(19) Евразийское патентное ведомство

(11) 036613 (13) B1

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ЕВРАЗИЙСКОМУ ПАТЕНТУ

(45) Дата публикации и выдачи патента
2020.11.30

(21) Номер заявки
201990063

(22) Дата подачи заявки
2016.06.22

(51) Int. Cl. *G06F 21/62* (2013.01)

(54) ДВУХРЕЖИМНАЯ СХЕМА ШИФРОВАНИЯ, ОБЕСПЕЧИВАЮЩАЯ СРАВНЕНИЕ, ОСНОВАННОЕ НА ИНДЕКСИРОВАНИИ

(43) **2019.06.28**
(86) **PCT/RU2016/000382**
(87) **WO 2017/222407 2017.12.28**

(71)(73) Заявитель и патентовладелец:
АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ ВЫСШЕГО ОБРАЗОВАНИЯ "СКОЛКОВСКИЙ ИНСТИТУТ НАУКИ И ТЕХНОЛОГИЙ" (RU)

(72) Изобретатель:
Панагиотис Каррас, Никитин Артем Антонович (RU)

(74) Представитель:
Фелицына С.Б. (RU)

(56) Panagiotis Karras et al.: "Adaptive Indexing over Encrypted Numeric Data", Proceedings of the 2016 International Conference on Management of Data, SIGMOD '16, 18 April 2016 (2016-04-18), pages 171-183, XP055351121, New York, New York, USA DOI: 10.1145/2882903.2882932 ISBN: 978-1-4503-3531-7 Retrieved from the Internet: URL: https://web.archive.org/web/20160418003601/http://stratos.seas.harvard.edu/files/stratos/files/securadapt.pdf?m=1455415195 [retrieved on 2017-03-02] last paragraph; page 4, left-hand column equations (5)-(6), Section 3.3; page 5, left-hand column, Section 4; page 6, left-hand column, Section 4.3; page 7, right-hand column, page 5, right-hand column last paragraph; page 8, left-hand column; figure 5 page 4, left-hand column, Section 3.3; page 5, Section 4.3; page 7 abstract

BIJIT HORE ET AL.: "A Privacy-Preserving Index for Range Queries", PROCEEDINGS OF THE THIRTIETH INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 1 January 2004 (2004-01-01), pages 720-731, XP009133028, ISBN: 978-0-12-088469-8 Abstract page 720 - page 721

ELAINE SHI ET AL.: "Multi-Dimensional Range Query over Encrypted Data", SECURITY AND PRIVACY, 2007. SP '07. IEEE SYMPOSIUM ON, 20 May 2007 (2007-05-20), pages 350-364, XP055329444, Pi DOI: 10.1109/SP.2007.29 ISBN: 978-0-7695-2848-9 abstract page 1 - page 2

(57) Раскрытие направлено на схему непрозрачного шифрования, которая обеспечивает обработку запросов диапазона над зашифрованными числовыми данными, полученными из внешних источников в облаке или на удаленном сервере, и тем самым инкрементное адаптивное индексирование, в результате чего индексируются только те данные, которые запрашиваются доверенными клиентами. В одном варианте осуществления предложен способ индексирования значений данных. Способ содержит этапы, на которых выполняют шифрование множества значений данных с использованием первого режима шифрования для получения зашифрованных значений данных; выполняют шифрование одного или более связанных значений с использованием первого режима шифрования для получения первых зашифрованных связанных значений и с использованием второго режима шифрования для получения вторых зашифрованных связанных значений, причем второй режим шифрования отличается от первого режима шифрования; сравнивают указанное одно или более связанных значений с множеством значений данных с использованием вторых зашифрованных связанных значений и зашифрованных значений данных; и индексируют зашифрованные значения данных на основании упомянутого сравнения с использованием первых зашифрованных связанных значений в качестве значений ключа.

036613 B1

036613 B1

Область техники, к которой относится изобретение

Настоящее раскрытие относится, в общем, к адаптивному индексированию зашифрованных числовых данных и, в частности, к адаптивному индексированию зашифрованных числовых данных без раскрытия общего порядка числовых данных.

Уровень техники

Введение.

Человечество вступило в эру "базы данных как услуги", поэтому необязательно, чтобы полномочия на управление данными были представлены в региональной настройке владельца данных, но они могут быть представлены поставщиком облачных услуг в качестве услуги, в которой присутствуют данные, полученные из внешних источников. Эта модель предоставляет пользователям (т.е. владельцу данных и его доверенным клиентам) возможность создавать, хранить, изменять и извлекать данные из любой точки мира, причем в дискуссию относительно применимости этой модели вошли такие области, как высокочастотная торговля, в результате чего фирма может использовать серверы поставщиков облачных услуг для тестирования торговых стратегий, выполнения анализа временных рядов, оценки рисков и даже выполнения торговых операций, при этом осуществляя сбор финансовых данных ежедневно или в более мелком временном масштабе. В то же время такая модель вызывает обеспокоенность в отношении безопасности и конфиденциальности; чтобы следовать той же области применения, фирма может развернуть на облаке торговые, аналитические модули и модули управления рисками с тем, чтобы отфильтровывать наиболее релевантные финансовые данные для внутреннего анализа. Таким образом, конфиденциальные данные и результаты запросов могут быть переданы по каналам утечки злонамеренным злоумышленникам и/или непосредственно честному, но любопытному поставщику услуг. Такие проблемы мотивировали исследования по шифрованию данных и реакцию на запросы, касающиеся зашифрованных данных, начиная с обработки SQL-запросов в отношении зашифрованных данных, и заканчивая разработкой индивидуальных схем, которые позволяют обрабатывать специализированные запросы, касающиеся зашифрованных данных, например запросы kNN и skyline. В CryptDB представлен взаимосвязанный набор эффективных схем шифрования с поддержкой языка структурированных запросов (SQL), которые позволяют выполнять SQL-запросы, касающиеся зашифрованных данных.

Во всех случаях цель состоит в том, чтобы сервер мог управлять данными с минимальным вмешательством владельца данных и клиентов. Сервер должен иметь возможность обрабатывать запросы, касающиеся зашифрованных данных, и доставлять зашифрованные результаты запроса клиенту, в то время как клиент должен только расшифровать данные и, таким образом, получить фактические результаты. Тем не менее, для того чтобы система могла эффективно управлять данными и обрабатывать запросы, она должна, прежде всего, иметь возможность индексировать данные в той степени, в которой это необходимо. В дополнение к этому, современные приложения, обрабатывающие непрерывно поступающие данные, обладают присущей им способностью выполнять инкрементное индексирование по требованию при обработке пользовательских запросов и в качестве побочного эффекта от таких запросов не требуют ни априорного времени простоя, ни априорных знаний о рабочей нагрузке; другими словами, система должна быть не только способной к индексированию; она должна быть способной к адаптивному индексированию и самоорганизации.

Эти требования к управлению данными в эпоху "базы данных как услуги" порождают противоречие. Требование, чтобы данные оставались конфиденциальными, приводит к тому, чтобы они были зашифрованными; требование к эффективному управлению данными приводит к их индексированию. Шифрование и индексирование находятся во внутреннем противоречии друг с другом; шифрование требует, чтобы поставщик услуг ничего не знал о значениях данных, в то время как индексирование требует, чтобы тот же поставщик услуг знал, в идеале, точные значения данных, находящиеся в его компетенции. Учитывая такие противоречивые цели, предыдущие исследования столкнулись с этими двумя проблемами отдельно друг от друга, и в них не было предпринято никаких усилий, чтобы совместно противостоять им. Решения для адаптивного индексирования и самоорганизации не учитывают безопасность и конфиденциальность, в то время как защищенные системы баз данных не предусматривают адаптивность в динамических средах. К сожалению, избегая такой цели, такие подходы предлагают фрагментарные решения проблемы безопасного и эффективного управления данными, извлекаемыми из внешних источников; в частности, существующие схемы шифрования страдают от одного или более следующих недостатков: (1) они являются слишком дорогими с точки зрения вычислений; или (2) утечка слишком большого количества информации о зашифрованных данных, и/или (3) требуется больше данных, чем фактических результатов запроса, которые необходимо извлечь и затем отфильтровать на этапе постобработки.

В данном документе предложена модель, которая позволяет преодолеть эту обособленность и достичь, казалось бы, противоречивой цели: обеспечить системы баз данных, которые были бы одновременно защищенными и адаптивными. В данном документе рассмотрена проблема адаптивного индексирования зашифрованной базы данных. Чтобы добиться такого результата, исследована степень, в которой шифрование и индексирование находятся во внутреннем противоречии друг с другом: изучены основные требования к индексированию и способы, с помощью которых могут быть удовлетворены такие требования, связанные с шифрованием. В данном документе предложена упрощенная и эффективная схема

шифрования, которая открывает меньше информации, чем предыдущие предложения, и позволяет адаптивно индексировать зашифрованные числовые данные. Эта способность индексирования направлена не на разработку полного индекса заранее, а на его построение в пошаговой постепенной манере таким образом, чтобы индексировались только те данные, которые запрашиваются. Такое адаптивное индексирование позволяет облегчить адаптацию системы к имеющейся рабочей нагрузке, а также уменьшить напряжение между шифрованием и индексированием. Предложенная схема базируется на простых операциях линейной алгебры для шифрования и дешифрования, при этом она позволяет эффективно обрабатывать запросы диапазонов и точек по шифротекстам, не раскрывая порядок среди значений атрибутов, как схема шифрования с сохранением порядка, а также без затрат на вычисления и хранение для схем, таких как схемы полностью гомоморфного шифрования.

Ниже приводится обзор предыдущих работ, касающихся систем данных, которые поддерживают запросы к зашифрованным данным, а также адаптивного индексирования.

Обработка зашифрованных данных.

Исследование в области обработки зашифрованных данных можно классифицировать по двум широким классам. Прежний класс решений предполагает обработку непосредственно зашифрованных данных. В работе Х. Хасигумуса и др. (H. Hacigumus., B.R. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In SIGMOD, pages 216-227, 2002) предложена схема группирования данных, которая обеспечивает приблизительную фильтрацию результатов запроса на сервере с последующей окончательной обработкой в клиенте после дешифрования. В работе Б. Хора и др. (B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In VLDB, pages 720-731, 2004) подробно рассмотрена эта схема с индексом, который поддерживает запутанные запросы диапазона, расширенные до многомерного случая В работе Б. Хора и др. (B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In VLDB, pages 720-731, 2004) подробно останавливаются на этой схеме с индексом, который поддерживает запутанные запросы диапазона, расширенные до многомерного случая (B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. VLDB Journal, 21(3):333-358, 2012). Однако такие схемы раскрывают распределение данных и вовлекают клиента в обработку запроса, в то время как единственная способность индексирования, которой они обладают, заключается в группировании данных.

Попытка детального индексирования потребует сортировки значений на сервере. Такой альтернативой является схема шифрования с сохранением порядка (OPES), предложенная Р. Агравалом и др. (R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In SIGMOD, 2004), которая построена на кодировании, которое сохраняет порядок числовых данных. Однако, как было отмечено в предыдущих исследованиях, OPES раскрывает порядок данных и, следовательно, не может побороть атаки, основанные на статистическом анализе зашифрованных данных. Можно утверждать, что OPES предоставляет решение с избыточным количеством ресурсов: эта схема представляет зашифрованные значения в виде, поддающемся сортировке, следовательно, позволяет сравнивать такие значения друг с другом.

Между тем, успехи в криптографии привели к способности выполнять произвольные вычисления над зашифрованными данными с целью получения правильного зашифрованного результата с помощью полностью гомоморфного шифрования (FHE) (C. Gentry. Fully homomorphic encryption using ideal lattices. In STOC, pages 169-178, 2009). Такие вычисления базируются на дорогой криптосистеме с открытым ключом, что приводит к непомерным затратам до девяти порядков величины. Кроме того, FHE не позволяет индексировать зашифрованные данные. При использовании FHE, даже если сервер выполняет все вычисления правильно, на практике он делает это только в зашифрованном виде. У него нет доступа к какой-либо истине в реальном мире, которая позволила бы построить индекс. В работе М. Мани и др. (M. Mani, K. Shah, and M. Gunda. Enabling secure database as a service using fully homomorphic encryption: Challenges and opportunities. CoRR, abs/1302.2654, 2013) обсуждены возможности FHE для обеспечения видения защищенной базы данных как услуги и сделаны выводы относительно того, что практичность остается очень важной задачей. Чтобы обработать простой запрос выбора с помощью двухэтапного процесса, сервер сначала вычисляет результаты запроса и отправляет клиенту зашифрованное число n кортежей результатов; клиент дешифрует n и запрашивает $n' \geq n$ строк из сервера. В результате сервер возвращает верхние -n' строк; таким образом, клиент должен активно участвовать в простых задачах обработки запросов.

В работе В.Х. Ванга и Л.В.С. Лакшманана (W.H. Wang and L.V.S. Lakshmanan. Efficient secure query evaluation over encrypted XML databases. In VLDB, pages 127-138, 2006) предложена схема для оценки запросов защищенным образом относительно зашифрованных данных в формате XML. Они рассматривают расширение схемы OPES до шифрования с сохранением порядка, разбиением и масштабированием (OPESS), при котором каждый открытый текст "разбивается" на большее количество шифротекстов, а разделенные данные "масштабируются", так что злоумышленник не может определить идентичность шифротекстов на основе знаний относительно периодичности данных. Однако эта схема не может поддерживать обновления; таким образом, она не подходит для динамической и адаптивной системы баз

данных.

В работе Е. Ши и др. (E. Shi, J. Bethencourt, H.T.-H. Chan, D.X. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In IEEE Symposium on Security and Privacy, pages 350-364, 2007) предложена схема шифрования для ответов на запросы многомерного диапазона. Еще одна проблема, с которой они сталкиваются, состоит в том, чтобы позволить аудитору дешифровать те и только те записи (например, журналы финансового аудита, медицинские истории болезни и т.д.), чьи атрибуты попадают в указанный диапазон; конфиденциальность не защищена, когда запись соответствует запросам. К сожалению, эта схема выявления совпадений не обеспечивает защиту значения атрибутов, в то же время позволяя поставщику услуг идентифицировать записи, соответствующие запросам относительно зашифрованных данных таким образом, чтобы их можно было использовать для индексирования.

В работе Д. Бонеха и Б. Уотерса (D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In TCC, pages 535-554, 2007) рассмотрена обработка сложных запросов, касающихся зашифрованных данных, на основе шифрования скрытого вектора (HVE). Понятие безопасности в этой работе сильнее, чем в работе Ши и др.: записи, соответствующие запросу, идентифицируются, но их значения атрибутов остаются скрытыми. Это понятие безопасности со скрытием совпадений соответствует требованиям индексирования при защите конфиденциальности значений атрибута. Однако предложенная схема не осуществима для огромных объемов динамических данных; она влечет за собой увеличения размера открытого ключа O(DT), времени шифрования и размера шифротекста для D-атрибутов и T-дискретных значений в расчете на каждый атрибут.

Ту и др. разработали систему MONOMI (S. Tu, M.F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. PVLDB, 6(5):289-300, 2013), которая может выполнять аналитические запросы к зашифрованным данным. Основываясь на системе CryptDB, система MONOMI использует несколько технологий с целью повышения производительности совместно с разработчиком, который выбирает эффективное физическое проектирование на стороне сервера, и планировщиком, который выбирает эффективные планы выполнения запросов с участием сервера и клиента. Однако поскольку система MONOMI обеспечивает индексирование, основанное на порядке, она делает это опираясь на OPES. Позже Р. Ли и др. (R. Li, A.X. Liu, A.L. Wang, and B. Bruhadeshwar. Fast range query processing with strong privacy protection for cloud computing. PVLDB, 7(14): 1953-1964, 2014) предложили схему обработки запросов диапазона, касающихся зашифрованных данных, в которой проверки неравенства проводятся посредством исчерпывающих проверок равенства; при этом не предложен механизм проверки чистого неравенства в отношении зашифрованных данных.

В целом, эти подходы не обеспечивают хорошего компромисса между конфиденциальностью и эффективностью. В последнее время в некоторых научно-исследовательских работах были представлены схемы шифрования, разработанные с учетом конкретных особенностей, которые обеспечивают обработку специализированных запросов, касающихся зашифрованных данных, таких как запросы kNN, запросы поиска диапазона и горизонта.

Альтернативой обработке зашифрованных данных напрямую является поддержание зашифрованного индекса на сервере, привлечение клиента для обхода этого индекса и определение точного местоположения данных, представляющих интерес, после нескольких итераций извлечения и дешифрования. В работе Дамиани и др. (E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In ACM CCS, pages 93-102, 2003) построено B-дерево по значениям открытого текста, но зашифровали каждый кортеж и непосредственно B-дерево на уровне узла; при этом содержание дерева не видимо серверу. В работе Ге и Здоника (T. Ge and S.B. Zdonik. Fast, secure encryption for indexing in a column-oriented DBMS. In ICDE, pages 676-685, 2007) предложено шифрование с быстрым сравнением (FCE). Обход индексов с помощью FCE требует сравнения между значениями ключей открытого текста и шифротекста путем частичного дешифрования на стороне клиента. Ванг и др. (S. Wang, D. Agrawal, and A. El Abbadi. A comprehensive framework for secure query processing on relational data in the cloud. In Secure Data Management, pages 52-69, 2011) представили безопасную схему обработки запросов, которая защищает данные как в хранилище, так и при доступе, основываясь на алгоритме распространения информации с привязкой (Salted IDA). С помощью этой схемы клиент поддерживает секретную матрицу распространения информации и ключи для декодирования матрицы D данных; клиент кодирует и распространяет D на n серверах, при использовании генератора псевдослучайных чисел с секретным начальным числом для добавления случайной "привязки" к каждой записи данных; при этом после извлечения данных привязка восстанавливается и выводится из декодированных данных с целью восстановления D. Таким образом, обработка запросов направляется клиентом, в то время как серверы могут получать доступ только к данным, следуя инструкциям клиента. В целом, подходы, основанные на зашифрованном индексе, не позволяют серверу создавать, поддерживать и перемещать индекс, опираясь на свои собственные устройства.

Третье направление включает в себя обработку зашифрованных данных на защищенном аппаратном оборудовании, примером которого в последнее время является система Cipherbase (A. Arasu, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann, and R. Ramamurthy. Transaction processing on confidential data using Cipherbase. In ICDE, pages 435-446, 2015). Тем не менее, для индекса диапазона система Ci-

pherbase раскрывает полную информацию об упорядоченном расположении ключей индекса даже для слабого злоумышленника; таким образом, индексы диапазона в системе Cipherbase обеспечивают "аналогичные гарантии конфиденциальности" и имеют те же недостатки, что и шифрование с сохранением порядка (OPE).

Адаптивное индексирование.

Помимо требований безопасности системы данных для современных приложений должны быть быстрыми и гибкими, легко адаптирующимися к быстро изменяющимся требованиям. Адаптивное индексирование - это недавно введенная концепция, согласно которой нет необходимости выполнять настройку индекса во время инициализации системы; вместо этого оно происходит во время обработки запроса: каждый запрос интерпретируется не только как запрос для определенного набора результатов, но также как совет относительно того, как физически хранить данные, инициируя небольшие действия, которые улучшают адаптивные индексы. Те части данных, которые запрашиваются, постепенно, шаг за шагом индексируются. Эта способность индексировать путем непрерывного реагирования на изменяющуюся нагрузку запросов порождает свойство самоорганизации.

В этом разделе взлом базы данных описан более подробно. Взлом баз данных ввел понятие непрерывного, инкрементного и адаптивного индексирования по требованию в контексте современных хранилищ столбцов. При взломе оператор выбора системы базы данных выполняет операции построения индексов в качестве побочного эффекта обработки действия фильтрации диапазона (или равенства); индекс строится и уточняется параллельно с исполнением запроса: чем больше запрашивается диапазон данных, тем больше уточняется его индекс. Физическое хранилище данных изменяется при каждом входящем запросе q диапазона, интерпретируя q как подсказку относительно того, как следует хранить данные. Эта подсказка может явно использовать границы запроса q или следовать более слабой интерпретации ради надежности. Без потери общности обсуждена строгая интерпретация. Предположим, что выполняется запрос A<10. Взломанная СУБД реагирует на q путем кластеризации всех кортежей A при A<10 в начале соответствующего столбца C, в то же время проталкивая все другие кортежи в конец столбца. Последующему запросу, запрашивающему $A \geq \nu_1$, где $\nu_1 \geq 10$, нужно будет только просмотреть последний фрагмент C, в то время как запрос, который запрашивает $A < \nu_2$, где $\nu_2 \leq 10$, ищет свою первую часть; при этом каждый последующий запрос в дальнейшем взламывает свой соответствующий фрагмент. На фиг. 1 показано как два запроса взламывают свой столбец посредством своих предикатов выбора. Запрос Q1 разделяет столбец на три фрагмента, и Q2 усиливает это разбиение путем дополнительного разделения первого и последнего фрагментов. Каждый запрос собирает свои подходящие кортежи в непрерывной области. Таким образом, исходный столбец A (включая позиции) копируется в столбец индекса взломщика, который затем непрерывно реорганизуется.

По мере обработки запросов адаптивный индекс столбца непрерывно разделяется на большее количество (и таким образом, более мелкие) фрагментов. Поэтому существует потребность в структуре данных для локализации фрагмента, представляющего интерес. В последнее время литература по адаптивному индексированию базировалась на AVL-дереве внутренней памяти для отслеживания всех фрагментов, созданных в результате физической реорганизации. По мере постепенного индексирования столбца запросы, запрашивающие диапазоны, которые точно совпадают со значениями, известными по индексу, отвечают за стоимость поиска только этого древовидного индекса; во всех случаях, даже когда нет точного совпадения, индекс уменьшает количество данных, к которым должен обращаться запрос, так как каждый запрос диапазона реорганизует не более двух фрагментов на краю соответствующего диапазона.

Эта физическая реорганизация на лету выполняется для отдельных фрагментов (или целого столбца для самого первого запроса) с помощью встроенных алгоритмов, основанных на последовательных шаблонах доступа. Алгоритм 1 показывает самый основной алгоритм взлома, который разделяет фрагмент на две части, идентифицируя кортежи, которые должны быть заменены операциями сравнения. В течение последних лет было предложено множество алгоритмов, которые разделяют фрагмент столбца на три фрагмента, N фрагментов с использованием разрядной кластеризации частичной сортировки фрагментов в сочетании с операциями разбиения/слияния, полной сортировки фрагментов при первом подходе и разделения фрагментов на основе случайно выбранных опорных точек, в отличие от предикатов запроса, но при этом основная идея остается той же самой. На фиг. 1 показан запрос Q1, который разбивает весь столбец на три фрагмента, за которым следует запрос Q2, который реорганизует фрагменты 1 и 3. Другие фрагменты работы включают обновления, распространяют физическую организацию из одного столбца в другие по требованию и используют многоядерные процессоры.

Алгоритм 1 CrackInTwo(c,posL,posH,med,inc).

Проведем физическую реорганизацию фрагмента столбца с между posL и posH таким образом, чтобы все значения ниже med находились в непрерывном пространстве. inc указывает на то, является или нет med исключающим, например, если inc = false, то ϕ_1 - "<"и ϕ_2 - ">="

1: x_1 = point at position posL

2: x_2 = point at position posH

3: while position(x_1) < position(x_2) do

4: if value(x_1) ϕ_x med then
5: x_1 = point at next position
6: else
7: while value (x_2) ϕ_2 med &&
position(x_2) > position(x_1) do
8: x_2= point at previous position
9: exchange (x_1, x_2)
10: x_1 = point at next position
11: x_2 = point at previous position

Физически реорганизуйте часть столбца с между posL и posH таким образом, чтобы все значения ниже, чем med, находились в непрерывном пространстве, inc указывает, содержит ли med или нет, например, если inc = false, то ϕ_1 - "<" и ϕ_2 - ">="

1: x_1 = point at position posL
2: x_2 = point at position posH
3: while position (x_1) < position (x_2) do
4: if value (x_1) ϕ_1 med then
5: x_1 = point at next position
6: else
7: while value (x_2) ϕ_2 med &&
position (x_2) > position (x_1) do
8: x_2 = point at previous position
9: exchange (x_1, x_2)
10: x_1 = point at next position
11: x_2 = point at previous position

В данном документе без потери общности рассматривается основная конструкция взлома. Основное внимание направлено на действия по реорганизации столбца на фрагменты по отношению к контрольной точке как к части непосредственно плана запроса; таким образом, оператор выбора взлома физически реорганизует правильные фрагменты столбца, чтобы привести все квалифицирующие значения в смежную область, которая используется для возврата результата, без дополнительной материализации результата. Кроме того, при помощи ABJI-дерева, используемого для индексирования, дерево обновляется, и во время обработки запроса также выполняются действия по повторному уравновешиванию в логарифмическом масштабе времени. Эта базовая операция является общей и универсально применимой ко всем адаптивным схемам индексирования. Последующее обсуждение показывает, как применить это базовое понятие к зашифрованным данным; при этом следует подчеркнуть, что предложенная схема расширяет базовую схему взлома таким образом, чтобы не нарушить его предположения относительно базовой архитектуры: предполагается, что данные хранятся по принципу "один столбец в одно время" в плотных массивах с фиксированной шириной, как в современных столбцовых хранилищах, одновременно на диске и в памяти.

Обработка запросов может происходить как в массовом, так и в векторном виде. Эти основные свойства относятся ко всем столбцовым хранилищам.

Важно отметить, что адаптивное индексирование никогда не требует полной сортировки отдельных фрагментов; когда фрагмент становится достаточно маленьким, чтобы поместиться в кэше L1, данные можно сканировать практически без затрат. Таким образом, запросы вызывают реорганизацию только тех фрагментов данных, которые превышают порог размера; этот порог может быть больше (например, размер кэша L3) без существенного снижения производительности. Из контекста настоящей заявки следует, что полный порядок данных никогда не передается по каналам утечки за счет полностью отсортированного индекса, как это делает OPES по умолчанию.

Сущность изобретения

Таким образом, существует потребность в разработке упрощенной схемы шифрования, которая предусматривает (1) обработку запроса диапазона в отношении зашифрованных числовых данных, полученных из внешних источников в облаке или на удаленном сервере, и тем самым (2) инкрементное адаптивное индексирование, в результате чего индексируются только те данные, которые запрашиваются доверенными клиентами, которые стали индексированными.

Для решения этой проблемы в первом аспекте выполнен способ поиска в базе данных. Способ содержит шифрование множества значений данных в базе данных с использованием первого режима шифрования для получения зашифрованных значений данных; шифрование связанного значения с использованием второго режима шифрования для получения зашифрованного связанного значения, причем второй режим шифрования отличается от первого режима шифрования; и сравнение связанного значения с множеством значений данных с использованием зашифрованных связанных значений и зашифрованных значений данных.

В другом варианте осуществления способа значения, зашифрованные с использованием первого

режима шифрования, не сопоставимы друг с другом.

В дополнительном варианте осуществления способа сравнение связанного значения с множеством значений данных с использованием зашифрованных связанных значений и зашифрованного значения данных содержит сравнение связанного значения с множеством значений данных с использованием операции над зашифрованными связанными значениями и зашифрованными значениями данных.

В еще одном дополнительном варианте осуществления способа одно или более зашифрованных значений данных являются значениями ключа, используемыми для индексирования зашифрованных значений данных, при этом сравнение связанного значения с множеством значений данных содержит сравнение связанного значения с множеством значений данных с использованием зашифрованных связанных значений, значений ключа и зашифрованных значений данных.

В другом варианте осуществления способа шифрование множества значений данных в базе данных содержит преобразование каждого множества значений данных в вектор значений; шифрование связанного значения содержит преобразование связанного значения в связанный вектор; и сравнение связанного значения с множеством значений данных содержит получение скалярных произведений векторов значений на связанный вектор.

В дополнительном варианте осуществления способа каждый из векторов значений содержит зашумленный подвектор значений; связанный вектор содержит зашумленный связанный подвектор; где зашумленный связанный подвектор ортогонален каждому из зашумленных подвекторов значений.

В еще одном дополнительном варианте осуществления способ содержит выбор случайного положительного множителя для каждого из векторов значений; и масштабирование каждого вектора значений с помощью соответствующего выбранного множителя.

В другом варианте осуществления способ содержит выбор первого случайного положительного множителя и второго положительного множителя для каждого из векторов значений; масштабирование каждого подвектора зашумленных значений с помощью соответствующего первого случайного множителя; и масштабирование остальной части каждого из векторов значений с помощью соответствующего второго случайного положительного множителя.

В дополнительном варианте осуществления способ содержит выбор обратимой матрицы; где преобразование каждого из значений данных в вектор значений содержит преобразование значения данных в вектор-столбец значений и умножение одной из обратной матрицы и транспонированной матрицы на вектор-столбец значений; преобразование связанного значения в связанный вектор содержит преобразование связанного значения в связанный вектор-столбец и умножение другой обратной матрицы и другой транспонированной матрицы на связанный вектор-столбец.

Во втором аспекте раскрытия выполнен способ индексирования значений данных. Способ содержит шифрование множества значений данных с использованием первого режима шифрования для получения зашифрованных значений данных; шифрование одного или более связанных значений с использованием первого режима шифрования для получения первых зашифрованных связанных значений и с использованием второго режима шифрования для получения вторых зашифрованных связанных значений, причем второй режим шифрования отличается от первого режима шифрования; сравнение одного или нескольких связанных значений с множеством значений данных с использованием вторых зашифрованных связанных значений и зашифрованных значений данных; и индексирование зашифрованных значений данных на основании упомянутого сравнения с использованием первых зашифрованных связанных значений в качестве значений ключа.

В другом варианте осуществления способа значения, зашифрованные с использованием первого режима шифрования, не сопоставимы друг с другом; и значения, зашифрованные с использованием второго режима шифрования, не сопоставимы друг с другом.

В дополнительном варианте осуществления способ содержит шифрование другого связанного значения с использованием первого режима шифрования для получения другого первого зашифрованного связанного значения и с использованием второго режима шифрования для получения другого второго зашифрованного связанного значения; сравнение другого связанного значения с множеством значений данных с использованием другого второго зашифрованного связанного значения, значений ключа и зашифрованных значений данных; добавление другого первого зашифрованного связанного значения в качестве другого значения ключа на основании упомянутого сравнения.

В еще одном дополнительном варианте осуществления способа индексирование зашифрованных значений данных содержит построение иерархической индексной структуры, в которой узлы в иерархической индексной структуре представляют собой значения ключа.

В другом варианте осуществления иерархическая индексная структура представляет собой двоичное дерево поиска.

В дополнительном варианте осуществления способа двоичное дерево поиска представляет собой АВЛ-дерево.

В еще одном дополнительном варианте осуществления способ содержит добавление неоднозначности по меньшей мере к одному из значений данных, в результате чего множество интерпретаций для каждого по меньшей мере из одного из значений данных является возможным, при этом единственная ин-

терпретация из множества интерпретаций является истинной.

В другом варианте осуществления способ содержит добавление неоднозначности по меньшей мере в одно из одного или более связанных значений с тем, чтобы стало возможным множество интерпретаций по меньшей мере для одного из одного или более связанных значений, при этом единственная интерпретация из множества интерпретаций является истинной.

В дополнительном варианте осуществления способ шифрования множества значений данных с использованием первого режима шифрования содержит преобразование значений данных в векторы значений; шифрование одного или более связанных значений с использованием первого режима шифрования содержит преобразование связанных значений в первые связанные векторы; шифрование одного или более связанных значений с использованием второго режима шифрования содержит преобразование связанных значений во вторые связанные векторы; и сравнение одного или нескольких связанных значений с множеством значений данных содержит получение скалярных произведений векторов значений на вторые связанные векторы.

В третьем аспекте раскрытия выполнен сервер для индексирования значений данных. Сервер содержит средство хранения, выполненное с возможностью хранения зашифрованных значений данных, причем зашифрованные значения данных представляют собой множество значений данных, зашифрованных с использованием первого режима шифрования; блок приема, выполненный с возможностью приема, из клиента, первых зашифрованных связанных значений и вторых зашифрованных связанных значений, причем первые зашифрованные связанные значения представляют собой одно или несколько связанных значений, зашифрованных с использованием первого режима шифрования, и вторые зашифрованные связанные значения представляют собой одно или несколько связанных значений, зашифрованных с использованием второго режима шифрования, при этом второй режим шифрования отличается от первого режима шифрования; и блок обработки, выполненный с возможностью сравнения одного или более связанных значений с множеством значений данных с использованием вторых зашифрованных связанных значений и зашифрованных значений данных; и индексирования зашифрованных значений данных на основании упомянутого сравнения с использованием первых зашифрованных связанных значений в качестве значений ключа.

В другом варианте осуществления сервера зашифрованные значения данных не сопоставимы друг с другом.

В дополнительном варианте осуществления сервера блок приема дополнительно выполнен с возможностью приема другого первого зашифрованного связанного значения и другого второго зашифрованного значения, причем другое первое зашифрованное связанное значение является другим связанным значением, зашифрованным с использованием первого режима шифрования, и другое второе зашифрованное значение является другим связанным значением, зашифрованным с использованием второго режима шифрования; и блок обработки дополнительно выполнен с возможностью сравнения другого связанного значения с множеством значений данных с использованием другого второго зашифрованного связанного значения, значений ключа и зашифрованных значений данных и добавления другого первого зашифрованного связанного значения в виде другого значения ключа на основании упомянутого сравнения, причем сервер дополнительно содержит блок отправки, выполненный с возможностью отправки результата упомянутого сравнения с клиентом.

В четвертом аспекте раскрытия предусмотрен клиент для приема результата поиска. Клиент содержит блок отправки, выполненный с возможностью отправки второго зашифрованного связанного значения на сервер, причем второе зашифрованное связанное значение является связанным значением, зашифрованным с использованием второго режима шифрования, где сервер содержит зашифрованные значения данных, причем зашифрованные значения данных представляют собой множество значений данных, зашифрованных с использованием первого режима шифрования; блок приема, выполненный с возможностью приема из сервера результата сравнения между связанным значением и множеством значений данных с использованием второго зашифрованного связанного значения и зашифрованных значений данных, причем результат содержит ряд зашифрованных значений данных; и блок обработки, выполненный с возможностью дешифрования набора зашифрованных значений данных.

В другом варианте осуществления клиента блок отправки дополнительно выполнен с возможностью отправки первого зашифрованного связанного значения на сервер, причем первое зашифрованное связанное значение является связанным значением, зашифрованным с использованием первого режима шифрования.

В дополнительном варианте осуществления блок обработки выполнен с возможностью шифрования связанного значения с использованием второго режима шифрования для получения второго зашифрованного связанного значения и шифрования связанного значения с использованием первого режима шифрования для получения первого зашифрованного связанного значения.

В еще одном дополнительном варианте осуществления клиента по меньшей мере одно из множества значений данных имеет неоднозначность, в результате чего множество интерпретаций для каждого, по меньшей мере, из некоторого из множества значений данных является возможным, причем единственная интерпретация из множества интерпретаций является истинной; при этом процессор выполнен с воз-

можностью извлечения истинных интерпретаций значений данных из набора зашифрованных значений данных.

Краткое описание чертежей

На фиг. 1 показан взлом столбца значений;

на фиг. 2 - зашифрованный вектор;

на фиг. 3 - зашифрованный вектор с неоднозначностью;

на фиг. 4 показано, как в зашифрованном АВЛ-дереве можно найти фрагмент; и

на фиг. 5 показано, как в зашифрованное АВЛ-дерево можно добавить узел.

Подробное описание изобретения

Предложенная схема шифрования.

Изобретение направлено на разработку схемы шифрования, которая обеспечивает защиту от атак, нацеленных на нарушение конфиденциальности данных, и в то же время позволяет выполнять самоорганизующиеся операции индексирования над зашифрованными числовыми данными, т.е. обеспечивает индексируемую схему шифрования. Однако в отличие от OPES предложенная схема шифрования не сохраняет порядок числовых данных, так что она позволяет выполнять эффективные запросы без утечки информации о порядке кортежей. Злоумышленник может быть либо внешним злонамеренным объектом, либо честным, но любопытным сервером. Инсайдерские атаки, например те, которые возникают со стороны злонамеренных партнеров, не рассматриваются. Предполагается, что клиентские машины будут безопасными; конфиденциальная информация, такая как секретные ключи на клиенте, и запросы клиента не известны злоумышленникам. Вычисления злоумышленников ограничены схемами размеров полиномов. Такая схема шифрования должна удовлетворять следующим требованиям.

1. Она должна предоставлять серверу определенную возможность проведения сравнений неравенств между значениями данных (открытыми текстами), используя их зашифрованные формы (шифротексты).

2. Эта возможность выполнять сравнения не должна напрямую раскрывать порядок среди зашифрованных числовых данных.

3. Операции сравнения не должны требовать дешифрования на сервере; значения атрибута должны всегда оставаться скрытыми для сервера.

4. Размеры ключа, время шифрования и размеры шифротекста должны быть управляемыми и не должны увеличиваться в зависимости от размера данных, количества атрибутов или количества дискретных значений.

5. Клиент не должен быть намеренно вовлечен в обработку простых запросов; сервер должен доставлять зашифрованные результаты, и только их, клиенту в течение одного сеанса связи.

6. Сервер должен иметь возможность корректного размещения значений вновь полученных данных и поддержки обновлений в зашифрованных данных.

Основная задача состоит в том, чтобы разрешить противоречие между требованиями (1) и (2), т.е. обеспечить сравнение неравенства без раскрытия порядка. Способ достижения этого результата состоит в том, чтобы постулировать, что сравнения не должны быть возможными среди зашифрованных данных, постоянно хранящихся на сервере. Если такие данные не сопоставимы друг с другом, то их нельзя непосредственно привести в отсортированный порядок. Затем необходимо выяснить, при каких обстоятельствах сервер должен иметь возможность выполнять проверку неравенств среди шифротекстов. Такие сравнения должны быть возможны только по требованию между шифротекстами (например, между границей запроса и значением кортежа), которые представляются сопоставимыми по их шифрованию; по существу, действия по индексированию также будут возможны только по требованию клиента. Такое индексирование по требованию будет совместимо с адаптивным индексированием: если будет выполняться индексирование по требованию для обеспечения безопасности, необходимо также выполнять точно такие же действия для обеспечения адаптивности, так как данные должны индексироваться в ответ на запросы. Затем, вместо того, чтобы находиться во внутреннем противоречии друг с другом, требования по безопасности и адаптивности усиливают друг друга.

Далее необходимо разработать способ создания сопоставимых шифротекстов. Такой способ может использовать два взаимодополняющих режима шифрования A и B, взаимодействующих друг с другом, так что зашифрованный текст, зашифрованный в режиме A, можно сравнить с зашифрованным текстом в режиме B. Затем шифрование значений атрибутов в режиме A и границ запросов в режиме B дает серверу возможность сравнивать первое с последним; как можно видеть, такие сравнения образуют основу операций адаптивного индексирования и выполняются с помощью операторов ϕ_1 и ϕ_2 в алгоритме 1. Предположим, что серверу необходимо выполнить проверку неравенства между значением v ключа в столбце C и связанным значением b запроса, т.е. проверить, справедливо ли неравенство $v \geq b$. Эквивалентно, сервер должен проверить, справедливо ли неравенство

$$v - b \geq 0 \qquad (1)$$

Используя векторную форму записи, неравенство (1) можно выразить в виде

$$\begin{pmatrix}1\\ b\end{pmatrix}\cdot\begin{pmatrix}v\\ -1\end{pmatrix}=v-b\geq 0 \qquad (2)$$

Задача состоит в том, чтобы разработать схему шифрования, которая позволила бы проводить вычисление произведений таких скалярных векторов замаскированным образом. В этой схеме используются три операции маскировки, которые должны выполняться поставщиком информации при загрузке данных на сервере и доверенным клиентом перед отправкой запроса: (1) добавление шума; (2) скалярное умножение и (3) умножение матриц. Детали этих операций представлены ниже.

Добавление шума.

Операция добавления шума позволяет построить два более длинных вектора b, ν путем добавления дополнительных зашумленных элементов к векторам длиной -2, которые используются в неравенстве (2), поэтому результат произведения векторов остается тем же самым, b и ν упоминаются как зашифрованный связанный вектор и вектор зашифрованных значений, соответственно. Конкретная длина l векторов b и ν, расположение и различия между добавленными зашумленными содержаниями и первоначальными содержаниями значений в них являются составной частью ключа шифрования, известного владельцу данных и доверенному клиенту, но не поставщику услуг. Без потери общности представлен пример использования векторов длиной 5. Неравенство (2) можно переписать следующим образом:

$$\mathbf{b}\cdot\mathbf{v}=\begin{pmatrix}-4\\ 1\\ 8\\ b\\ 4\end{pmatrix}\cdot\begin{pmatrix}3\\ v\\ -2\\ -1\\ 7\end{pmatrix}=v-b\geq 0 \qquad (3)$$

В этом примере первоначальные содержания векторов занимают в них позиции 1 и 3, тогда как позиции {0,2,4} зарезервированы для зашумленных содержаний, которые взаимно уничтожаются в произведении скалярных векторов, так как они образуют два ортогональных вложенных подвектора n_b и n_v длиной -3 с внутренним произведением 0

$$\mathbf{n}_b\cdot\mathbf{n}_v=\begin{pmatrix}-4\\ 8\\ 4\end{pmatrix}\cdot\begin{pmatrix}3\\ -2\\ 7\end{pmatrix}=0 \qquad (4)$$

В данном документе эти ортогональные подвекторы называются зашумленными подвекторами b и ν. Следует подчеркнуть, что одни и те же точные зашумленные подвекторы не должны использоваться для каждого зашифрованного значения v ключа или каждого зашифрованного связанного значения b, при этом одинаковые позиции должны использоваться для зашумленных содержаний во всех зашифрованных значениях в одном и том же столбце. Можно использовать любую пару векторов, ортогональных друг другу, поэтому их внутреннее произведение равно 0. Тем не менее, вектор b, полученный для заданного связанного значения b, должен быть ортогонален всем векторам v, полученным для значений атрибута. Поэтому нельзя выбрать пары ортогональных векторов совершенно произвольным образом. Однако достаточно выбрать единичный вектор u и убедиться, что каждое связанное значение b скрыто путем

вложения в вектор-столбец $\binom{1}{b}$ случайно выбранного зашумленного вектора n_b,

коллинеарного u, $\mathbf{n}_b=\lambda(b)\cdot\mathbf{u}$, при этом значения l и b занимают определенные предварительно выбранные позиции, как показано выше, для получения b. Затем любое значение v данных скрыто путем аналогичного вложения произвольно выбранного вектора $\mathbf{n}_v=\mathbf{u}^{\perp}(v)$, ортогонального u, в вектор-столбце $\binom{v}{-1}$, для получения v. Следует подчеркнуть, что векторы $\mathbf{u}^{\perp}(v)$ не должны быть коллинеарными друг другу. Будет достаточно, что любой вектор ортогонален u. Подводя итог и принимая во внимание единичный вектор u, зашумленные подвекторы, вложенные в b, должны быть коллинеарны u, в то время как вложенные в v должны быть ортогональны u. Затем проверку неравенства можно выполнить посредством произведения скалярных векторов, так как v - b вычисляется как b · v.

В вышеупомянутом примере используемый единичный вектор

$$\mathbf{u}=\begin{pmatrix}\frac{-1}{\sqrt{6}}\\ \frac{2}{\sqrt{6}}\\ \frac{1}{\sqrt{6}}\end{pmatrix}.$$

в частности, $\|\mathbf{u}\| = 1$. Зашумленный подвектор b получается путем умножения u на случайно выбранный коэффициент λ(b). В вышеупомянутом случае $\lambda(b) = 4\sqrt{6}$. С другой стороны, зашумленный подвектор ν получается путем случайного выбора любого вектора $\mathbf{u}^{\perp}(v)$ длиной l - 2, ортогонального u.

Скалярное умножение.

Схема шифрования, разработанная в данное время, позволяет вычислить разность ν - b с помощью сервера путем вычисления скалярного произведения двух векторов. Эта схема решает задачу выполнения проверок неравенств и, следовательно, обработку запросов диапазона в отношении зашифрованных числовых данных, в то время как схема шифрования не сохраняет порядок данных, как это делает OPES. Однако учитывая зашифрованный связанный вектор b, злоумышленник может вычислить любое скалярное произведение вида b · ν и, следовательно, разность ν - b, для любого вектора ν зашифрованных значений; затем, сравнивая полученные значения ν - b, можно получить не только порядок, но и точные разности между зашифрованными значениями данных.

Чтобы предотвратить такого рода утечку, предпочтительно избегать вычисления точных разностей вида ν - b. В конце концов, нет необходимости получать правильные нормы этих разностей. Достаточно просто получить их знак. Норму |ν - b| можно изменять до тех пор, пока знак получается правильно. Этот эффект может быть достигнут путем добавления скалярного умножения в операции скрытия. Затем для каждого значения ν данных поставщик данных выбирает случайный положительный множитель ξ(ν) > 0 и повторно вычисляет вектор ν зашифрованных значений в виде ν'=ξ(ν)ν. В дальнейшем ν ссылается на этот повторно вычисленный вектор значений. В результате скалярное произведение принимает вид b · ν = ξ(ν)(ν-b). Сервер по-прежнему получает правильный знак разности ν - b без точной нормы этой разности, подверженной утечке. В действительности злоумышленник не может восстановить порядок значений данных, используя информацию, полученную из скалярных произведений.

Вместо использования общего множителя масштабирования для зашифрованных значений и зашумленных компонентов, порожденных некоторыми псевдослучайными функциями ξ(x), мы можем использовать две отдельные псевдослучайные функции $\xi_1(\nu)$ и $\xi_2(\nu)$. Таким образом, зашумленные и определенные компоненты могут масштабироваться независимым образом, но по всей вероятности будут распределяться тем же самым образом.

Умножение матриц.

В результате выполнения операции добавления шума и скалярного умножения получается вектор ν, представляющий каждое значение ν ключа, и вектор b, представляющий собой связанное значение b запроса, так что знак, но не норма ν-b, может быть получен из скалярного произведения b · ν. Тем не менее, безопасность, которая обеспечивает эти операции, не может противостоять простому нарушению: злоумышленник, изучающий позицию в векторе ν, где находятся значения ξ(ν)ν и -ξ(ν), может эффективно получить все исходные значения ключа. Таким образом, обеспечивается дополнительный уровень шифрования для скрытия этого нарушения.

Предположим, что ν и b являются векторами длиной l, и допустим, что M представляет собой любую обратимую матрицу размером l ×l, и M^{-1} является ее обратной матрицей. M образует часть ключа шифрования; это известно поставщику данных и его доверенным клиентам, но не поставщику услуг. Любое значение ν ключа и любую контрольную точку b можно зашифровать следующим образом:

$$\mathbf{E}_v(v) = M^{-1}\mathbf{v} \qquad (5)$$

$$\mathbf{E}_b(b) = M^T\mathbf{b} \qquad (6)$$

где

$\mathbf{E}_v(\cdot)$, $\mathbf{E}_b(\cdot)$ - функции шифрования,

b и ν представлены в виде вектор-столбцов, и

M^T - транспонированная матрица M.

Затем

$$\mathbf{E}_b(b)\cdot\mathbf{E}_v(v) = \mathbf{E}_b(b)^T\mathbf{E}_v(v)$$

$$= (M^T\mathbf{b})^T(M^{-1}\mathbf{v})$$

$$= (\mathbf{b}^T M)(M^{-1}\mathbf{v})$$

$$= \mathbf{b}^T\mathbf{v} = \mathbf{b}\cdot\mathbf{v} = \xi(v)(v-b)$$

где вектор-строка a^T - транспонированный вектор-столбец a.

Специалистам в данной области техники очевидно, что одинаковый результат может быть получен,

когда $E\nu(\nu) = M^T$ и $E_b(b) = M^{-1}b$.

В действительности любая проверка неравенства может быть эффективно проведена с использованием зашифрованных векторов $E\nu(\nu)$ и $E_b(b)$. $E\nu(\nu)$ вырабатывается поставщиком данных и передается на сервер при выработке (или обновлении данных). $E_b(b)$ получается доверенным клиентом или снова поставщиком данных от имени клиента при выдаче запроса. Эти два этапа шифрования (и обратная операция - дешифрование) являются единственными поставщиками данных рабочей нагрузки, и клиент должен их пройти. Другие вычисления и обработка запросов выполняются сервером как с незашифрованной базой данных. В то же время, учитывая эти зашифрованные векторы, злоумышленник не может определить значения ν или b без знания обратимой матрицы M.

Ключ шифрования.

Вкратце, общий ключ для шифрования включает в себя:

1. Единичный вектор u.

2. Произвольную ориентацию каждого вектора $\mathbf{u}^{\perp(\nu)}$, ортогонального u, который индивидуально выбирается для построения ν для значения ν данных.

3. Каждый коэффициент $\lambda(b)$, используемый для получения зашумленного вектора, коллинеарного u, для того, чтобы построить b для границы b запроса.

4. Каждый случайный положительный множитель $\xi(\nu)$, используемый для скрытия нормы ν - b.

5. Позиции, занятые содержаниями $\binom{\xi(\nu)\nu}{-\xi(\nu)}$ и $\binom{1}{b}$ в ν и b соответственно.

6. Обратимую матрицу.

Стойкость к атакам.

Теперь будут изложены в общих чертах возможности этой упрощенной схемы шифрования, чтобы противостоять атакам при условии, что честный, но любопытный злоумышленник осведомлен о внутренних принципах работы схемы шифрования и стремится узнать ключ из известных шифротекстов и потенциально известных открытых текстов. Так как схема шифрования состоит из трех уровней, ее дальнейшее обсуждение будет разбито на две части.

Уровни скалярного умножения и шума.

Оставив в стороне часть перемножения матриц, предположим, что злоумышленница Алиса, которая непосредственно наблюдает за зашумленными векторами, прежде чем они будут умножены на M. При такой атаке на известный шифротекст Алисе нужно было бы только отсортировать зашумленные содержания из содержаний значений этих векторов. Она может использовать информацию о том, что зашумленные элементы вектора ν зашифрованных значений и вектора b зашифрованных контрольных точек ортогональны друг другу, и выдвинуть гипотезу относительно позиций зашумленных содержаний в зависимости от содержаний значений в наблюдаемых векторах, т.е. произвольно выбрать 2 позиции из l позиций в этих векторах, где находятся предполагаемые содержания значений. Чтобы проверить эту гипотезу, она может проверить, производят ли предполагаемые зашумленные подвекторы неизменно внутреннее произведение 0 по всем многочисленным различным наблюдаемым парам {b, ν}. Если это так, то она может с уверенностью сделать вывод, что ее гипотеза проверена. Вопрос состоит в том, сколько попыток должна сделать Алиса для того, чтобы получить правильную гипотезу в результате исчерпывающего поиска. Ее гипотеза сводится к выбору 2 из l векторных элементов. Так как существуют $\binom{\ell}{2} = \frac{\ell(\ell-1)}{2} = O(\ell^2)$ способов сделать такой выбор, Алиса может прийти к правильной гипотезе за полиномиальное время. Как указанно выше, уровень шума схемы шифрования легко раскрыть. Вот почему был добавлен уровень умножения матриц, который будет обсужден ниже.

Уровень умножения матриц.

Как только Алиса увидит, что зашумленные векторы умножены на M, она не может выполнить вышеуказанную атаку. Теперь матрица M умножения вводит шаблон, состоящий из l^2 неизвестных. Единичный вектор u несет в себе более l - 2 неизвестных в шаблоне; каждый зашифрованный вектор ν несет в себе более l - 3 неизвестных, так как можно получить только один из его l - 2 зашумленных элементов, используя l - 3 других элементов, и ортогональность к u и коэффициент $\xi(\nu)$, который был умножен на $\{\nu,-1\}$ для того, чтобы получить свои зашумленные содержания; каждый зашифрованный вектор b несет в себе коэффициент $\lambda(b)$, который умножается на u для получения своих зашумленных содержаний. Атака, которая все еще может приносить плоды, известна как атака на основе подобранного открытого текста в случае, если Алиса может получить доступ к парам из исходного значения (ν или b) и его зашифрованной формы ($E\nu(\nu)$ или $E_b(b)$ соответственно). Для каждой такой пары она может построить l скалярных линейных уравнений. В конечном счете, можно показать, что Алисе достаточно знать $N \geq \frac{\ell^2+\ell-2}{\ell-1}+1 = O(\ell)$ пар "открытый текст-шифротекст" значений b; затем она может решить полученную систему уравнений для каждого из $\frac{\ell(\ell-1)}{2}$ способов позиционирования зашумленных содержаний в ν и b и снова определить то, что приводит к решению, неизменно приводя к произведениям ортогональных за-

шумленных векторов. Безопасность предложенной схемы шифрования в целях предотвращения известных атак на основе открытого текста сильно зависит от выбранного размера l шифротекста, в то время как предложенная схема предоставляет владельцу данных гибкость при выборе значения l по желанию.

Индексирование зашифрованных данных.

Ниже будет обсуждено, как сервер может выполнить адаптивное индексирование данных, зашифрованных схемой шифрования. Примечательно, что выбор l в схеме определяет размеры матрицы M и влечет за собой l-кратное увеличение требований к хранению. Эти затраты на хранение имеют доступную цену, которую нужно заплатить для выполнения операций по безопасному адаптивному индексированию.

Проблема утечки из-за структуры.

До сих пор подчеркивалось, что схема шифрования с сохранением порядка исключается, поэтому порядок значений не вытекает к злоумышленникам. Тем не менее, операции адаптивного индексирования постепенно приводит столбец базовых данных к сортированному порядку. В конце концов, взлом базы данных можно правильно описать как инкрементную быструю сортировку, в то время как другую альтернативу для адаптивного индексирования, т.е. адаптивное слияние можно рассматривать как инкрементную сортировку при внешнем слиянии. Механизм адаптивного индексирования, представленный его собственными устройствами, будет стремиться приводить данные к сортированному порядку. С учетом рабочей нагрузки W, при которой для любой пары значений в столбце v_1, v_2, существует по меньшей мере одна граница $b \in \{v_1, v_2\}$ запроса, адаптивное индексирование с помощью W приведет данные в конечном счете к точному сортированному порядку.

В действительности, даже если шифрование само по себе не выдает порядок значений, этот порядок в конечном итоге может наблюдаться в записях порядка, которые приводятся после применения достаточного количества операций взлома. Злоумышленник, который может идентифицировать целевой кортеж, может логически вывести свою позицию в сохраненном порядке путем наблюдения структуры построенного дерева индексов. Чем более улучшенным становится дерево, тем больше может утечь информации относительно порядка лежащих в основе кортежей.

Умышленное допущение ошибок.

Чтобы смягчить проблему утечки порядка из-за структуры, вводятся ошибочные интерпретации данных, извлеченных из внешних источников, в то время как сервер не может различить, какая интерпретация является действительной. Таким образом, даже злоумышленник (или сервер), знающий внутреннюю работу схемы шифрования, не сможет сделать уверенные выводы об относительных позициях кортежей.

В данном документе раскрыты два возможных способа интерпретации каждого зашифрованного значения данных, без утечки того, какое одно из двух значений является правильным для каждого кортежа t. Сервер просто выполняет действия над обоими значениями, поддерживая обе возможные интерпретации каждой записи тогда, когда это необходимо. Только одна из этих версий соответствует истинной записи, но только владелец данных и его доверенные клиенты могут провести это различие. С точки зрения сервера, каждая интерпретация вероятна в равной степени. Другими словами, сервер одновременно управляет многочисленными возможными мирами. Такие многочисленные реальности конфигурируются путем увеличения длины каждого вектора $E_v(v)$ на значение θ и произвольного добавления θ в качестве префикса или суффикса к нему для того, чтобы получить $\overline{\mathbf{E}}_v(v)$; θ выбирается таким образом, чтобы как l-префикс, так и l-суффикс $\overline{\mathbf{E}}_v(v)$ работали последовательно в качестве действительных векторов $E_v(v)$, сопряженных с векторами $E_b(b)$: в обоих случаях после умножения на матрицу M, содержания, полученные в позициях, зарезервированных для зашумленных содержаний в векторе v, ортогональны u.

Чтобы облегчить последующее обсуждение, введен ряд матриц, которые полезны для формального представления создания векторов и добавления шума посредством операций линейной алгебры. Эти матрицы и цели, которым они служат, представлены вместе в таблице.

Используемые матрицы

Матрица	Назначение
E_{nm}	*матрица расширения*: расширяет вектор $n-m$ нулями
P_{nm}	*матрица перестановок*: выполняет перестановку расширенного значения /связанных векторов
P^{c}_{nm}	*дополнительная матрица перестановок*: выполняет перестановку вектора с шумом
S	*сдвиговая матрица*: циклически сдвигает элементы вектора вниз
S^{T}	*симметричная сдвиговая матрица*: циклически сдвигает элементы вектора вверх

Значение θ можно вычислить следующим образом.

Вычисления начинаются с расчета $(v;-1)^{T}$, зашумленного подвектора $\mathbf{n}_{v} \propto \mathbf{u}^{\perp}$ и $(1;b)^{T}$ и зашумленного подвектора $\mathbf{n}_{b} = \lambda(b)\cdot\mathbf{u}$, где u - секретный единичный вектор, и $\mathbf{u}^{\perp}$ - вектор, ортогональный u. Затем для целей схемы шифрования сначала необходимо формально описать разложения этих векторов на два вектора с размером l. С этой целью используются две матрицы расширения размером n×m, которые обозначены как E_{nm}; такие матрицы перемножают векторы, расположенные слева, и тем самым расширяют эти векторы с помощью n-m нулей. Структура матрицы E_{nm} может быть представлена в виде $\begin{pmatrix} I \\ 0 \end{pmatrix}$, где I - единичная матрица. Затем ее можно представить в следующем виде:

$$E_{\ell 2}\begin{pmatrix} v \\ -1 \end{pmatrix} = \begin{pmatrix} v \\ -1 \\ 0 \\ \dots \\ 0 \end{pmatrix}, E_{\ell(\ell-2)}\mathbf{n}_{v} = \begin{pmatrix} \mathbf{n}_{v} \\ \dots \\ 0 \end{pmatrix}$$

На следующем этапе необходимо перемешать содержания этих расширенных векторов, чтобы привести их содержания в позиции, предсказанные схемой шифрования. С этой целью используются две матрицы перестановки размером n×n, которые обозначены как P_{nm} и P^{c}_{nm}; эти матрицы перемешивают расширенное значение/границу и векторы c, согласно схеме добавления шума; только их первые m шумом, соответственно строк имеют ненулевые содержания, следовательно, $P_{nm}(P_{nm})^{T} = \begin{pmatrix} I_{m} & 0 \\ 0 & 0 \end{pmatrix}$; P^{c}_{nm} выбирается таким образом, чтобы $P_{nm}(P^{c}_{nm})^{T} = 0$, т.е. P_{nm} и P^{c}_{nm} не имеют пересечений после перестановки: элементы вектора перемешиваются в позициях, дополняющих друг друга. Наконец, допустим, что ξ(v) и λ(b) будут "масштабирующими" функциями, используемыми в схеме шифрования, как определено в разделе "Ключ шифрования". Таким образом, масштабированное зашумленное значение и связанные векторы могут быть представлены в виде

$$\mathbf{v} = \xi(v)\left(P_{\ell 2}E_{\ell 2}\begin{pmatrix} v \\ -1 \end{pmatrix} + P^{c}_{\ell(\ell-2)}E_{\ell(\ell-2)}\mathbf{n}_{v}\right)$$

$$\mathbf{b} = P_{\ell 2}E_{\ell 2}\begin{pmatrix} 1 \\ b \end{pmatrix} + P^{c}_{\ell(\ell-2)}E_{\ell(\ell-2)}\lambda(b)\mathbf{u}$$

Окончательный вектор зашифрованных зашумленных значений обозначается как $E\nu = M^{-1}v$, где M - матрица шифрования. Описанная схема допустимых отклонений и ошибок позволяет получить один из следующих двух векторов:

$$\begin{pmatrix} \mathbf{E}_{v} \\ \theta \end{pmatrix} \text{ или } \begin{pmatrix} \theta \\ \mathbf{E}_{v} \end{pmatrix}$$

Без потери общности рассмотрим только первый вариант. В этом первом варианте необходимо представить вектор $\mathbf{E}^{f}_{v}$ поддельных зашифрованных зашумленных значений, т.е. l-суффикс $\overline{\mathbf{E}}_{v}(v)$, который злоумышленник может рассматривать как истинный. С этой целью используется сдвиговая матрица

l×l, которая обозначается как S; умножение на эту матрицу приводит к циклическому сдвигу компонентов вектора вниз.

Например, для

$$n=3:\ S=\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}.$$

Симметрично, ее транспонированная матрица S^T сдвигает компоненты вектора вверх. Таким образом

$$\mathbf{E}_v^f = S^T\mathbf{E}_v + (\theta - \mathbf{E}_v\cdot\mathbf{e}_1)\mathbf{e}_\ell$$

где e_k - единичный вектор со всеми нулевыми компонентами кроме k^{th}; приведенное выше уравнение просто сохраняет (ξ-1)-префикс, сдвинутый на одну позицию вверх, при этом θ размещается в l-й позиции.

Это вектор $\mathbf{E}_v^f$ должен иметь все свойства действительного вектора зашифрованных значений. Затем "зашумленный" l - 2×1 подвектор соответствующего вектора $\mathbf{v}^f = M\mathbf{E}_v^f$ поддельных значений должен быть ортогональным к n_b

$$\mathbf{n}_v^f\cdot\mathbf{n}_b = E_{\ell(\ell-2)}^T(P_{\ell(\ell-2)}^c)^T\mathbf{v}^f\cdot\mathbf{n}_b = 0 \Leftrightarrow$$

$$E_{\ell(\ell-2)}^T(P_{\ell(\ell-2)}^c)^T M\left[S^T\mathbf{E}_v + (\theta-\mathbf{E}_v\cdot\mathbf{e}_1)\mathbf{e}_\ell\right]\cdot\mathbf{u} = 0 \Leftrightarrow$$

$$\left[\mathbf{E}_v^T S + (\theta-\mathbf{E}_v\cdot\mathbf{e}_1)\mathbf{e}_\ell^T\right]M^T P_{\ell(\ell-2)}^c E_{\ell(\ell-2)}\cdot\mathbf{u} = 0 \Leftrightarrow$$

$$\theta = \mathbf{E}_v\cdot\mathbf{e}_1 - \frac{\mathbf{E}_v^T SW\cdot\mathbf{u}}{\mathbf{e}_\ell^T W\cdot\mathbf{u}},$$

где

$$W = M^T P_{\ell(\ell-2)}^c E_{\ell(\ell-2)}.$$

На фиг. 2 показан вид зашифрованного вектора, в то время как на фиг. 3 показан зашифрованный вектор $\overline{\mathbf{E}}_v(v)$ с добавленной неоднозначностью. Размер вектора с добавленной неоднозначностью на единицу больше, чем исходный зашифрованный вектор, следовательно, каждое исходное значение получает два представления в базе данных: одно, представленное действительным зашифрованным вектором, как l-префикс на фигуре, и другое, представленное поддельным вектором, как ξ-суффикс на фигуре. Оба значения получены из вектора с добавленной неоднозначностью, как показано на фиг. 3.

При добавлении этой неоднозначности относительно идентичности действительного зашифрованного вектора, доверенный клиент должен представить эту идентичность идентифицируемой. Это может быть достигнуто путем выбора (ранее объявленного случайным) множителя ξ(ν) конкретным способом, например, постулируя, что он будет нечетным целым числом. Владелец данных и доверенный клиент могут затем определить действительный вектор Eν(ν) с учетом $\overline{\mathbf{E}}_v(v)$ следующим образом: они дешифруют как l-префикс, так и l-суффикс $\overline{\mathbf{E}}_v(v)$, умножая каждый префикс на матрицу M. Вектор, который выдает нечетное целое число в позиции ξ(ν), является действительным вектором. Тот факт, что только одна попытка дешифрования выдает нечетное целое число в этой позиции, проверяется владельцем данных во время шифрования.

Однако то, что клиент может определить так легко, недоступно для сервера. В рамках этой схемы, сервер генерирует и управляет каждой возможной интерпретацией $\overline{\mathbf{E}}_v(v)$ по отдельности; причем эта избыточность создает неясность, которая снижает достоверность выводов злоумышленника относительно порядка значения. Конечный результат является аналогичным добавлению поддельных записей в базу данных. Однако безопасность, обеспечиваемая предложенной им схемой, выше, чем безопасность, обеспечиваемая простой поддельной вставкой. В случае добавления подделок злоумышленник, обладающий достаточными базовыми знаниями, может идентифицировать единственную запись, представляющую интерес, и сделать выводы относительно утечки информации о своей позиции в индексе. С другой стороны, в предложенной схеме позиция записи, представляющая интерес, в индексе является неопределенной даже тогда, когда идентифицирована эта запись, представляющая интерес, так как каждая отдельная запись порождает две возможные интерпретации; такое положение дел придает дополнительную безопасность предложенному построению.

Тем не менее, предложенный подход для выработки неопределенных данных имеет несколько не-

достатков: значения в префикса/суффикса вырабатываются детерминировано относительно соответствующих зашифрованных значений, внося дополнительную утечку, в то время как их область отклоняется от области зашифрованных действительных компонентов вектора, таким образом, θ можно легко различить. В другом варианте осуществления предложен другой подход, который не может обеспечивать какую-либо возможность сжатия, но обеспечивает гарантии безопасности.

1. Во время инициализации, в зависимости от "действительного" распределения Rν данных (полученного из внешних данных, переданных из внешних источников, путем выборки или опытным путем), "поддельное" распределение Fν данных вырабатывается клиентом и становится частью секретного ключа.

2. Всякий раз, когда клиент обновляет базу данных, он шифрует каждое "действительное" значение ν^f наряду с "поддельным" значением ν^f выбранным из Fv, и отправляет их одновременно на сервер в произвольном порядке. Таким образом, распределение поддельных значений может контролироваться напрямую, и оно является менее детерминированным.

Выработка неоднозначных запросов.

Неоднозначные "поддельные" значения данных вводятся для того, чтобы сделать злоумышленника менее уверенным по отношению к распределению данных в рамках FA наряду с добавлением экспоненциальной сложности в случае известной атаки на основе открытого текста (KPA) на данные (см. раздел 5 анализа безопасности). Однако необходимо также использовать неоднозначность при выдаче запросов, так как связанные значения запроса раскрывают шаблоны поиска и также подвергаются KPA. Для достижения этого эффекта предложен следующий подход.

1. Во время инициализации в зависимости от "действительного" распределения Rν данных "поддельное" распределение Fν запроса вырабатывается клиентом и становится частью секретного ключа.

2. Всякий раз, когда клиент выдает запрос, он шифрует каждый "действительный" запрос b_L;b_R диапазона; наряду с "поддельным" запросом b_L^f, b_R^f диапазона, выбранным из F_b, и отправляет их одновременно в произвольном порядке.

3. Сервер не может определить, какой запрос является "действительным"; он обрабатывает оба запроса и отправляет результаты в том же самом порядке.

4. Клиент отбрасывает "поддельную" часть результатов и фильтрует ложно-положительные результаты.

Таким образом, даже в том случае, если злоумышленник имеет доступ к признанному авторитету, который вырабатывает шифротексты заданных границ запроса, все еще существует экспоненциальная сложность для KPA и неопределенность в отношении распределения данных в FA с управляемыми издержками удвоенной сложности связи и удвоенного размера индекса.

Кроме того, эта неоднозначная выработка запросов решает отдельную проблему в контексте адаптивного индексирования: неявная схема взлома выполняет индексирование вслепую на основе границ входящих запросов; таким образом, они оставляют в стороне вопрос об устойчивости рабочей нагрузки и уязвимы к ненадлежащему исполнению в случае запросов, выбирающих точки наихудших случаев, точно так же, как и быстрая сортировка, уязвима для квадратичной сложности наихудшего случая. Стохастический взлом [16] предлагает добавить случайность при выборе отправных точек взлома, тем самым достигая устойчивости к произвольным рабочим нагрузкам. В нашей схеме эта случайность эффективно переносится со стороны сервера на сторону клиента в форме выработки неоднозначных запросов: введенная неоднозначность служит не только целью безопасности, но также и целью устойчивости рабочей нагрузки за один прием.

Кроме того, чтобы клиент мог фильтровать ложно-положительные результаты, было предложено сделать член масштабирования целочисленным для "действительных" значений и дробным для "поддельных" значений. Поскольку клиенту нет необходимости отфильтровывать ложно-положительные результаты по самим же выданным запросам, мы предлагаем, чтобы коэффициент масштабирования был дробным или одновременно представлял собой "действительные" и "поддельные" запросы, тем самым делая шифрование с привязкой к запросу односторонней функции.

Управление зашифрованным АВЛ-деревом.

Предложенная схема шифрования позволяет серверу управлять столбцом зашифрованных значений, даже при вставленной неоднозначности, и проводить сравнения между границами запросов и значениями данных. Тем не менее, предложенная система должна также построить АВЛ-дерево; в частности, границы запроса, которые соответствуют контрольным точкам b, используются в качестве значения ключа в узлах дерева и применяются в последующих обходах дерева. Во время такого обхода новое связанное значение b' должно сравниваться с предыдущим значением b, используемым теперь в качестве ключа; затем в каждом узле должны сравниваться друг с другом два значения b' и b контрольных точек.

Для того чтобы адаптировать этот механизм обхода дерева к зашифрованным данным, сервер должен иметь возможность сравнивать два значения b' и b контрольных точек друг с другом. Однако в предложенной схеме шифрования используются два различных режима шифрования, один для контрольных точек b и другой для значений ν, полученных для того, чтобы обеспечить сравнение только b с ν, но не с

другими значениями b и ν друг с другом. Эта проблема решена за счет того, что клиент, выдающий запрос, шифрует контрольную точку b обоими способами, т.е. своим собственным способом, как $E_b(b)$, и как значение $E\nu(b)$ атрибута, и передает их на сервер.

Впоследствии, первое значение используется для проверки неравенства со значениями атрибута, а второе значение используется для его хранения в качестве ключа в построенном индексе АВЛ-дерева. При поиске зашифрованного АВЛ-дерева для нового значения b' контрольной точки требуемое сравнение производится путем вычисления

$$\mathbf{E}_b(b') \cdot \mathbf{E}_v(b) = \xi(b) \cdot (b - b')$$

Рассмотрим пример зашифрованного АВЛ-дерева, показанного на фиг. 4. При поиске фрагмента в этом дереве используется граница b запроса, зашифрованная как $E\nu(b)$, и значение ключа непосредственно в дереве, зашифрованное как $E\nu(b)$; таким образом, граница запроса и значения ключа взаимодействуют друг с другом и обеспечивают проведение сравнения, которые приводят к обходу, показанному на фигуре.

Алгоритм findpiece(objAVLTree,N ,E_b,posL ,posH).

Данный алгоритм обходит АВЛ-дерево и возвращает значения верхней границы (posH) и нижней границы (posL) фрагмента, в котором расположен связанный вектор E_b запроса.

```
1:  posL = 0
2:  posH = N
3:  isFound = true
4:  rootNode = objAVLTree.getRoot()
5:  if rootNode is not empty then
6:    minNode = objAVLTree.findMinNode(root)
7:    maxNode = objAVLTree.findMaxNode(root)
8:    fNode = objAVLTree.findNode(Eb)
9:    if ScalarProduct(Eb,maxNode.key) > 0 then
10:     isFound = false
    **Case 1**
11:   if isFound == false && fNode is not minNode then
12:     posL = maxNode.position
    **Case 2**
13:   else if fNode == minNode then
14:     if ScalarProduct(Eb,fNode.key) < 0 then
15:       posH = fNode.position
16:     else
17:       posL = fNode.position
18:     fNode = objAVLTree.findSuccessor(fNode)
19:     if ScalarProduct(fNode.key,maxNode.key) < 0 then
20:     posH = fNode.position
    **Case 3**
```

```
21:    else if ScalarProduct(Eb,fNode.key) < 0 then
22:        posH  = fNode.position
23:        posL  = objAVLTree.findPredecessor(fNode).position
       **Case 4**
24:    else
25:        posL  = fNode.position
26:        fNode = objAVLTree.findSuccessor(fNode)
27:        if ScalarProduct(Eb,fNode.key) < 0 then
28:            posH  = fNode.position
```

Алгоритм findpiece иллюстрирует нахождение позиций взлома для границы b запроса; он использует наибольшие и наименьшие значения в АВЛ-дереве для того, чтобы определить находится ли граница запроса вне диапазона дерева; в противном случае он получает конечный узел fNode с помощью операции поиска по дереву на основании заданной границы E_b запроса; в целом, он различает следующие случаи.

Случай 1 имеет место, если b больше самого большого значения в дереве; после этого самое большое значение в дереве возвращается в виде нижней границы диапазона фрагмента.

Случай 2 имеет место, когда возвращаемый fNode равен наименьшему значению в дереве; затем, если граница b запроса превышает это значение, диапазон возвращается с этого значения до его следующего элемента; если граница запроса меньше, то наименьшее значение возвращается как верхняя граница диапазона фрагмента.

Случай 3 имеет место, если b меньше fNode; после этого диапазон возвращается в место между fNode и его предшествующим элементом.

Случай 4 имеет место, когда b больше fNode; после этого диапазон возвращается в место между fNode и его последующим элементом, если такой существует.

Во всех случаях сравнения выполняются посредством скалярных векторных произведений в соответствии с предложенной схемой.

Располагая фрагментом, к которому относится граница b запроса, необходимо завершить операцию взлома путем добавления границы b, представленной как $E\nu(b)$, так и $E_b(b)$ в позиции конечного узла непосредственно в АВЛ-дереве (и возможно повторно уравновесить дерево) для того, чтобы облегчить поиск в будущем.

На фиг. 5 показан обход АВЛ-дерева вместе с добавлением конечного узла для b_8 на последнем этапе, после разбиения фрагмента, к которому принадлежит b_8, на две части. Алгоритм addCrack показывает, как выполняется эта операция; в первых трех случаях рассмотрена ситуация, когда узел для b уже существует в дереве; четвертый случай добавляет новый узел.

Алгоритм addCrack (objAVLTree).

Этот алгоритм добавляет новый узел, для границы b запроса, в N значений индексирования АВЛ-дерева, соответствующих позиции pos в массиве данных.

```
1:  if pos == 0 or pos ≥ N then return
2:  rootNode = objAVLTree.getRoot()
3:  isFound = true
5:  if rootNode is not empty then
6:    minNode = objAVLTree.findMinNode(root)
7:    maxNode = objAVLTree.findMaxNode(root)
8:    fNode = objAVLTree.findNode( E_b )
9:    if ScalarProduct( E_b ,maxNode.key) > 0 then
10:     isFound = false
 **Case1**
11:   if isFound == true then
12:     if fNode.position == pos then return
12:     tmp = fNode
13:     if ScalarProduct(tmp.key, E_b ) == 0 then
14:       tmp = objAVLTree.findSuccessor(tmp)
15:     if ScalarProduct(tmp.key, E_b ) > 0 then
16:       if tmp.position == pos then return
 **Case2**
17:   if fNode is not minNode then
18:     if isFound == false then fNode = maxNode
19:     else fNode = objAVLTree.findPredecessor(fNode)
20:     if fNode.position == pos then return
 **Case3**
21:   if isFound == true && ScalarProduct(fNode.key, E_b ) == 0 then
22:     fNode.setPosition(pos)
23:     return
 **Case4**
24:   objAVLTree.insert( E_v ,pos, E_b )
25:   return
```

Заключение.

В данном документе представлена новая упрощенная, основанная на линейной алгебре схема шифрования, причем эта схема предусматривает (1) обработку запроса диапазона в отношении зашифрованных числовых данных, полученных из внешних источников в облаке, и тем самым (2) инкрементное адаптивное индексирование, в результате чего индексируются только те данные, которые запрашиваются доверенными клиентами, которые стали индексированными. Схема представляет числовые значения в виде коротких векторов и опирается на простые операции линейной алгебры для шифрования и дешиф-

рования; это не позволяет раскрывать ни фактические значения данных, ни их порядок. Хотя структура индекса может раскрывать порядок в долгосрочной перспективе, это происходит только после выполнения важных операций индексирования; кроме того, в схеме предложен дополнительный компонент скрытия, этот компонент намеренно вносит неоднозначность в предложенные построения путем обеспечения двух различных интерпретаций каждого вектора зашифрованных значений. Схема обеспечивает безопасность, необходимую для критичных по времени операций, таких как высокочастотная торговля и обработка финансовых транзакций в облаке.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. Способ поиска в базе данных, содержащий этапы, на которых

выполняют на сервере базы данных шифрование множества значений данных в базе данных с использованием первого режима шифрования для получения зашифрованных значений данных;

выполняют на клиентской машине шифрование связанного значения с использованием второго режима шифрования для получения зашифрованного связанного значения, причем второй режим шифрования отличается от первого режима шифрования;

принимают от клиентской машины на сервере базы данных зашифрованное связанное значение;

сравнивают на сервере базы данных связанное значение с множеством значений данных с использованием зашифрованного связанного значения и зашифрованных значений данных без дешифрования зашифрованного связанного значения; и

передают на клиентскую машину от сервера базы данных результат сравнения, причем результат сравнения содержит набор зашифрованных значений данных;

причем значения, зашифрованные с использованием первого режима шифрования, невозможно сравнивать друг с другом.

2. Способ по п.1, в котором на этапе сравнения связанного значения с множеством значений данных с использованием зашифрованных связанных значений и зашифрованного значения данных сравнивают связанное значение с множеством значений данных с использованием операции над зашифрованными связанными значениями и зашифрованными значениями данных.

3. Способ по п.1, в котором

одно или более зашифрованных значений данных представляют собой значения ключа, используемые для индексирования зашифрованных значений данных,

при этом на этапе сравнения связанного значения с множеством значений данных сравнивают связанное значение с множеством значений данных с использованием зашифрованных связанных значений, значений ключа и зашифрованных значений данных.

4. Способ по п.1, в котором

на этапе шифрования множества значений данных в базе данных преобразуют каждое из множества значений данных в вектор значений;

на этапе шифрования связанного значения преобразуют связанное значение в связанный вектор и

на этапе сравнения связанного значения с множеством значений данных получают скалярное произведение векторов значений на связанный вектор.

5. Способ по п.4, в котором

каждый из векторов значений содержит зашумленный подвектор значений;

связанный вектор содержит зашумленный связанный подвектор;

причем зашумленный связанный подвектор ортогонален каждому из зашумленных подвекторов значений.

6. Способ по п.4, дополнительно содержащий этапы, на которых

выбирают случайный множитель для каждого из векторов значений и

масштабируют каждый вектор значений с помощью соответствующего выбранного множителя.

7. Способ по п.5, дополнительно содержащий этапы, на которых

выбирают первый случайный положительный множитель и второй случайный положительный множитель для каждого из векторов значений;

масштабируют каждый зашумленный подвектор значений с помощью соответствующего первого случайного множителя и

масштабируют остальную часть каждого из векторов значений с помощью соответствующего второго случайного положительного множителя.

8. Способ по п.4, дополнительно содержащий этап, на котором

выбирают обратимую матрицу; при этом

на этапе преобразования каждого из значений данных в вектор значений преобразуют значения данных в вектор-столбец значений и умножают одну из обратной матрицы и транспонированной матрицы на вектор-столбец значений; и

на этапе преобразования связанного значения в связанный вектор преобразуют связанное значение в связанный вектор-столбец и умножают другую из обратной матрицы и транспонированной матрицы на

связанный вектор-столбец.

9. Способ индексирования значений данных, содержащий этапы, на которых

выполняют шифрование множества значений данных с использованием первого режима шифрования для получения зашифрованных значений данных;

выполняют шифрование одного или более связанных значений с использованием первого режима шифрования для получения первых зашифрованных связанных значений и с использованием второго режима шифрования для получения вторых зашифрованных связанных значений, причем второй режим шифрования отличается от первого режима шифрования;

сравнивают указанное одно или более связанных значений с множеством значений данных с использованием вторых зашифрованных связанных значений и зашифрованных значений данных без дешифрования вторых зашифрованных связанных значений; и

индексируют зашифрованные значения данных на основании упомянутого сравнения с использованием первых зашифрованных связанных значений в качестве значений ключа,

при этом значения, зашифрованные с использованием первого режима шифрования, невозможно сравнивать друг с другом; и

значения, зашифрованные с использованием второго режима шифрования, невозможно сравнивать друг с другом.

10. Способ по п.9, дополнительно содержащий этапы, на которых

выполняют шифрование другого связанного значения с использованием первого режима шифрования для получения другого первого зашифрованного связанного значения и с использованием второго режима шифрования для получения другого второго зашифрованного связанного значения;

сравнивают указанное другое связанное значение с множеством значений данных с использованием указанного другого второго зашифрованного связанного значения, значений ключа и зашифрованных значений данных;

добавляют указанное другое первое зашифрованное связанное значение в качестве другого значения ключа на основании упомянутого сравнения.

11. Способ по п.9, в котором на этапе индексирования зашифрованных значений данных выполняют построение иерархической структуры индексов, причем узлы в иерархической структуре индексов представляют собой значения ключа.

12. Способ по п.11, в котором иерархическая структура индексов представляет собой двоичное дерево поиска.

13. Способ по п.12, в котором двоичное дерево поиска представляет собой АВЛ-дерево.

14. Способ по п.9, дополнительно содержащий этап, на котором добавляют неоднозначность по меньшей мере в одно из значений данных с тем, чтобы стало возможным множество интерпретаций для каждого из указанного по меньшей мере одного значения данных, при этом только одна интерпретация из множества интерпретаций является истинной.

15. Способ по п.9, дополнительно содержащий этап, на котором добавляют неоднозначность по меньшей мере в одно из указанного одного или более связанных значений с тем, чтобы стало возможным множество интерпретаций для указанного по меньшей мере одного из одного или более связанных значений, при этом только одна интерпретация из множества интерпретаций является истинной.

16. Способ по п.9, в котором

на этапе шифрования множества значений данных с использованием первого режима шифрования преобразуют значения данных в векторы значений;

на этапе шифрования одного или более связанных значений с использованием первого режима шифрования преобразуют связанные значения в первые связанные векторы;

на этапе шифрования одного или более связанных значений с использованием второго режима шифрования преобразуют связанные значения во вторые связанные векторы и

на этапе сравнения указанного одного или более связанных значений с множеством значений данных получают скалярные произведения векторов значений на вторые связанные векторы.

17. Сервер индексирования значений данных, содержащий

средство хранения, выполненное с возможностью хранения зашифрованных значений данных, причем зашифрованные значения данных представляют собой множество значений данных, зашифрованных с использованием первого режима шифрования;

блок приема, выполненный с возможностью приема от клиента первых зашифрованных связанных значений и вторых зашифрованных связанных значений, причем первые зашифрованные связанные значения представляют собой одно или более связанных значений, зашифрованных с использованием первого режима шифрования, а вторые зашифрованные связанные значения представляют собой одно или более связанных значений, зашифрованных с использованием второго режима шифрования, при этом второй режим шифрования отличается от первого режима шифрования; и

блок обработки, выполненный с возможностью

сравнения одного или более связанных значений с множеством значений данных с использованием вторых зашифрованных связанных значений и зашифрованных значений данных без дешифрования вто-

рых зашифрованных связанных значений; и

индексирования зашифрованных значений данных на основании упомянутого сравнения с использованием первых зашифрованных связанных значений в качестве значений ключа,

при этом зашифрованные значения данных невозможно сравнивать друг с другом.

18. Сервер по п.17, в котором

блок приема дополнительно выполнен с возможностью приема другого первого зашифрованного связанного значения и другого второго зашифрованного значения, причем другое первое зашифрованное связанное значение является другим связанным значением, зашифрованным с использованием первого режима шифрования, а другое второе зашифрованное значение является другим связанным значением, зашифрованным с использованием второго режима шифрования; а

блок обработки дополнительно выполнен с возможностью

сравнения другого связанного значения с множеством значений данных с использованием указанного другого второго зашифрованного связанного значения, значений ключа и зашифрованных значений данных; и

добавления другого первого зашифрованного связанного значения в качестве другого значения ключа на основании упомянутого сравнения,

при этом сервер дополнительно содержит блок отправки, выполненный с возможностью отправки результата упомянутого сравнения клиенту.

19. Клиентская машина для приема результата поиска, содержащая

блок отправки, выполненный с возможностью отправки второго зашифрованного связанного значения на сервер, причем второе зашифрованное связанное значение является связанным значением, зашифрованным с использованием второго режима шифрования, причем сервер содержит зашифрованные значения данных, представляющие собой множество значений данных, зашифрованных с использованием первого режима шифрования;

блок приема, выполненный с возможностью приема от сервера результата сравнения между связанным значением и множеством значений данных с использованием второго зашифрованного связанного значения и зашифрованных значений данных без дешифрования второго зашифрованного связанного значения, причем результат содержит набор зашифрованных значений данных; и

блок обработки, выполненный с возможностью дешифрования набора зашифрованных значений данных,

при этом зашифрованные значения данных невозможно сравнивать друг с другом.

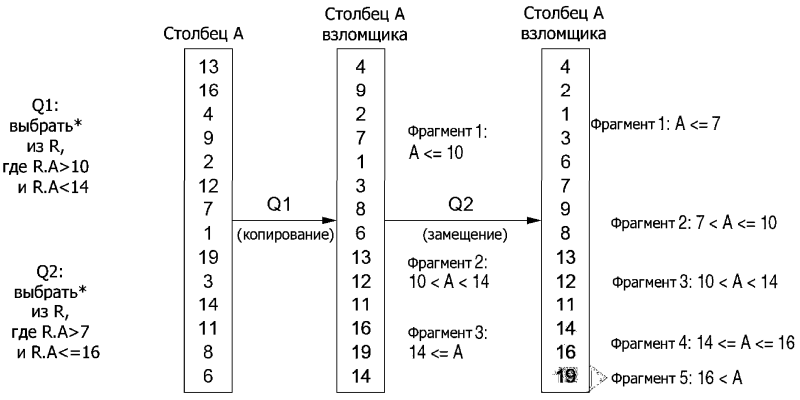
20. Клиентская машина по п.19, в которой блок отправки дополнительно выполнен с возможностью отправки первого зашифрованного связанного значения на сервер, причем первое зашифрованное связанное значение является связанным значением, зашифрованным с использованием первого режима шифрования.

21. Клиентская машина по п.20, в которой блок обработки выполнен с возможностью шифрования связанного значения с использованием второго режима шифрования для получения второго зашифрованного связанного значения и шифрования указанного связанного значения с использованием первого режима шифрования для получения первого зашифрованного связанного значения.

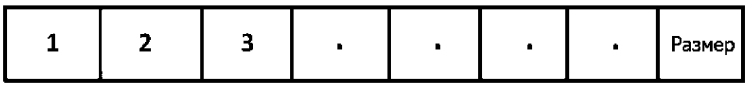
22. Клиентская машина по п.19, в которой

по меньшей мере одно из множества значений данных имеет неоднозначность, так что для каждого, по меньшей мере, из некоторых из множества значений данных возможно множество интерпретаций, причем только одна интерпретация из множества интерпретаций является истинной;

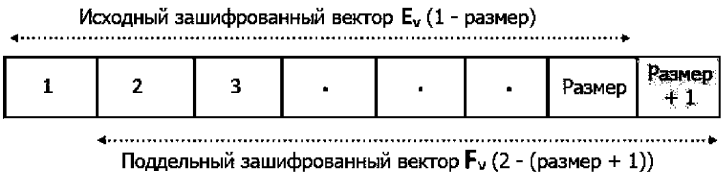
при этом процессор выполнен с возможностью извлечения истинных интерпретаций значений данных из набора зашифрованных значений данных.



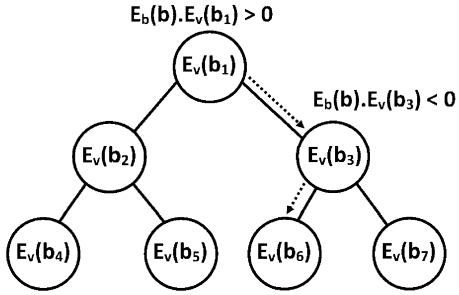
Фиг. 1



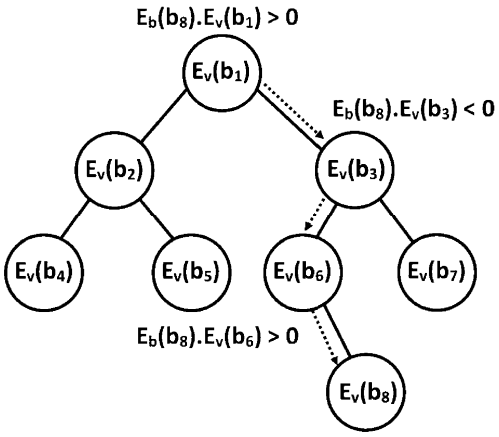
Фиг 2



Фиг. 3



Фиг. 4



Фиг. 5