑19

[46] A. Singh, J. V Guttag. *A comparison of non-symmetric entropy-based classification trees and support vector machine for cardiovascular risk stratification* // Engineering in medicine and biology society, embc, 2011 annual international conference of the ieee IEEE, 2011, p. 79–82. ↑19

[47] A. Singh, G. Nadkarni, J. Guttag, E. Bottinger. *Leveraging hierarchy in medical codes for predictive modeling* // Proceedings of the 5th acm conference on bioinformatics, computational biology, and health informatics ACM, 2014, p. 96–103. ↑19

[48] M. A Pathak, B. Raj, S. Rane, P. Smaragdis. *Privacy preserving speech processing*, http://www.cs.illinois.edu/~paris/pubs/pathak-spm2013.pdf. ↑20

Об авторах:

Людмила Климентьевна Бабенко

Институт компьютерных технологий и информационной безопасности Южного федерального университета

e-mail: blk@tsure.ru

Филипп Борисович Буртыка

Институт компьютерных технологий и информационной безопасности Южного федерального университета

e-mail: bbfilipp@ya.ru

Олег Борисович Макаревич

Институт компьютерных технологий и информационной безопасности Южного федерального университета

e-mail: mak@tsure.ru

Алина Викторовна Трепачева

Институт компьютерных технологий и информационной безопасности Южного федерального университета

e-mail: atrepacheva@sfedu.ru

Образец ссылки на эту публикацию:

Л. К. Бабенко, Ф. Б. Буртыка, О. Б. Макаревич, А. В. Трепачева. *Защищенные вычисления и гомоморфное шифрование* // Программные системы: теория и приложения: электрон. научн. журн. 2014. Т. ??, № ?, с. ??–??. *URL:* http://psta.psiras.ru/read/

Ludmila Babenko, Filipp Burtyka, Oleg Makarevich, Alina Trepacheva. *Private Computations and Homomorphic Encryption.*

Abstract. Resume in English will be added there.

Key Words and Phrases: Fully Homomorphic Encryption, Homomorphic Encryption, Obfuscation, Cloud Computations, Smart Cards, Mobile Ad-Hoc networks, Machine Learning.