УДК 004

Обзор средств контроля целостности отчужденных вычислений (*Обзор*)

И. В. Марков
Московский физико-технический институт (государственный университет),
г. Долгопрудный, Московская обл., Россия

Структура вычислительного центра облачного провайдера сложна и неоднородна, что ставит под сомнение постоянное корректное исполнение любого вычисления. В конечном счете публичный вычислительный центр (ПВЦ) — это черный ящик; программное обеспечение таких облаков, как правило, предоставляется с закрытым исходным кодом. Поэтому сбои, к которым могут привести неверная конфигурация, искажение данных при хранении или перемещении, проблемы с оборудованием, действия злоумышленников, сложно обнаружить. Из этого следует один из основных вопросов отчужденных вычислений: как обеспечить проверку результатов, вычисленных на недоверенной стороне? Проводится обзор возможных решений обеспечения контроля целостности отчужденного выполнения сервисов.

Ключевые слова: проверяемые вычисления, облачные вычисления, гомоморфное шифрование.

С развитием технологии облачных вычислений все больше предприятий переносит свои приложения из частных вычислительных центров (ЧВЦ) в публичные (ПВЦ). За счет данного переноса предприятия пытаются получить следующие преимущества публичного облачного сервиса над частным:

- доступность;
- низкая стоимость поддержки инфраструктуры вычислительного центра;
- надежность.

В то же время при переносе приложения или сервиса в публичное облако в общем случае предприятие не имеет оснований доверять вычислительным ресурсам ПВЦ. При эксплуатации ЧВЦ у сотрудников есть возможность контролировать работу каждого физического ресурса, управлять потоками данных. В случае ПВЦ имеется только интерфейс доступа к выделенным физическим ресурсам, предоставляемый провайдерами. Предприятие отправляет запрос на удаленный сервис ПВЦ для выполнения определенного вычисления на предоставленных предприятием входных данных. Однако из-за того, что у предприятия есть только доступ к виртуальным ресурсам, заказчик вычисления не может быть уверен, был ли запущен нужный ему процесс и были ли вычисления корректно выполнены. Для этого необходимы дополнительные средства, обеспечивающие контроль выполнения требуемых вычислений. Проводится обзор возможных решений обеспечения контроля целостности отчужденного выполнения сервисов. В первом разделе описываются первые теоретические попытки разрешения указанного противоречия. Во втором разделе речь идет о подходе, на который опираются основные исследования в этой области, и методах, реализующих данный подход. В третьем разделе проводятся обзор и сравнение реализованных систем, основанных на описанных методах, по указанным ранее требованиям.

Марков Илья Валерьевич, студент кафедры "Защита информации".
E-mail: markovilya197@gmail.com

Статья поступила в редакцию 13 июня 2018 г.

Первые теоретические решения

Тема проверяемых вычислений изучалась многими исследователями на протяжении нескольких последних десятилетий. Основная формулировка проблемы звучит следующим образом: как обеспечить проверку результатов, вычисленных на недоверенной стороне?

Самый простой ответ на данный вопрос, который и пытались использовать изначально для решения поставленной проблемы, это реплицировать [1], повторять вычисления на нескольких разных серверах. Однако репликация подразумевает, что ошибки вычислений не скоррелированы, не могут произойти одновременно на нескольких узлах. Это может оказаться неверным предположением. Также существуют подходы, основывающиеся на доверенных вычислениях [2] или доверенном аппаратном обеспечении [3]. Такие технологии подразумевают наличие доверенных узлов вычислений, а также тот факт, что аппаратное обеспечение или гипервизор работают корректно [4]. На практике эти предположения также оказываются неверными.

Исследователи обращают внимание на другой подход. Он заключается в перенесении ответственности за предоставление доказательства на сторону, осуществляющую вычисление (*proof based verifiable computing PBVC*). Этот метод избавляет от предположений об аппаратном обеспечении, от необходимости иметь доверенные узлы исполнения на стороне вычислений. Реализация данной идеи основывается на нетривиальных технологиях Probabilistically Checkable Proofs (PCP [5]), вероятностно проверяемое доказательство) или Fully Homomorphic Encryption [6] (FHE, полное гомоморфное шифрование). Эти технологии напрямую невозможно использовать на практике из-за их слишком низкой производительности и требования большого количества оперативной памяти. В данном подходе в отличие от предыдущих решений нет не отражающих реальность предположений о надежности аппаратного или программного обеспечения, о возможности доверять некоторым узлам системы.

Все идеи, основанные на PBVC, — теоретические. Процессы верификации при их реализации очень сложны. Начиная с 2013 г. появилось несколько проектов, которые поменяли общее мнение о безнадежности proof-based verifications. Эти проекты представляют собой системы, позволяющие выполнять верификацию некоторых функций за разумное время.

Структура PBVC

Опишем общую структуру системы проверяемых вычислений. Различные криптографические допущения или криптографические методы, применяемые в технологиях, реализующих PBVC, довольно сильно отличаются друг от друга, но их объединяет основной принцип построения архитектуры системы.

Системы, реализующие данный подход, состоят из двух сторон: проверяющая (*verifier*) и вычисляющая или доказывающая (*prover*). Схема общего вида протокола показана на рис. 1. Верификационная часть работает с логическими схемами, в которые конвертируются необходимые операции и функции. Таким образом, *prover* выполняет вычисления (п. 3 на рис. 1) логической схемы C на входных данных x и выдает на выходе результат y. При выполнении шага 3 *prover* вычисляет результат, и от него ожидается получение корректной записи $\{C, x, y\}$, где x — входные данные; y — результат; C — логическая схема. Важно отметить, что если $y \neq F(x)$, то корректной записи для такой пары x, y просто не существует. На этапе 4 возможны различные схемы протоколов верификации.

Схема с интерактивным взаимодействием. Verifier посылает запросы по одному и в финальном раунде выносит решение, была ли переданная запись верна. Этот подход основывается на интерактивных протоколах доказательства [7—9]. Самая известная работа, представляющая этот подход (названный протоколом GKR), описана в упомянутой ранее статье Muggles [10], которая позднее была улучшена и дополнена [11—14]. В этом решении возможна потоковая обработка данных.

Схема с извлекаемым обязательством. Доказательство представляется в виде линейной функции. Протокол верификации исполняется в два раунда. Сначала prover фиксирует запись с помощью криптографических примитивов, называемых криптографическим обязательством (*cryptographic commitment*). Затем verifier генерирует запросы по закодированному результату — просит выдать ему

Рис. 1. Схема общего вида протокола PBVC [20]

некоторые значения функции. Prover отвечает на эти запросы в соответствии со сгенерированным в первом раунде обязательством. Данное решение описано в теории работы Killian [15], позднее улучшенной Ishai с соавторами [16] в проекте IKO.

Скрытые запросы. В данной схеме verifier предварительно зашифровывает свои запросы (как и предыдущем методе, запрос — некоторые значения функции, которые будет впоследствии проверять verifier) и посылает их перед запросом на вычисление. Затем во время проверки с помощью криптографических методов достигается следующее: prover отвечает на запросы verifier, будучи неспособным понять, какие места encoding запрашиваются, а verifier восстанавливает ответы на свои запросы. Это решение рассмотрено в работе Gennaro [17], а затем дополнено и реализовано в проектах [18, 19].

Реализации

Описанные техники более производительны, чем решения, появившиеся в самом начале исследований данной проблемы, но при интеграции этих идей в системы, от которых необходимо выдерживать реальные нагрузки и работать с реальными данными, возможны некоторые вариации. Далее описываются реализованные системы с опубликованными результатами измерения эффективности.

CMT. Этот проект входит в ряд работ, реализующих первый подход PCP, описанный Goldwasser [10]. Для класса функций, к которым применима CMT [11], эффективность системы вычисления и проверки по сравнению с прошлыми теоретическими результатами выше. Более того, CMT спроектирована для поточных вычислений, при которых verifier и prover забывают прошлые входные данные по мере прихода новых. Но класс вычислений, с которыми может работать CMT, сильно ограничен: логическая схема должна иметь блоки, подходящие под определенный шаблон, и не иметь блоков порядкового сравнения. Дальнейшая разработка в проекте Allspice [14] сохранила эффективность и расширила класс функций, к которым применимо решение.

Pepper, Ginger, Zaatar. Эти проекты входят в ряд работ, реализующих второй метод PCP. Pepper [21] и Ginger [22] представляют вычисление в виде системы уравнений над конечным полем. Утверждается, что это представление более емко, чем логические схемы в IKO или арифметические схемы в CMT. Оба проекта используют механизм криптографического обязательства, применяют несколько системных оптимизаций, таких как распараллеливание на распределенных системах. В обеих системах класс обрабатываемых функций шире, чем у CMT. Zaatar [23] — продолжение разработки Ginger с заменой PCP на линейную PCP. Эта модификация привела к улучшению производительности verifier.

Pinocchio. Pinocchio [24] представляет собой реализацию протокола GGPR [17], применяющего третий подход PCP (скрытые запросы). GGPR может быть рассмотрен как вероятностно проверяемое кодирование с добавочным уровнем криптографии. Кодирование в GGRP — более емкое, что приводит к уменьшению накладных расходов на prover и verifier. Криптография, используемая в этом проекте, предоставляет также и следующие преимущества. Она дает возможность скрывать запросы, позволяя переиспользовать их. В результате это дает возможность уменьшить число сообщений при исполнении протокола. Pinocchio уменьшает стоимость подготовительных операций не для набора входных данных, а для всех входных данных для одной функции. Компилятор Pinocchio первым начал принимать программу с C-синтаксисом. Также Pinocchio поддерживает дополнительные требования нулевого разглашения и публичной верификации.

BCGTV. BCGTV [18] компилирует программы на C в новое представление логических схем [25]. BCGTV объединяет свои логические схемы с механизмами вероятностных доказательств из Pinocchio. Логическая схема BCGTV представляет собой исполнение на MIPS-подобной архитектуре общего назначения, называемой TinyRAM. Эти схемы позволяют использовать такие важные свойства высокоуровневого программирования, как циклы переменной длины, потоки управления, работа с оперативной памятью.

Pantry. Pantry [26] расширяет модель вычислений Pinocchio, позволяет работать с оперативной памятью, объединяет в своей модели подходы скрываемых запросов и извлекаемых обязательств. Каждая операция в этой модели над ее состоянием конвертируется в вычисление хэш-функции, что выливается в дорогостоящие операции. Имеется возможность потоковой обработки данных.

Сравнение систем

Распространенными методами проверяемых вычислений являются PCP, FHE, которые обеспечивают высокую точность проверки. Некоторые схемы, основанные на PCP, предназначены для решения конкретных функций, таких как сложение и умножение, и не могут поддерживать общие функции при обеспечении высокой эффективности проверки. Однако в этих схемах не реализована публичная проверка с нулевым уровнем знаний. После

появления гомоморфного шифрования проверяемые вычисления могут поддерживать обобщенный класс функций проще, чем до него. Для схемы на основе FHE в теоретических исследованиях были предложены более совершенные варианты, которые могут поддерживать неинтерактивность, нулевую разглашаемость и даже публичную проверку. Однако из-за неэффективности FHE, высокого потребления памяти и высокой цены вычисления такие схемы трудно применить на практике. В случае, например, Pinocchio эту открытую проблему пытались решить путем балансирования требования корректности и эффективности другими методами (введение QAP, использование скрытых запросов), и она имела лучшую производительность, чем Zaatar. В BCGTV, Pantry, Buffet было введено особое представление логических схем, что позволило значительно расширить диапазон функций, поддерживаемых этими системами. Одним из важнейших пунктов при сравнении систем является производительность их компонент. В работе [15] приведены результаты бенчмарков систем, реализованных исследовательской группой авторов, которые соответствуют результатам, полученным в оригинальных работах. Все эксперименты выполнялись на одном и том же оборудовании (Intel Xeon E5-2680 процессор, 2.7 Ghz, 32GB RAM) с prover на одной машине и verifier на других. Результаты BCGTV экстраполированы, так как на современном аппаратном обеспечении использование TinyRAM на этих бенчмарках ограничено.

На рис. 2 показаны результаты работы верификатора для различных систем. Для того чтобы система имела смысл, время верификации должно быть меньше времени исполнения задачи. Как видно из рис. 2, почти все системы добиваются поставленной задачи, но время исполнения верификационной части не отличается на порядки от времени исполнения локального вычисления.

На рис. 3 показаны результаты изменения времени исполнения функции на стороне prover. Можно отметить, что современные системы проверяемых вычислений пока сильно отстают от желаемого уровня вычисления без верификации.

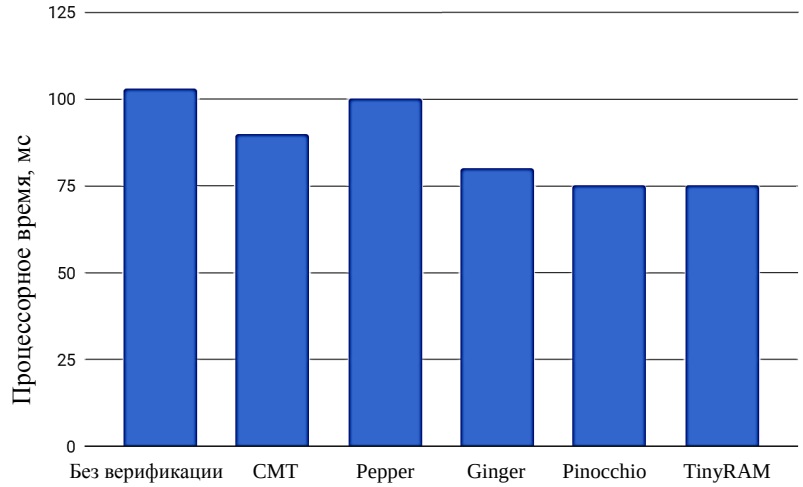

Рис. 2 Производительность верификатора для различных систем

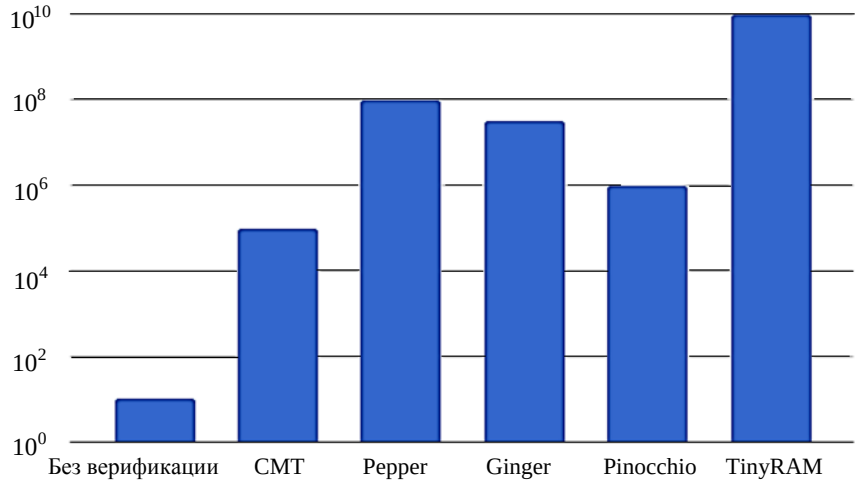

Рис. 3 Время вычисления на prover относительно времени исходного вычисления

Выводы

Различия в производительности разных систем незаметны на фоне общей проблемы производительности решений в данной области исследований. Теоретические исследования в этой области вообще не подразумевают практическое применение и прямые реализации протоколов, описанных в самом начале теоретических исследований, запущенные даже на простых функциях, заняли бы многие миллионы лет. Современные системы позволяют работать со сложными функциями за разумное время. Но их производительность все еще отстает на несколько порядков от производительности, при которой имеет смысл использовать облачные системы. Тем не менее эти системы могут быть полезны в некоторых сценариях применения. Например, для систем, требующих высокой надежности вычислений, может быть разумным пожертвовать производительностью в пользу уверенности в том, что удаленная машина работает корректно. Таким образом, текущее состояние исследований не позволяет говорить о переводе всевозможных программ в сервис контроля исполнения, но позволяет реализовывать системы, эффективно работающие на простых функциях. Можно предположить, что необходимо проверять возможность использования системы проверки вычислений для каждого сервиса индивидуально, т. е. необходимо найти соответствие между возможностями существующих на текущий момент решений и технологий, обеспечивающих контроль отчужденного исполнения сервисов, и свойствами и требованиями систем, которые потенциально могут эти решения эксплуатировать.

Литература

1. *Canetti R., Riva B., Rothblum G. N.* Practical delegation of computation using multiple servers: Proceedings of the 18th ACM Conference on Computer and Communications Security, 2011. P. 445—454.
2. *Li W., Xue K., Xue Y., Hong J.* TMACS: A Robust and Verifiable Threshold Multi-Authority Access Control System in Public Cloud Storage: IEEE Transactions on Parallel and Distributed Systems, 2016. V. 27 (5). P. 1484—1496.
3. *Schiffman J., Sun Y., Vijyakumar H., Jaeger T.* Cloud Verifier: Verifiable Auditing Service for IaaS Clouds: IEEE Ninth World Congress on Services, 2013. P. 1.
4. *Sadeghi A. R., Schneider T., Winandy M. A. R.* Token-based cloud computing: Secure outsourcing of data and arbitrary computations with lower latency: Proceedings of the 3rd International Conference on Trust and Trustworthy Computing, 2010. P. 417—429.
5. *Lai J., Deng R. H., Pang H., Weng J.* Verifiable computation on outsourced encrypted data: Proceedings of European Symposium on Research in Computer Security, 2014. P. 273—291.
6. *Gentry C.* A fully homomorphic encryption scheme: PhD dissertation. CA. Stanford: Stanford, University, 2009.
7. *Babai L.* Trading group theory for randomness: Proceedings of the 17th annual ACM symposium on Theory of computing, 1985. P. 421—429.
8. *Goldwasser S., Micali S., Rackoff C.* The knowledge complexity of interactive proof systems: Proceedings of the 17th annual ACM symposium on Theory of computing, 1985. P. 291—304.
9. *Lund C., Fortnow L., Karloff H. J., Nisan N.* Algebraic methods for interactive proof systems: JACM. 1992. V. 39 (4). P. 859—868.
10. *Goldwasser S., Kalai Y. T., Rothblum G. N.* Delegating computation: Interactive proofs for muggles: Proceedings of the Annual ACM Symposium on Theory of Computing, 2008. V. 62 (4). P. 113—122.
11. *Cormode G., Mitzenmacher M., Thaler J.* Practical verified computation with streaming interactive proofs: Proceedings of ITCS, 2012. P. 90—112.
12. *Thaler J.* Time-optimal interactive proofs for circuit evaluation: Lecture Notes in Computer Science, 2013. V. 8043. P. 351—401.
13. *Thaler J., Roberts, M. Mitzenmacher M., Pfister H.* Verifiable computation with massively parallel interactive proofs: Proceedings of USENIX HotCloud Workshop, 2012. P. 22—30.
14. *Vu V., Setty S., Blumberg A.J., Walfish M.* Hybrid architecture for interactive verifiable computation: IEEE Symposium on Security and Privacy, May 2013.
15. *Kilian J.* A note on efficient zero-knowledge proofs and arguments: In Proceedings of STOC, 1992. P. 723—732.
16. *Ishai Y., Kushilevitz E., Ostrovsky R.* Efficient arguments without short PCPs: Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity, 2007. P. 278—291.
17. *Gennaro R., Gentry C., Parno B., Raykova M.* Quadratic span programs and succinct NIZKs without PCPs: Proceedings of the 19th International Conference on Advances in Cryptology, 2013. Part I. V. 8269. P. 41—60.
18. *Ben-Sasson E., Chiesa A., Genkin D., Tromer E.* SNARKs for C: Verifying program executions succinctly and in zero knowledge: Proceedings of CRYPTO, 2013.
19. *Ben-Sasson E., Chiesa A., Tromer E., Virza M.* Succinct non-interactive zero knowledge for a von Neumann architecture: Proceedings of the 23rd USENIX Conference on Security Symposium, 2014. P. 781—796.
20. *Walfish M., Blumberg A. J.* Verifying computations without reexecuting them: from theoretical possibility to near practicality // Communications of the ACM (CACM), 2015. V. 58 (2). P. 74—84.
21. *Setty S., McPherson R., Blumberg A. J., Walfish M.* Making argument systems for outsourced computation practical (sometimes): Network and Distributed System Security Symposium, 2012.
22. *Setty S., Vu V., Panpalia N., Braun B., Blumberg A. J., Walfish M.* Taking proof-based verified computation a few steps closer to practicality: Proceedings of the 21st USENIX Conference on Security Symposium, 2012. P. 12—20.
23. *Setty S., Braun B., Vu V., Blumberg A. J., Parno B., Walfish M.* Resolving the conflict between generality and plausibility in verified computation: Proceedings of the 8th ACM European Conference on Computer Systems, 2013. P. 71—84.
24. *Parno B., Gentry C., Howell J., Raykova M.* Pinocchio: Nearly practical verifiable computation: Proceedings of the 2013 IEEE Symposium on Security and Privacy, 2013. P. 238—252.
25. *Ben-Sasson E., Chiesa A., Genkin D., Tromer E.* Fast reductions from RAMs to delegatable succinct constraint satisfaction problems: Proceedings of the 4th conference on Innovations in Theoretical Computer Science, 2013. P. 401—414.
26. *Braun B., Feldman A.J., Ren Z., Setty S., Blumberg A. J., Walfish M.* Verifying computations with state: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, 2013. P. 341-357.

Review of verifiable computation

(*Review*)

I. V. Markov
Moscow Institute of Physics and Technology (State University),
Dolgoprudny, Moscow region, Russia

The structure of the cloud provider's computing center is complex and heterogeneous, which makes the constant correct execution of any calculation doubtful. Ultimately, the cloud provider system is a black box, the software of such clouds is usually provided with closed source code. Therefore, it is difficult to detect failures, which can be caused by incorrect configuration, data corruption, equipment problems or malicious actions. This leads to one of the main issues of cloud computing: how to check the verification of the results calculated on the untrusted side? The paper reviews possible solutions to ensure the integrity control of the alienated services.

Keywords: verifiable computing, cloud computing, homomorphic encryption.

Bibliography — 26 references. *Received June 13, 2018*