РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 762 141** (13) **C2**

(51) МПК
G06F 21/62 (2013.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06F 21/53 (2021.05); G06F 21/57 (2021.05); G06F 21/64 (2021.05); G06F 21/74 (2021.05); H04L 9/0643 (2021.05); H04L 9/3247 (2021.05)

(21)(22) Заявка: **2019126638, 20.12.2017**

(24) Дата начала отсчета срока действия патента:
20.12.2017

Дата регистрации:
16.12.2021

Приоритет(ы):
(30) Конвенционный приоритет:
24.01.2017 US 15/414,355

(43) Дата публикации заявки: **26.02.2021** Бюл. № **6**

(45) Опубликовано: **16.12.2021** Бюл. № **35**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **26.08.2019**

(86) Заявка PCT:
US 2017/067451 (20.12.2017)

(87) Публикация заявки PCT:
WO 2018/140160 (02.08.2018)

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, строение 3, ООО "Юридическая фирма Городисский и Партнеры"

(72) Автор(ы):
КОСТА, Мануэль (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ ТЕКНОЛОДЖИ ЛАЙСЕНСИНГ, ЭлЭлСи (US)

(56) Список документов, цитированных в отчете о поиске: **JOHN P MECHALAS ET AL. "INTEL SOFTWARE GUARD EXTENSIONS TUTORIAL SERIES: Part 1, Intel SGX Foundation", опубл. 07.07.2016, Найдено в Интернет по адресу:https://https://software.intel.com/content/www/us/en/develop/articles/intel-software-guard-extensions-tutorial-part-1-foundation.html. US 2015/0121536 A1, 30.04.2015. US 9514317 B2, 06.12.2016. WO** (см. прод.)

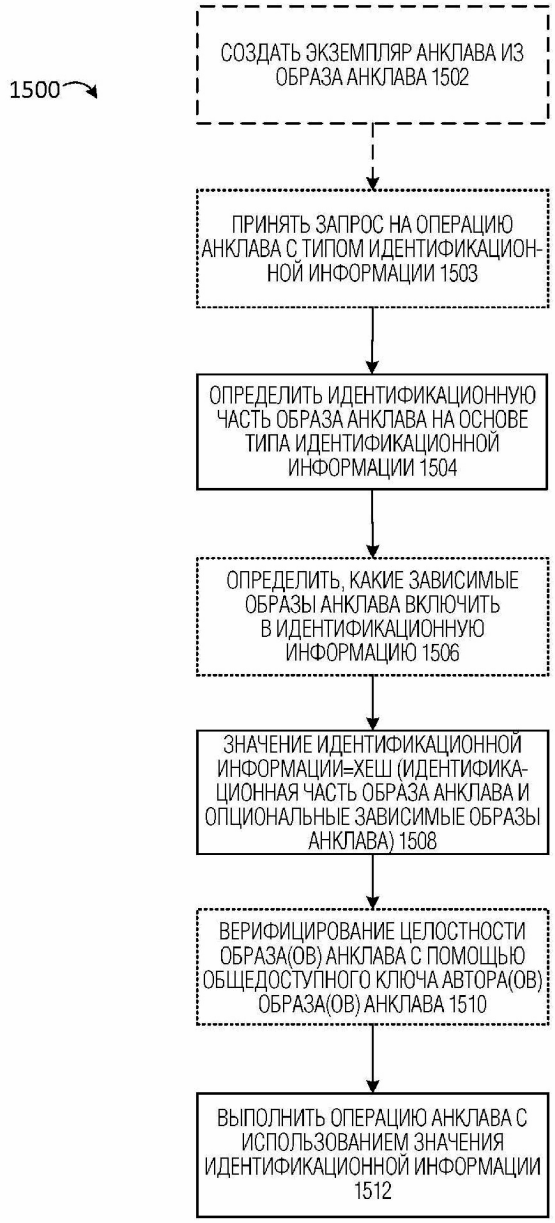
(54) АБСТРАКТНАЯ ИДЕНТИФИКАЦИЯ АНКЛАВА

(57) Реферат:

Группа изобретений относится к защищенным вычислительным системам. Техническим результатом является повышение безопасности и конфиденциальности данных. Способ содержит этапы: прием типа идентификационной информации и запроса на операцию, связанную с анклавом, экземпляр которого создан, при этом образ анклава, из которого был создан экземпляр анклава, включает в себя ссылки на дополнительные зависимые образы анклава; определение того, какие из этих дополнительных образов анклава должны быть включены, в качестве ввода в хеш-функцию, на основе того, что каждый дополнительный образ анклава включается в идентификационную часть образа анклава; вычисление хеш-функции на основе типа идентификационной информации и информации, извлеченной из образа анклава, чтобы обеспечить, в качестве вывода хеш-функции, значение идентификационной информации для анклава, причем целостность значения идентификационной информации является

RU 2762141 C2

верифицируемой путем верифицирования подписи в образе анклава с помощью общедоступного ключа, связанного с автором образа анклава; и выполнение упомянутой операции с этим значением идентификационной информации. 3 н. и 19 з.п. ф-лы, 23 ил.



ФИГ. 15

(56) (продолжение):
2014196966 A1, 11.12.2014. RU 2602793 C2, 20.11.2016. RU 2599340 C2, 10.10.2016.

RU 2762141 C2

RU 2762141 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 762 141** (13) **C2**

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

(12) ABSTRACT OF INVENTION

(52) CPC
G06F 21/53 (2021.05); G06F 21/57 (2021.05); G06F 21/64 (2021.05); G06F 21/74 (2021.05); H04L 9/0643 (2021.05); H04L 9/3247 (2021.05)

(21)(22) Application: **2019126638, 20.12.2017**

(24) Effective date for property rights:
20.12.2017

Registration date:
16.12.2021

Priority:
(30) Convention priority:
24.01.2017 US 15/414,355

(43) Application published: **26.02.2021** Bull. № **6**

(45) Date of publication: **16.12.2021** Bull. № **35**

(85) Commencement of national phase: **26.08.2019**

(86) PCT application:
US 2017/067451 (20.12.2017)

(87) PCT publication:
WO 2018/140160 (02.08.2018)

Mail address:
129090, Moskva, ul. B.Spasskaya, 25, stroenie 3, OOO "Yuridicheskaya firma Gorodisskij i Partnery"

(72) Inventor(s):
COSTA, Manuel (US)

(73) Proprietor(s):
MICROSOFT TECHNOLOGY LICENSING, LLC (US)

RU 2762141 C2

(54) ABSTRACT ENCLAVE IDENTIFICATION

(57) Abstract:

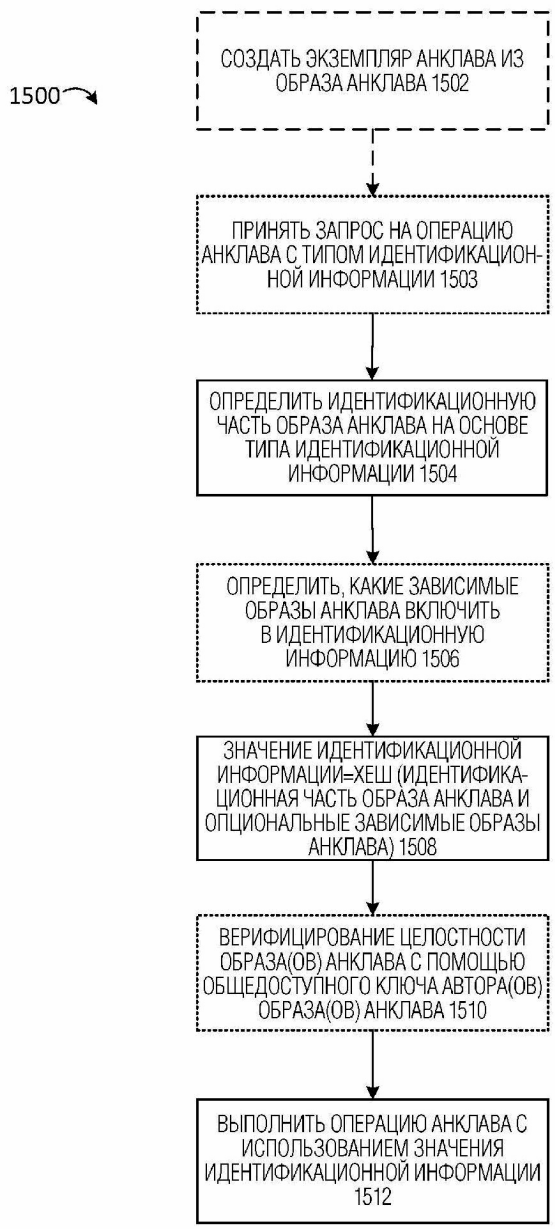
FIELD: protected computing systems.

SUBSTANCE: method contains stages of: receiving a type of identification information and a request for an operation related to an enclave, an instance of which was created, while an enclave image, of which the enclave instance was created, includes links to additional dependent enclave images; determining, which of these additional enclave images should be included as an input to a hash function, based on the fact that each additional enclave image is included in an identification part of the enclave image; calculating the hash function based on the type of identification information and information extracted from the enclave image to provide, as an output of the hash function, a value of identification information for the enclave, wherein the integrity of the value of identification information is verifiable by verifying a signature in the enclave image using a public key associated with an author of the enclave image; and performing the specified operation with this value of identification information.

EFFECT: increase in safety and data confidentiality.

22 cl, 23 dwg

RU 2762141 C2



ФИГ. 15

RU 2762141 C2

RU 2762141 C2

Область техники

[0001] Настоящее раскрытие относится к безопасным вычислительным системам.

Предшествующий уровень техники

[0002] Безопасные изолированные области или доверенные среды исполнения обеспечивают безопасный контейнер, упоминаемый как анклав в настоящем документе, для исполнения доверенного кода на компьютере, который может также иметь менее доверенный код в области вне изолированной области. Изолированная область анклава включает в себя участок памяти, который защищен во время исполнения кода, находящегося вне анклава. Изолированная память может содержать как код, так и данные для анклава, и защита этой памяти может включать в себя ограничения на исполнение кода, содержащегося в памяти анклава, в дополнение к ограничениям на считывание из или запись в память анклава. Аспекты безопасности анклава, такие как изоляция памяти и ограничения исполнения, могут быть обеспечены, например, аппаратными средствами в компьютерном процессоре. Аттестация программного обеспечения может обеспечивать доверие в отношении безопасности изоляции конкретного анклава и в отношении кода анклава, который загружен в изолированной области памяти этого конкретного анклава. Аттестация может дополнительно обеспечивать доказательство целостности аппаратных средств и платформы программного обеспечения, на которой работает аттестованный анклав.

[0003] Системы анклава, такие как виртуальный безопасный режим (VSM) от Microsoft и расширения безопасности программного обеспечения (SGX) от Intel обеспечивают безопасность отчасти путем изолирования анклава от другого кода, работающего либо в пользовательском режиме, либо в режиме ядра. Гарантии целостности и конфиденциальности могут обеспечивать анклав более высоким уровнем доверия в аутентичности кода, работающего в анклаве, и доверия в безопасном исполнении кода анклава. Гарантия целостности может быть обеспечена посредством аттестации программного обеспечения конкретного анклава. Аттестация программного обеспечения может включать в себя криптографически подписанный хеш содержимого (инструкций и данных) внутри анклава и может комбинироваться с данными о среде анклава. Когда анклав используется в комбинации с модулем безопасности аппаратных средств (HSM), таким как аппаратные средства, соответствующие стандарту модулем доверенной платформы (TPM) группы доверенных вычислений (TCG), анклав может обеспечивать дополнительный уровень гарантий безопасности и конфиденциальности.

[0004] В дополнение к безопасности, обеспеченной изоляцией доверенного локального анклава от недоверенного локального кода вне изоляции анклава, аттестация программного обеспечения анклава может обеспечить возможность удаленного доверенного вычисления (компьютинга). Аттестация удаленного анклава может обеспечивать доверие как в целостности исполнения инструкций в анклаве, так и в конфиденциальности данных, обрабатываемых анклавом. Когда аттестация удаленного анклава обеспечена аппаратными средствами от доверенного производителя, анклав может быть доверенным, даже когда анклав находится на неизвестном компьютере, которым владеет и который поддерживает недоверенная сторона. Это является частым случаем, например, когда вычислительные ресурсы арендуются в Интернет-ресурсе облачных вычислений.

Краткое описание сущности изобретения

[0005] Предложены способы и системы для абстрагирования платформы анклава. Способ может содержать прием, платформой абстракции анклава, первого запроса использовать анклав от клиента анклава. Первый запрос может соответствовать

протоколу абстракции клиента. Первый запрос может быть преобразован во второй запрос, который соответствует нативному протоколу анклава, ассоциированному с нативной платформой анклава. Второй запрос может затем быть отправлен в нативную платформу анклава. Первый запрос может представлять собой, например, запрос реализовать (создать экземпляр) анклав, запрос верифицировать отчет аттестации анклава, запрос вызова в анклав или запрос распределить память, которая совместно используется как анклавом, так и клиентом анклава. Нативная платформа может соответствовать архитектуре анклава расширений безопасности программного обеспечения (SGX) от Intel, и нативная платформа может соответствовать архитектуре анклава виртуального безопасного режима (VSM) от Microsoft.

Краткое описание чертежей

[0006] Фиг. 1 изображает пример высокоуровневой блок-схемы системы анклава.

[0007] Фиг. 2 изображает примерный процесс для пересылки сообщений с гарантией конфиденциальности.

[0008] Фиг. 3 изображает примерный процесс для пересылки сообщений с гарантией целостности.

[0009] Фиг. 4 изображает примерный процесс для пересылки сообщений с гарантией свежести.

[0010] Фиг. 5 изображает примерный процесс для аттестации программного обеспечения анклава.

[0011] Фиг. 6 изображает примерный протокол обмена ключами Диффи-Хеллмана (DKE).

[0012] Фиг. 7 изображает примерную цепочку доверия для аттестации программного обеспечения.

[0013] Фиг. 8 представляет собой блок-схему интерфейсов компонентов программного обеспечения для примерной локальной системы анклава.

[0014] Фиг. 9 представляет собой блок-схему интерфейсов компонентов программного обеспечения для примерной локальной системы анклава с уровнем абстракции.

[0015] Фиг. 10 представляет собой блок-схему интерфейсов компонентов программного обеспечения, например, удаленной системы анклава с уровнем абстракции.

[0016] Фиг. 11 изображает примерную универсальную вычислительную среду.

[0017] Фиг. 12 изображает примерную блок-схему последовательности операций для способа абстрагирования нативной платформы анклава.

[0018] Фиг. 13 изображает примерную блок-схему последовательности операций для способа абстрагирования нативной платформы анклава.

[0019] Фиг. 14 изображает примерную блок-схему последовательности операций для способа выполнения операции анклава с абстрактной идентификационной информацией анклава.

[0020] Фиг. 15 изображает примерную блок-схему последовательности операций для способа 1500 выполнения операции анклава с абстрактной идентификационной информацией анклава.

[0021] Фиг. 16 изображает примерную систему с эквивалентностью абстрактной идентификационной информации анклава.

[0022] Фиг. 17 изображает примерную блок-схему последовательности операций для параллельной обработки с двумя эквивалентными анклавами.

[0023] Фиг. 18 изображает примерную блок-схему последовательности операций для последовательной обработки с двумя эквивалентными анклавами.

[0024] Фиг. 19 представляет собой блок-схему примерной системы распределенного запечатывания данных.

[0025] Фиг. 20 представляет собой примерную блок-схему последовательности операций для распределенного запечатывания и распечатывания данных.

[0026] Фиг. 21 представляет собой блок-схему примерного анклава хранилища ключей.

[0027] Фиг. 22 представляет собой примерную блок-схему последовательности операций для нескольких операций анклава хранилища ключей.

[0028] Фиг. 23 представляет собой примерную блок-схему последовательности операций для операции анклава хранилища ключей с запертым в хранилище ключом.

Подробное описание иллюстративных вариантов осуществления

[0029] Раскрыта модель абстракции для анклавов, которая упрощает разработку клиентов анклава и программного обеспечения, которое работает внутри анклава. Модель абстракции может представлять собой упрощение и унификацию архитектур нативных платформ анклава, таких как SGX от Intel и VSM от Microsoft. Компонент программного обеспечения уровня абстракции может переводить коммуникацию между клиентом анклава и одной или несколькими нативными платформами, между программным обеспечением внутри анклава и одной или несколькими нативными платформами анклава и между программным обеспечением внутри анклава и клиентом анклава. Такая платформа абстракции может обеспечивать выгоду, заключающуюся в обеспечения одной версии программного обеспечения анклава и программного обеспечения клиента анклава возможности работать поверх нескольких нативных платформ анклава, таких как SGX и VSM. В дополнение к упрощению задачи написания программного обеспечения для анклавов и клиентов анклава, это позволяет конечным пользователям анклавов запускать программное обеспечение анклава и клиента анклава на компьютере, который поддерживает любую поддерживаемую нативную архитектуру анклава без необходимости находить версии программного обеспечения как анклава, так и клиента анклава, которые приспособлены к нативной платформе анклава конкретного компьютера.

[0030] Модель абстракции анклава может, например, включать в себя примитивы для: администрирования срока жизни анклава, локальной и удаленной аттестации анклава, запечатывания данных в анклав, управления переносом программы в и из анклава и других характеристик безопасности, таких как монотонные счетчики и доверенное время. Также предложена абстрактная или многоуровневая идентификация анклава, которая абстрагирует идентификацию анклава за пределами одного двоичного кода или одного хеша содержимого анклава. Интерфейсы компонентов программного обеспечения, такие как интерфейс прикладного программирования (API) или двоичный интерфейс прикладного программирования (ABI), представлены для разработки анклавов и программ клиента анклава с использованием примитивов модели абстракции.

[0031] Абстрактная идентификация может включать в себя вложенную идентификацию или иерархию идентификации, которые могут использоваться, чтобы безопасно идентифицировать группы экземпляров анклава. Экземпляр анклава в настоящем документе может относиться к одному и тому же двоичному коду анклава, загруженному в анклав на одной и той же машине, и иметь одну и ту же идентификационную информацию. С другой стороны, новая версия двоичного кода или тот же самый двоичный код, загруженный на другую машину, может рассматриваться как отличающийся экземпляр. Эти разные экземпляры могут также иметь одну и ту же идентификационную информацию на более высоком уровне в иерархии идентификации.

Абстрагированная идентификация анклава обеспечивает возможность группам связанных двоичных кодов анклава идентифицироваться в качестве связанных. Например, разным версиям одного и того же анклава, таким как версии двоичных кодов анклава до и после того, как неполадка исправлена, может быть дано то же самое название, независимо от версии. На более высоком уровне абстракции, всем анклавам в семействе анклавов может даваться одно имя или идентификатор семейства. В этом случае, все анклавы, которые выполняют связанные, но разные функции, могут быть идентифицированы вместе. Другие уровни идентификации или группировки идентификаторов описаны ниже.

[0032] Любой уровень абстрактной идентификации может использоваться для различных криптографических операций, таких как запечатывание данных, аттестация анклава или гарантирование свежести данных (посредством монотонных счетчиков). Например, путем запечатывания данных, созданных одним экземпляром анклава, для идентификационной информации более высокого уровня, эти данные могут затем позже безопасно потребляться другим экземпляром анклава с той же самой идентификационной информацией анклава более высокого уровня. Путем запечатывания данных, например, для семейства анклавов, любой экземпляр анклава, который является членом этого семейства, и только члены этого семейства, будут способны распечатать (вскрыть) данные. Аттестация идентификационной информации семейства из экземпляра анклава гарантирует, что экземпляр анклава является членом этого семейства. Монотонные счетчики, привязанные к абстракции идентификации, могут обеспечивать гарантии свежести, связанные со всеми экземплярами анклава, которые являются членами идентификации абстракции.

[0033] Раскрытая модель абстракции включает в себя интерфейсы компонентов программного обеспечения, такие как интерфейс прикладного программирования (API) или двоичный интерфейс прикладного программирования (ABI), которые могут обеспечивать разработку программного обеспечения анклавов и хостов анклавов. API представляет собой набор определений, протоколов и инструментов подпрограммы программирования для создания программного обеспечения. API может определять вводы и выводы компонента программного обеспечения, типы данных, используемые компонентом программного обеспечения, и функциональность или операцию компонента программного обеспечения, независимых от любой конкретной реализации компонента программного обеспечения. API может определяться на компьютерном языке высокого уровня, таком как C, C++, C# и т.п. Формализованное определение API может облегчить взаимодействие между компонентами программного обеспечения, например, двумя компонентами программного обеспечения, написанными в разное время или разными авторами. API может быть формализован частично при помощи языка описания интерфейса (IDL), такого как Язык описания интерфейса Microsoft (MIDL) или IDL Группы управления объектами (OMG). ABI также представляет собой интерфейс между компонентами программного обеспечения, но представляет собой интерфейс объектного кода. Например, ABI может представлять собой точки входа (или адреса инструкций) объектного кода, являющегося результатом компилирования реализации исходного кода API совместно с протоколами для использования этих точек входа, такими как протоколы, специфицирующие регистры машины, которые хранят аргументы, когда вызываются точки входа.

[0034] В дополнение к обеспечению возможности взаимодействия с разными уровнями идентификации анклава, как описано выше, API анклава может абстрагировать отличия между архитектурами платформы анклава, например, между архитектурами для

безопасного изолированного исполнения, обеспечиваемого Расширениями защиты программного обеспечения (SGX) от Intel, Виртуальным безопасным режимом (VSM) от Microsoft и ARM TrustZone, Безопасной зашифрованной виртуализацией (SEV) от AMD, и архитектурами на основе программируемых вентильных матриц (FPGA). API включают в себя интерфейсы для платформы анклава, которая абстрагирует некоторые подробности абстрагированных архитектур анклава. Эти интерфейсы платформы анклава включают в себя API хоста анклава, API платформы анклава и API удаленной аттестации. API хоста анклава может использоваться недоверенным процессом хостирования, чтобы администрировать срок жизни анклава и обеспечивать связь на/от анклава. API платформы анклава может быть обеспечен доверенной платформой анклава к анклаву и может включать в себя примитивы безопасности для аттестации, запечатывания и связи с недоверенным кодом, исполняющимся на компьютере, хостирующем компьютер анклава, а также поддержки для времени выполнения ядра, такой как администрирование памяти и планирование треда. API удаленной аттестации может использоваться, чтобы выполнять удаленную аттестацию, где анклав и его клиент не хостируются (размещаются) на том же компьютере. Например, API удаленной аттестации может использоваться локальным клиентом, чтобы верифицировать, что данные исходили (или были отправлены) из анклава с определенной идентификационной информацией, работающего при изоляции, обеспечиваемой платформой анклава на удаленном компьютере. Более обобщенно, API удаленной аттестации может использоваться, чтобы устанавливать безопасные каналы связи между локальным клиентом и удаленным анклавом.

[0035] Анклавы обычно обеспечивают решения проблем, которые являются специфическими и обусловленными областью компьютерной технологии. Конкретно, анклавы обеспечивают механизм для сегрегации доверенного кода от недоверенного кода, где сегменты доверенного и недоверенного кода находятся в адресном пространстве одного компьютерного процессора. Например, анклавы обеспечивают решение безопасности для проблемы потенциально недоверенного кода (такого как код, потенциально содержащий неполадки или вирусы), исполняющегося на том же универсальном компьютере, что и код, который должен осуществлять доступ к чувствительным или закрытым данным. Варианты осуществления настоящего раскрытия обеспечивают дополнительно улучшенные решения таких проблем безопасности, возникающих в области компьютерной технологии, включая: упрощение разработки программного обеспечения путем обеспечения возможности одному анклаву или клиенту анклава быть подготовленными для нескольких нативных платформ анклава; упрощение корпоративного администрирования компьютера путем сокращения числа компонентов программного обеспечения, которые должны быть настроены на конкретные свойства аппаратных средств конкретного компьютера; и обеспечение новых безопасных вычислительных сценариев с распределенным запечатыванием данных, таких как распределение безопасной обработки анклава по анклавам, хостируемым на нескольких компьютерах.

[0036] Фиг. 1 изображает высокоуровневую блок-схему системы анклава совместно с некоторыми доверенными отношениями. Система 100 анклава включает в себя анклав 176 (альтернативно называемый контейнером анклава или безопасной средой исполнения), который включает в себя безопасную изолированную область памяти, которая содержит код 180 и данные 182. Код 180 может быть открытым или закрытым, и данные 182 могут быть открытыми или закрытыми. Например, закрытые данные или код могут принадлежать владельцу 142 данных (или быть конфиденциальными для

него), в то время как открытые данные или код могут быть обеспечены другой стороной, такой как провайдер 148 программного обеспечения. В одном варианте осуществления, код, который исполняется в контейнере 176 анклава, может быть полностью открытым, а не закрытым, в то время как данные, которые открытый код анклава использует в качестве ввода или создает как вывод, могут быть закрытыми. В другом варианте осуществления, возможно обратное, где код является закрытым, в то время как данные являются открытыми. В еще одном варианте осуществления, как код, так и данные ввода могут быть открытыми, в то время как вывод исполняющегося кода с данными ввода может быть закрытым. Возможны другие открытые и закрытые комбинации кода, данных ввода и данных вывода.

[0037] Контейнер анклава 176 хостируется на доверенных аппаратных средствах 172, которые могут одновременно хостировать недоверенное программное обеспечение 174. Первичная цель системы 100 анклава может включать в себя по меньшей мере один аспект, выбранный из списка, состоящего из: поддержания целостности кода 180, поддержания конфиденциальности кода 180, поддержания целостности данных 182 и поддержания конфиденциальности данных 182. Целью может быть защита содержимого анклава 176 от недоверенного программного обеспечения 174 (например, раскрытия недоверенному программному обеспечению, модификации посредством недоверенного программного обеспечения или т.п.). Доверенные аппаратные средства создаются производителем 162 и находятся во владении и администрировании владельца 152 инфраструктуры.

[0038] Клиент 104 анклава может представлять собой процесс или программу вне контейнера анклава, для которых анклав 176 выполняет вычисления при помощи кода 180 и данных 182. В варианте осуществления локального анклава, клиент 104 анклава может также работать на доверенных аппаратных средствах 172. В варианте осуществления удаленного анклава, клиент анклава может работать на одном компьютере, в то время как доверенные аппаратные средства 172 представляют собой другой, удаленный компьютер, соединенный с компьютером клиента анклава посредством сети. В случае локального анклава, процесс клиента анклава может также представлять собой процесс хоста анклава контейнера 176 анклава, при этом процесс клиента анклава может администрировать создание локального анклава 176. В случае удаленного анклава, анклав 176 может, например, работать на облачном компьютере Интернета, где владелец 152 инфраструктуры представляет собой провайдера услуги облачных вычислений, и облачный компьютер включает в себя доверенные аппаратные средства 172, которые произведены производителем 162.

[0039] Клиент 104 анклава может включать в себя способ конфигурирования 106, чтобы конфигурировать запрошенное вычисление при помощи анклава 176. Способ конфигурирования 106 может включать в себя предписания создания безопасного контейнера анклава 176, предписание создания экземпляра анклава (например, путем отправки запроса на недоверенное программное обеспечение 174, чтобы запросить создание экземпляра анклава), что может включать в себя копирование двоичного кода в безопасный контейнер, и предписание или запрашивание вычисления в анклаве, например, путем вызова способа в коде, скопированном в безопасный контейнер. Запрошенное вычисление может включать в себя исполнение кода 180, и данные 182 могут вводиться в запрошенное вычисление или могут быть его результатом. Данные, введенные в запрошенное вычисление, могут быть зашифрованы при нахождении вне анклава, и зашифрованные данные ввода могут быть дешифрованы перед использованием внутри анклава. Когда анклав 176 завершил запрошенную задачу,

данные, представляющие результат задачи, зашифровываются и отправляются обратно на клиент 104 анклава. Когда клиент 104 анклава принимает зашифрованные результаты, способ верификации 108 может подтвердить целостность и свежесть принятых результатов. Один провайдер 148 программного обеспечения может обеспечить код 180 для работы внутри анклава 176 и по меньшей мере часть способа верификации 108, который выполняется как часть клиента 104 анклава.

[0040] Доверие владельца данных в отношении конфиденциальности приватной части данных 182 и приватной части кода 180, а также доверие в отношении корректности результатов, произведенных анклавом 176, может быть основано на доверенных отношениях. Например, владелец 142 данных может доверять клиенту 104 анклава, который может не работать в пределах самого контейнера анклава. Владелец данных может дополнительно доверять провайдеру 148 программного обеспечения самого анклава. И владелец данных может доверять производителю доверенных аппаратных средств 172. Доверенные аппаратные средства 172 могут принимать множество форм в зависимости от используемой архитектуры анклава и могут включать в себя модуль безопасности аппаратных средств (HSM), где HSM соответствует, например, стандарту модуля доверенной платформы (TPM). Доверенные аппаратные средства 172 могут включать в себя, например, TPM и могут содержать только аппаратные средства. Например, реализация доверенных аппаратных средств 172 с использованием архитектуры анклава VSM от Microsoft может включать в себя серийный процессор с инструкциями для инструкций виртуализации операционной системы и TPM. Архитектура анклава VSM от Microsoft может включать в себя гипервизор для администрирования гостевых разделов (виртуальных процессоров), и гипервизор может предоставлять интерфейсы гипервызова гостевым разделам, чтобы разрешать гостевым разделам взаимодействовать с гипервизором. Контейнер анклава в архитектуре VSM от Microsoft может быть реализован как особый тип гостевого раздела. Пример доверенных аппаратных средств 172 с архитектурой анклава SGX от Intel может включать в себя процессор со специальными индивидуальными для конкретного анклава инструкциями и функциональностью безопасности.

[0041] Анклав, такой как анклав 176, может обеспечивать изолированную среду исполнения, которая может защищать код или данные, такие как код 180 и данные 182, путем обеспечения области памяти с ограничениями на считывание, запись или исполнение из этой защищенной области. Эта защищенная область памяти представляет собой безопасный контейнер для конфиденциального кода и данных. Ограничения на исполнение из защищенной области памяти анклава могут включать в себя ограничения на переносы исполнения, такие как инструкции вызова или перехода, между кодом вне анклава и кодом внутри анклава, и наоборот. Разные ограничения могут вводиться в действие между вызовом в анклав извне анклава и вызовом вне анклава изнутри анклава. Задействование этих переносов исполнения между областями внутри и вне анклава может быть обеспечено аппаратными средствами, например при помощи стандартной технологии виртуализации аппаратных средств или при помощи специальных аппаратных средств, таких как платформа SGX от INTEL.

[0042] Фиг. 2 изображает примерный процесс 200 для пересылки сообщений с гарантией конфиденциальности. Гарантия конфиденциальности обеспечивает некоторый уровень гарантии того, что связь между двумя сторонами, такими как Анна 210 и Бен 230 в этом примере, остается скрытой от третьих сторон, когда сообщения пересылаются через общедоступную или незащищенную среду связи, такую как сеть 218. В этом примере, Анна 212 желает отправить конфиденциальное сообщение Бену 230. Блок 212

сообщения, содержащий конфиденциальные данные, зашифровывается с использованием общедоступного ключа 216 посредством операции шифрования 214. Операция шифрования 214 может представлять собой, например, режим Галуа/счетчика стандарта расширенного шифрования (AES-CGM), но может также представлять собой любую операцию шифрования, известную специалистам в области цифровой криптографии. Зашифрованный блок 220 может пройти через сеть 218 к Бену, где зашифрованный блок 234 дешифруется при помощи конфиденциального ключа 236, чтобы сформировать незашифрованный блок 232 сообщения для Бена. При помощи тщательного выбора ключей и алгоритмов шифрования, как известно в области компьютерного шифрования данных, сообщение может оставаться конфиденциальным даже при прохождении через общедоступную сеть.

[0043] Фиг. 3 изображает примерный процесс 300 для пересылки сообщений с гарантией целостности. Гарантия целостности обеспечивает некоторый уровень гарантии того, что коммуникация между двумя участниками, такими как Анна 310 и Бен 350 этом примере, не подвергается вмешательству или иным изменениям, когда сообщения пересылаются через общедоступную или незащищенную среду связи, такую как сеть 340. В примере на фиг. 3, Анна 310 отправляет сообщение 314 Бену 350, так что существует уверенность в том, что сообщение 314 не подвергается вмешательству или иным искажениям, когда Бен 350 принимает его. Чтобы обеспечить эту гарантию целостности, процесс безопасного хеширования 316 работает в отношении сообщения 314, чтобы сформировать значение 318 хеша. Значение 318 хеша затем подписывается посредством процесса подписания 320 с использованием конфиденциального ключа Анны, чтобы сформировать подпись 342. Подпись 342 может быть отправлена по общедоступной сети 340 или посредством другого процесса незащищенной связи совместно с копией сообщения 314 в качестве сообщения 344. Бен затем принимает сообщение 354, для которого Бен желает верифицировать целостность, так что Бен может иметь уверенность в том, что сообщение 354 является тем же самым, что и сообщение 314, которое отправила Анна, после пересылки через недоверенную сеть 340. Чтобы верифицировать целостность, принятое сообщение 354 обрабатывается безопасным хешированием 356, которое идентично безопасному хешированию 316, чтобы сформировать значение 358 хеша. Принятая подпись 342 верифицируется посредством процесса верификации 360 подписи с использованием общедоступного ключа Анны. Значение 318 хеша, которое извлекается из подписи 342, затем сравнивается со значением 358 хеша, чтобы верифицировать 380, что подписи являются одними и теми же. Если они являются одними и теми же, сообщение принимается 384, как имеющее целостность. С другой стороны, если сообщение 314 было изменено каким-либо образом до приема в качестве сообщения 354, то подпись не будет корректной, и сообщение будет отклонено 388.

[0044] В некоторых вариантах осуществления, безопасное хеширование 316 и безопасное хеширование 356 могут представлять собой криптографическую хеш-функцию. Криптографическая хеш-функция представляет собой одностороннюю функцию, которая отображает данные произвольного размера на битовую строку (обычно намного меньшего) фиксированного размера. Вывод хеш-функции может называться значением хеша или просто хешем. Хорошая хеш-функция будет односторонней в том, что будет сложно определить ввод произвольного размера при наличии только вывода хеша. При хорошей хеш-функции, даже малое изменение во вводе будет создавать изменение в выводе.

[0045] Система связи может комбинировать гарантии конфиденциальности и

целостности. Например, шифрование блока сообщения согласно фиг. 2 может комбинироваться с подписанием хеша сообщения согласно фиг. 3. Система комбинации может потребовать двух наборов пар общедоступного/конфиденциального ключей, один для отправителя и один для получателя. Система комбинации может использовать один конфиденциальный ключ у получателя, чтобы дешифровать сообщение (конфиденциальный ключ Бена, как на фиг. 2), в то же время используя другой конфиденциальный ключ, чтобы подписать хеш сообщения (конфиденциальный ключ Анны, как на фиг. 3).

[0046] Фиг. 4 изображает примерный процесс 400 для пересылки сообщений с гарантией свежести. Гарантия свежести обеспечивает некоторый уровень уверенности в том, что когда несколько сообщений отправлены от одной стороны к другой, например, от Анны 410 к Бену 450 в этом примере, сообщение, принятое в получателе, представляет самое последнее сообщение. Гарантия свежести может строиться на гарантии целостности и может предотвращать атаку повторением перехваченных данных. При обеспечении гарантии целостности, для третьей стороны с мошенническим намерением будет трудно создать свое собственное сообщение и отправить его Бену так, чтобы Бен понимал это сообщение как созданное Анной. Однако третья сторона может взять сообщение, действительно созданное Анной, возможно, перехваченное в момент, когда оно отправлялось по общедоступной сети, и третья сторона может повторно отправить его Бену в другое более позднее время (т.е. повторить перехваченное сообщение). Бен определит, что сообщение было действительно создано Анной (поскольку оно было), но Бен не будет знать, что Анна не является тем лицом, которое отправило его в этот раз. Это может определяться как атака путем повтора перехваченных данных на Бена или Анну третьей стороной. Фиг. 4 представляет собой примерное решение для предотвращения атак повтором перехваченных данных путем обеспечения гарантии свежести с использованием случайных (контрольных) слов и временных меток. Случайное слово представляет собой одноразовое случайное число, связанное с системой, для обеспечения того, что то же самое число никогда не используется в качестве случайного слова более одного раза. В некоторых системах случайное слово может представлять собой просто монотонно возрастающее число, так что каждое используемое число выше, чем все числа, которые шли до него.

[0047] На фиг. 4, сообщение 414 Анны может быть отправлено по общедоступной сети 430 как сообщение 436 вместе со случайным словом 434 и временной меткой 432. Случайное слово 434 генерируется криптографически безопасным генератором 426 псевдослучайных чисел (CSPRNG), и временная метка 432 создается синхронизированным тактовым сигналом 424. Существует много систем CSPRNG, которые известны специалистам в области цифровой криптографии. Синхронизированный тактовый сигнал (часы) 424 на стороне Анны сети 430 синхронизирован с синхронизированным тактовым сигналом (часами) 480 на стороне Бена сети. На стороне Бена, когда сообщение 454 принято, прилагаемое случайное слово 434 сохраняется в кэше последних случайных слов 470. Свежесть этого принятого сообщения 450 может быть верифицирована при помощи двух проверок. Сначала, случайное слово 434 проверяется в блоке 460 путем сравнения случайного слова 434 с кэшем последних случайных слов 470, чтобы определить, наблюдалось ли раньше принятое в настоящее время случайное слово 434. Если принятое случайное слово 434 наблюдалось раньше, сообщение 454 отклоняется как сообщение повтора перехваченных данных в блоке 468. Если принятое случайное слово 434 не наблюдалось раньше, сообщение определяется как OK в блоке 464 для этой первой проверки. Во-вторых,

принятая временная метка 432 сравнивается с локальным синхронизированным тактовым сигналом 490. Если временная метка является недавней, сообщение 454 определяется как приемлемое в блоке 494, в ином случае сообщение 454 отклоняется как истекшее в блоке 498. Величина задержки, допустимой при определении того, является ли недавняя временная метка недавней, в блоке 490, может зависеть от ожидаемого рассогласования тактовых сигналов между синхронизированным тактовым сигналом 424 и синхронизированным тактовым сигналом 480 и временных задержек при обработке и передаче сообщения через сеть 430.

[0048] Система связи может комбинировать гарантию свежести с одной или обеими из гарантии конфиденциальности, как на фиг. 2, и гарантии целостности, как на фиг. 3. В системе, комбинирующей все три, сообщение 436, отправляемое по сети 430, будет представлять собой зашифрованную версию исходного сообщения 414 Анны, и подпись, содержащая подписанный хеш сообщения 414, будет включена совместно с временной меткой 432, случайным словом 434 и сообщением 436.

[0049] Фиг. 5 изображает примерный процесс 500 для аттестации программного обеспечения анклава. Аттестация программного обеспечения, при комбинировании с протоколом согласования ключей, таким как протокол согласно фиг. 6, может гарантировать верификатору, что он установил совместно используемый секрет с конкретным элементом программного обеспечения, которое хостируется внутри изолированного контейнера, созданного доверенными аппаратными средствами. В варианте осуществления согласно фиг. 5, клиент 510 анклава (верификатор аттестации) может желать использовать услуги безопасного вычисления анклава на доверенной платформе 530. Доверенная платформа 530 хостируется на компьютере (не изображен), так что доверенная платформа 530 может содержать поднабор хостирующего компьютера. Доверенная платформа 530 может содержать элементы аппаратных средств и элементы программного обеспечения хостирующего компьютера. Анклав содержит безопасный контейнер 536 анклава, и код и данные внутри него, такие как открытый код и данные 538 и секретный код и данные 542.

[0050] Три процесса комбинируются в примерном процессе 500: процесс обмена ключами, который создает совместно используемый ключ SK; процесс аттестации для аттестации клиента 510 анклава для анклава на доверенной платформе 530; и безопасное вычисление. Совместно используемый ключ SK из первого процесса используется для коммуникации вводов и выводов безопасного вычисления. При обмене ключами, клиент анклава вычисляет g^A, хранящийся в блоке 512, из конфиденциального ключа A клиента анклава и функции g генератора, например, как описано ниже для протокола обмена ключами Диффи-Хеллмана (DKE) на фиг. 6. Вычисленный g^A затем отправляется в сообщении 520 в доверенную платформу 530. Сообщение 520 может быть безопасно отправлено без шифрования и до завершения аттестации. Программное обеспечение внутри безопасного контейнера 536 анклава может использовать конфиденциальный ключ B анклава, чтобы вычислить g^B с использованием той же самой функции g генератора. Как B, так и g^B могут храниться в контейнере анклава в блоке 540.

[0051] Чтобы аттестовать идентификатор анклава (чтобы обеспечить уверенность в том, какой код работает внутри безопасного контейнера 536 анклава), сообщение 522 аттестации отправляется на клиент 510 анклава. Сообщение аттестации может представлять собой специальное сообщение, подписанное для обеспечения целостности как на фиг. 3. Специальное сообщение может содержать информацию идентификации об анклаве. При комбинировании с DKE, как в варианте осуществления на фиг. 5,

специальное сообщение может также включать в себя параметры обмена ключами. В варианте осуществления согласно фиг. 5, начальное состояние 538 безопасного контейнера 536 анклава открытого кода и открытых данных используется в качестве идентификатора анклава, хотя возможны другие идентификаторы. Вместо отправки всего начального состояния 538 в сообщении аттестации, хеш начального состояния, M=Hash(начальное состояние), отправляется взамен. Сообщение 522 аттестации включает в себя содержимое сообщения (M и g^B) и подпись содержимого сообщения ($Sign_{AK}(Hash(g^B), M)$). Подпись содержимого сообщения может создаваться, например, программным обеспечением внутри безопасного контейнера 536 анклава, запрашивая доверенную платформу 530, хостирующую анклав, аттестовать хеш вычисленного g^B и идентификатором анклава. Доверенная платформа 530 может сделать это путем обеспечения подписи с использованием ключа аттестации платформы (AK) 532, чтобы создать $Sign_{AK}(Hash(g^B), M)$. В этом примере, идентификатор анклава представляет собой хеш M начального состояния 538, но возможны другие формулировки идентификационной информации. Эта подпись $Sign_{AK}(Hash(g^B), M)$ аттестации связывает значение g^B с идентификатором M анклава и также связывает g^B и M с доверенной платформой 530. Клиент 510 анклава может затем верифицировать сообщение аттестации путем верифицирования подписи аттестации и идентификатора анклава. Подпись может быть верифицирована как на фиг. 3 с использованием общедоступного ключа, соответствующего ключу AK аттестации. Верифицирование подписи может обеспечивать гарантию целостности идентификатора анклава в сообщении аттестации. Идентификатор анклава может быть верифицирован путем сравнения информации идентификатора в сообщении аттестации с независимо известным значением идентификатора. Например, если информация идентификатора в сообщении аттестации представляет собой хеш начального состояния анклава, клиент 510 анклава может знать хеш начального состояния или может быть способен вычислить такой хеш из известного начального состояния, и это значение может затем сравниваться со значением идентификатора, обеспеченным в сообщении аттестации.

[0052] Чтобы создать совместно используемый ключ SK, как клиент 510 анклава, так и код внутри безопасного контейнера 536 могут сгенерировать g^{AB} (функцию g генератора, примененную к произведению A на B), которая может служить как совместно используемый ключ SK. Совместно используемый ключ SK может использоваться для шифрования сообщений для конфиденциальности между клиентом 510 анклава и анклавом, например для отправки данных ввода на, и данных вывода из, код(а) внутри контейнера 536 анклава. Отметим, что совместно используемый ключ создается независимо на каждой стороне канала связи в блоках 540 и 514, ни клиент анклава, ни анклав при этом не знают конфиденциальный ключ друг друга. Например, в варианте осуществления на фиг. 5, секретный код и данные могут быть безопасно обеспечены клиентом 510 анклава путем шифрования секретного кода и данных при помощи ранее установленного совместно используемого ключа SK, создавая Enc_{SK} (секретный код/данные) перед отправкой его в сообщении 524 на доверенную платформу 530. В других вариантах осуществления, секретный код и данные 542, исполняемые в безопасном контейнере 536 анклава или используемые в нем, могут исходить из других местоположений. Безопасное вычисление может выполняться внутри безопасного контейнера 536 анклава с использованием секретного кода и/или данных 542, чтобы

создать результат 544 вычисления. Результаты 516 вычисления могут затем безопасно сообщаться обратно клиенту 510 анклава путем шифрования результатов с помощью совместно используемого ключа SK (Enc_{SK}(результаты)) перед отправкой их на клиент анклава в сообщении 526.

[0053] Процесс согласно фиг. 5, описанный выше, обеспечивает клиенту анклава гарантию того, что он осуществляет связь с "реальным" анклавом, имеющим определенный идентификатор, и что анклав защищен платформой анклава. Он не обеспечивает коду внутри контейнера анклава каких-либо гарантий об объекте на другой стороне канала связи. В альтернативном варианте осуществления (не изображен), такая гарантия о клиенте анклава может быть обеспечена посредством работы клиента анклава в качестве самого анклава. В этом альтернативном варианте осуществления, клиент анклава может запросить у доверенной платформы анклава подпись на хеше g^A, которая может затем быть верифицирована другим анклавом.

[0054] Аттестация может совершаться локально или удаленно. На фиг. 5, клиент 510 анклава может или не может находиться на том же самом компьютере, что и доверенная платформа 530, так что коммуникации между клиентом 510 анклава и доверенной платформой 530 могут происходить в пределах одного компьютера (например, путем пересылки буферов данных между разными процессами на одном и том же компьютере) или могут происходить по компьютерной сети, которая соединяет клиент 510 анклава с доверенной платформой 530. Локальная аттестация может выполняться, когда клиент 510 анклава и доверенная платформа 530 хостируются на одном и том же локальном компьютере. Артефакт, или результат, локальной аттестации называется отчетом аттестации и представляет собой $Sign_{AK}(Hash(g^A), M)$ в примере согласно фиг. 5. Удаленная аттестация может происходить, когда клиент 510 анклава и доверенная платформа 530 хостируются на разных компьютерах. Артефакт, или результат, удаленной аттестации называется котировкой (оценкой) аттестации, которая в некоторых случаях может быть очень сходной с отчетом локальной аттестации или идентичной ему. В других случаях котировка аттестации может включать в себя дополнительный артефакт доверия, связанный с компьютером или нативной платформой, обеспечивающей котировку. Такой дополнительный артефакт доверия может включать в себя сертификат работоспособности хоста, такой как TCG log, ассоциированный с TPM. Аттестация анклава может выполняться на любом уровне идентификации анклава. Анклав может иметь вложенную или иерархическую идентификацию, и аттестация на более высоких уровнях идентификации может аттестовать, что созданный экземпляр анклава является членом нарастающе большей группы потенциальных экземпляров анклава с возрастанием уровня идентификации. Более высокие уровни могут соответствовать расширенному набору потенциальных экземпляров анклава более низкого уровня. Примерная иерархия идентификаторов может включать в себя, начиная с самого низкого, наиболее конкретного уровня до более высоких, менее конкретных уровней: ExactHash, InstanceHash, ImageID, FamilyID и AuthorID.

[0055] Идентификационная информация анклава может быть получена из двоичных файлов (двоичных кодов анклава), загруженных в безопасный контейнер анклава. Двоичный код анклава может быть подписан его автором с использованием конфиденциального ключа автора. Для аттестации уровня ExactHash, начальное состояние 538, используемое для вычисления отчета аттестации (ввод в хеш-функцию для создания отчета аттестации), может включать в себя все содержимое каждого двоичного файла, загруженного в безопасный контейнер 536, за исключением двоичных

кодов, ассоциированных с доверенной платформой 530.

[0056] Аттестация на уровне идентификации InstanceHash может включать в себя поднабор начального состояния 538. Поднабор может быть специфицирован во время, когда двоичные файлы анклава (двоичные файлы), которые загружены в безопасный контейнер 536, были исходно подписаны автором этих двоичных файлов анклава. Например, первый (или первичный) двоичный файл анклава может включать в себя список идентификаторов других двоичных файлов анклава, от которых зависит первый двоичный файл анклава. Для каждого перечисленного идентификации, флаг может быть включен в первый двоичный файл, чтобы указывать, измерен или нет каждый перечисленный двоичный файл посредством хеш-функции, чтобы создать отчет аттестации InstanceHash.

[0057] Аттестация до более высоких уровней ID анклава может не включать в себя прогон всего содержимого любых двоичных кодов анклава через хеш-функцию. Вместо этого, только структура данных ID может прогоняться через хеш-функцию. Например, двоичный файл анклава может включать в себя список идентификаторов анклава более высокого уровня, таких как универсально уникальные идентификаторы (UUID), указывающие: ID образа (ImageID), уникальный для этого конкретного двоичного файла анклава; ID семейства (FamilyID), уникальный для группы двоичных файлов анклава, включающей в себя этот конкретный двоичный файл анклава, которые разработаны одним и тем же автором; и ID автора (AuthorID), уникальный для группы семейств двоичных файлов анклава, которые все разработаны одним и тем же автором. ImageID, FamilyID и AuthorID могут быть назначены автором двоичного кода анклава во время, когда двоичный код исходно подписывается. Обманные действия (спуфинг) над идентификаторами анклава можно предотвратить, при этом клиент анклава может осуществлять доступ к двоичным кодам анклава и верифицировать подпись автора на этих двоичных кодах с использованием общедоступного ключа автора (или общедоступного ключа, ассоциированного с автором). Это верифицирует целостность двоичных кодов анклава, включая любые идентификаторы более высокого уровня, назначенные автором, как созданные автором этого анклава.

[0058] Фиг. 6 изображает примерный протокол 600 обмена ключами Диффи-Хеллмана (DKE). DKE представляет собой один примерный процесс для установления совместно используемого ключа K по каналу связи, имеющему только гарантию целостности; могут использоваться другие процессы для создания совместно используемых ключей, известные в области цифровой криптографии. В примере DKE на фиг. 6, секретный ключ K совместно используется между Анной 610 и Беном 650 без какой-либо коммуникации K явно по общедоступной (незащищенной) среде связи между Анной 610 и Беном 650. До того, как процесс начинается, 1) большое простое число p и 2) порождающий элемент g в $\mathbf{Z}p$ могут быть установлены и известны как Анне, так и Бену. Затем начинается процесс, где Анна и Бен выбирают произвольное число между 1 и p. Произвольное число Анны представляет собой A, выбранное в блоке 612, и произвольное число Бена представляет собой B, выбранное в блоке 652. Анна использует свое произвольное число, чтобы вычислить $g^A \bmod p$ в блоке 614, и передает эту величину в блоке 616, которая принимается Беном в блоке 656. Бен использует свое произвольное число, чтобы вычислить $g^B \bmod p$ в блоке 654, которое передается в блоке 656 к Анне и принимается в блоке 618. Анна может создать совместно используемый ключ K как $(g^B \bmod p)^A = g^{AB} \bmod p$ в блоке 620, и Бен может создать совместно используемый ключ K как $(g^A \bmod p)^B = g^{AB} \bmod p$ в блоке 660. Чтобы предотвратить атаки через посредника,

связь между Анной и Беном по незащищенной сети от блоков 616 и 658 может включать в себя гарантию целостности сообщения, например, созданную с использованием процесса, такого как процесс на фиг. 3.

[0059] Фиг. 7 изображает примерную цепочку доверия 700 для аттестации программного обеспечения. Цепочка доверия в отношении аттестации программного обеспечения может брать начало в ключе подписи, которым владеет производитель доверенной платформы, такой как доверенная платформа 530 на фиг. 5. Доверенная платформа может включать в себя компоненты аппаратных средств, такие как безопасный процессор или аппаратный модуль безопасности (HSM), и таким образом производитель может представлять собой провайдера компьютерных аппаратных средств и может также обеспечивать программное обеспечение для доверенной платформы. Производитель может быть доверенным верификатором 702, и верификатор может знать общедоступный ключ PubRK 732 корневого ключа 736 производителя. Клиент 510 анклава на фиг. 5 представляет собой пример верификатора 702, которому может быть желательно иметь уверенность в безопасном контейнере 708. Производитель может действовать как орган сертификации и обеспечивать каждый экземпляр доверенной платформы, который создает, например, каждый безопасный процессор, уникальным ключом 722 аттестации, который используется, чтобы создать подписи аттестации. Производитель также выпускает сертификат 728 подтверждения для каждого ключа 722 аттестации. Корневой ключ 736 производителя включает в себя конфиденциальный ключ PrivRK 734, который используется, чтобы подписывать сертификат 728 подтверждения. Подписание сертификата подтверждения обеспечивает гарантию целостности, например, как показано на фиг. 3.

[0060] Сертификат 728 подтверждения включает в себя общедоступный ключ PubAK 724 ключа 722 аттестации. Сертификат 728 подтверждения может указывать, что ключ 722 аттестации следует использовать для аттестации программного обеспечения, и может сообщаться верификатору 702. Верификатор может представлять собой любой объект, желающий верифицировать аттестацию безопасного контейнера 708, например, верификатор 702 может представлять собой клиента 510 анклава на фиг. 5, которому желательно, чтобы безопасное вычисление было выполнено внутри безопасного контейнера 708. Верификатор 702 может проверить сертификат подтверждения с использованием PubRK 732, чтобы верифицировать целостность и происхождение сертификата подтверждения. Верификатор может также извлекать PubAK 724 из сертификата подтверждения. Сертификат подтверждения может быть ассоциирован с политикой сертификации, что может требовать, чтобы ключ 722 аттестации использовался только для создания подписей аттестации и чтобы конфиденциальный ключ PrivAK 726 ключа 722 аттестации содержался исключительно в хранилище, которое отделено от обычно доступной памяти компьютера доверенной платформы, например, в хранилище защищенных от несанкционированного вмешательства аппаратных средств 730. Защищенные от несанкционированного вмешательства аппаратные средства могут быть, например, аппаратными средствами, соответствующими стандарту модуля доверенной платформы (TPM).

[0061] Безопасный контейнер 708 может быть реализован на доверенной платформе 736. Реализация безопасного контейнера 708 может включать в себя определение изолированного пространства памяти для безопасного контейнера, который ограничен от доступа небезопасной обработкой. Небезопасная обработка может включать в себя, например, доступ извне доверенной платформы, но на компьютере, хостирующем доверенную платформу, или доступ изнутри других безопасных контейнеров внутри

доверенной платформы. Реализация безопасного контейнера 708 может также включать в себя загрузку открытого кода и данных в безопасный контейнер, например, начальное состояние 535 на фиг. 5.

[0062] Реализованный безопасный контейнер 708 может обмениваться ключами с верификатором 702, чтобы установить совместно используемый ключ для конфиденциальной связи. Процесс обмена ключами может представлять собой процесс обмена ключами согласно фиг. 5 или процесс DKE согласно фиг. 6. Верификатор отправляет сообщение 1 704 обмена ключами на доверенную платформу 736, например, как в блоке 616 на фиг. 6, и доверенная платформа 736 отправляет сообщение 2 706 обмена ключами обратно на верификатор 702, например, как в блоке 658 на фиг. 6.

[0063] Подпись 710 аттестации может создаваться после того, как безопасный контейнер 708 реализован и обмен ключами завершен. Реализованный безопасный контейнер 708 может быть измерен путем запуска криптографической хеш-функции на всем или части безопасного контейнера. Это может включать в себя прогон хеш-функции по содержимому изолированной памяти, и двоичным файлам, которые загружены в изолированную память, любую другую память, ассоциированную с доверенной платформой, которая используется или испытывает воздействие во время реализации безопасного контейнера, или любой их поднабор или часть. Вывод прогона этой хеш-функции представляет собой измерение 712, которое является частью подписи 710 аттестации. Криптографический хеш сообщений 704 и 706 обменов ключами может также быть включен в подпись 710 аттестации и изображен как данные 714. Измерение 712 и данные 714 могут быть подписаны с использованием конфиденциального ключа PrivAK 726 аттестации. Подпись аттестации может затем быть отправлена на верификатор 702 совместно с измерением 712 и данными 714. Верификатор может верифицировать целостность подписи аттестации с использованием PubAK 724 из сертификата подтверждения, что, в примере на фиг. 7, также позволяет осуществить верификацию целостности измерения 712 и данных 714. Верификатор 702 может верифицировать целостность безопасного контейнера 708 путем сравнения измерения 712 с ожидаемым результатом (ожидаемым результатом, определяемым, например, путем локального выполнения того же самого хеша измерения 712) и верифицировать, что подпись аттестации была создана для этого конкретного экземпляра пути коммуникации верификатора 702, путем проверки данных 714 (например, поскольку хеш данных 714 привязан к сообщению 2 706 обмена ключами). После этих операций верификации и верификации сертификата подтверждения, описанного выше, верификатор теперь имеет некоторую уверенность в том, что он может установить связь, имеющую как конфиденциальность, так и целостность, с безопасным контейнером 708 с использованием установленного совместно используемого ключа, что аппаратные средства доверенной платформы могут быть доверенными в соответствии с их производителем, и что состояние программного обеспечения доверенной платформы, используемой для создания безопасного контейнера, известно. Верификатор 702 теперь готов запросить безопасную обработку в пределах безопасного контейнера 708 с использованием частного кода и/или закрытых данных.

[0064] Платформа абстракции и примитивы анклава

[0065] Фиг. 8 представляет собой блок-схему интерфейсов компонентов программного обеспечения для примерной локальной системы анклава. Система 800 анклава включает в себя компьютер 810 с нативной платформой 812 анклава, хостирующей анклав 814, и клиент 816 анклава. Нативная платформа 812 может представлять собой компонент аппаратных средств и/или программного обеспечения на основе, например, SGX от

Intel или VSM от Microsoft. Анклав 810 может представлять собой анклав 176 согласно фиг. 1. Нативной протокол 844 для анклавов может использоваться для связи между анклавом 814, клиентом 816 и нативной платформой 812. Как изображено на фиг. 8, нативной протокол 844 включает в себя интерфейс 820 в анклаве 814, интерфейсы 822 и 824 в нативной платформе и интерфейс 826 в клиенте. Эти интерфейсы могут представлять собой API или ABI в компонентах программного обеспечения.

[0066] Использование этих интерфейсов 820, 822, 824 и 826 программного обеспечения может включать в себя перенос управления исполнения между компонентами программного обеспечения. Перенос управления может включать в себя исполнение инструкции вызова или перехода в точку входа (адрес инструкции) в компоненте программного обеспечения, на который переносится управление. Например, если нативная платформа 812 представляет собой компонент программного обеспечения, перенос управления из нативной платформы 812 на клиент 816 может происходить посредством интерфейса 826 программного обеспечения, когда инструкция вызова или перехода в нативной платформе 812 исполняется, задавая адрес в пределах клиента 816 для вызова или перехода. Задаваемый адрес внутри клиента 816 может представлять собой точку входа для функции или способ в интерфейсе 816. Перенос управления указан стрелкой на фиг. 8, например: из нативной платформы 812 на анклав 814 через интерфейс 820; из анклава 814 на нативную платформу 812 через интерфейс 822; из клиента 816 на нативную платформу 812 через интерфейс 824 и из нативной платформы 812 на клиент 816 через интерфейс 826. Нативной протокол 844 может включать в себя шаблоны связи через интерфейсы 820, 822, 824 и 826.

[0067] В некоторых вариантах осуществления, нативная платформа 812 может быть реализована, по меньшей мере частично, как компонент аппаратных средств, например, со специальными процессорными инструкциями для администрирования анклава. Такая специальная инструкция аппаратных средств может исполняться как часть компонента программного обеспечения нативной платформы 812. В альтернативных вариантах осуществления может не существовать компонента программного обеспечения для некоторых или всех из функций нативной платформы 812. В этих альтернативных вариантах осуществления, интерфейсы 822 и 824 нативной платформы могут представлять собой инструкции аппаратных средств вместо точек входа программного обеспечения, поэтому функция нативной платформы 812 может использоваться анклавом 814 или клиентом 816 или может использоваться путем исполнения специальной инструкции аппаратных средств в анклаве 814 или клиенте 816, соответственно, вместо исполнения инструкции вызова или перехода.

[0068] В некоторых вариантах осуществления, клиент 816 анклава 814 может сам представлять собой анклав. Например, клиент 816 анклава может использовать интерфейс 824, чтобы запросить создать анклав 814. В этих вариантах осуществления, связь между анклавом 814 и клиентом 816 через нативную платформу 812 в действительности представляет собой связь между двумя анклавами. Когда клиент 816 также представляет собой анклав, клиент 816 анклава может также использовать интерфейс 822 и предоставлять интерфейс, аналогичный 820 (не изображен).

[0069] Фиг. 9 представляет собой блок-схему интерфейсов компонента программного обеспечения, например, локальную систему анклава с уровнем абстракции. Система 900 анклава включает в себя платформу 912 абстракции для перевода между нативным протоколом 944 и протоколами 940, 942 абстракции. Нативная платформа 918 может быть аналогична платформе 812 абстракции на фиг. 8, и интерфейсы 928 и 930 могут комбинировать функции интерфейсов 820, 822, 824 и 825 на фиг. 8. Протокол 940

абстракции анклава включает в себя интерфейсы 920, 922 для анклава 914, в то время как протокол 942 абстракции клиента включает в себя интерфейсы 924, 926 для клиента 916. Как на фиг. 8, клиент 916, работающий на компьютере 910, может запросить создание анклава 914 посредством интерфейса 924. Уровень 912 абстракции может вызвать создание анклава 914 с использованием нативного протокола 944 и интерфейсов 928, 930 с нативной платформой 918. Клиент 916 и анклав 914 могут использовать протоколы 940 и 942 абстракции, когда нативная платформа 918 и нативной протокол 944 основаны на разных архитектурах анклава, таких как SGX от Intel или VSM от Microsoft. Как на фиг. 8, клиент 916 анклава 914 может сам представлять собой анклав, и нативная платформа 918 может включать в себя компоненты аппаратных средств и/или программного обеспечения.

[0070] Анклав 914 и клиент 916 могут не осуществлять связь напрямую и могут вместо этого осуществлять связь только посредством платформы 912 абстракции. Прямая связь может быть невозможной или нежелательной, например, из-за изоляции памяти анклава 914. Изоляция памяти анклава может предотвращать считывание из, запись в или исполнение (переход в или из) изолированной памяти анклава.

[0071] Анклав 914 может включать в себя инструкции, находящиеся внутри безопасного контейнера анклава компьютера 910. Клиент 916 может включать в себя инструкции, расположенные в адресном пространстве памяти компьютера 910, но вне безопасного контейнера анклава 914. Платформа 912 абстракции может быть реализована различными способами, включая инструкции, которые находятся внутри или вне безопасного контейнера анклава 914, и может также включать в себя инструкции, исполняемые из гипервызовов. В случае, где платформа 912 абстракции включена, по меньшей мере частично, в безопасный контейнер анклава 914, код платформы абстракции внутри безопасного контейнера может быть разработан отдельно от остатка кода анклава 914 и может взаимодействовать только с другим кодом анклава посредством общедоступных API/ABI. Такой код платформы абстракции может быть статически связан или динамически связан с остатком кода внутри безопасного контейнера анклава. Статически связанный код платформы абстракции может представлять собой объектный код, который ассоциирован с платформой абстракции и включен (статически связан), совместно с кодом, который является более специфическим для анклава 914, в двоичный образ, из которого может быть реализован анклав 914. В случае динамически связанной платформы абстракции, код анклава, который является более специфическим для анклава 914, и код, ассоциированный в более общем виде с платформой абстракции, могут исходить из отдельных двоичных образов. Что касается динамически связанного примера, см. фиг. 14.

[0072] Фиг. 10 представляет собой блок-схему интерфейсов компонента программного обеспечения для примерной удаленной системы анклава с уровнем абстракции. Удаленная система 1000 анклава включает в себя анклав 1014 на компьютере 1010 и клиент 1056 анклава 1014 на отдельном компьютере 1050. Комбинация клиентской заглушки 1016 и удаленной (дистанционной) платформы 1052 абстракции может облегчать взаимодействие между анклавом 1014 и клиентом 1056. Множество элементов в компьютере 1010 могут быть идентичны или аналогичны идентично названным элементам компьютера 910 на фиг. 9. Конкретно, платформа 1012 абстракции, протоколы 1040, 1042, 1044 и нативная платформа 1018 могут быть аналогичны или идентичны соответствующим элементам 912, 940, 942, 944 и 918, соответственно.

[0073] Клиентская заглушка 1016 может осуществлять связь с удаленной платформой 1052 абстракции посредством сетевой связи 1080. Удаленный протокол 1082 клиента

и интерфейсы 1064, 1066 могут быть аналогичны протоколу 1042 абстракции клиента и интерфейсам 1024, 1026. Однако удаленный протокол клиента может включать в себя дополнительную функциональность для дистанционного взаимодействия. Например, способ в интерфейсе 1064, такой как CreateEnclave, чтобы запросить создание анклава, может дополнительно включать в себя способность специфицировать компьютер хоста анклава, такой как компьютер 1010, где запрашивается создание анклава. Котировка аттестации анклава 1014, предоставляемая клиенту 1056 посредством удаленного протокола клиента, может предоставляться вместо отчета аттестации или в дополнение к нему. Компьютер 1050 с клиентом 1056 может или не может включать в себя нативную платформу 1058 анклава. Если представлена нативная платформа 1058, она может или не может соответствовать примерной нативной платформе 1018 архитектуры анклава, и тем самым нативной протокол 1044 и удаленный нативной протокол 1084 могут не быть одними и теми же.

[0074] В альтернативном варианте осуществления (не показан), клиентская заглушка 1016 может не существовать, и платформа 1012 абстракции может напрямую осуществлять связь с удаленной платформой 1052 абстракции по сети.

[0075] Протоколы абстракции анклава, такие как 940, 942, 1040, 1042, 1082 на фиг. 9 и 10, могут включать в себя множество методов интерфейса или точек входа, чтобы администрировать и использовать анклавы, которые построены на нескольких нативных платформах, таких как SGX от Intel и VSM от Microsoft. Эти методы могут обеспечивать примитивы анклава, которые могут быть реализованы на нескольких нативных платформах, тем самым обеспечивая "абстракцию" нативных платформ. Примитивы анклава, раскрытые здесь, включают в себя администрирование срока жизни анклава, аттестацию, запечатывание данных, перенос управления, монотонные счетчики и доверенное время.

[0076] Примитивы для администрирования срока жизни анклава могут включать в себя способы для побуждения реализации или завершения анклава, такого как анклав 914. Примитивы администрирования срока жизни могут быть частью протокола 942 абстракции клиента, и, более конкретно, могут быть реализованы платформой 912 абстракции в качестве части интерфейса 924 для использования клиентом 916.

[0077] Способ для реализации или создания анклава может включать в себя задание исполняемого образа кода и/или данных, подлежащих загрузке в изолированную память безопасного контейнера анклава. Этот код, до или после того, как он загружен в контейнер анклава, может стать частью начального состояния, используемого для аттестации реализованного анклава (как объяснено выше в отношении фиг. 5). Например, исполняемый образ анклава (двоичный код анклава) может быть задан клиентом анклава путем обеспечения указателя на буфер в памяти, содержащей исполняемый образ. Альтернативно, образ анклава может быть задан путем указания файла в файловой системе, содержащей двоичный код анклава. В некоторых вариантах осуществления, заданный образ анклава может быть зашифрован; в других вариантах осуществления, анклав может быть не зашифрован; в других вариантах осуществления, анклав может быть частично зашифрован. Измерение двоичного кода анклава для аттестации может происходить по зашифрованному исполняемому образу или после дешифрования.

[0078] Код и/или данные, подлежащие загрузке первоначально в анклав, могут быть указаны заданием файла, содержащего первичный образ анклава. В дополнение к этому коду и/или данным, первичный образ анклава может включать в себя дополнительные метаданные, такие как желательный размер анклава (объем памяти, требуемый внутри

контейнера анклава), местоположения точек входа внутри кода в файле и список файлов зависимых образов. Файлы зависимых образов представляют собой файлы других (не-первичных) образов, которые могут также загружаться в анклав совместно с кодом и данными в файле первичного образа. Файлы зависимых образов могут сами содержать списки дополнительных файлов зависимых образов. В случае локальной системы анклава согласно фиг. 9, первичный и зависимые образы могут наполнять любое доступное устройство хранения данных, например, посредством локально доступной файловой системы. В случае системы удаленного анклава согласно фиг. 10, файл первичного образа может находиться в любом устройстве хранения данных, доступном компьютеру 1010 или компьютеру 1050. Если клиент 1056 запрашивает создание анклава на компьютере 1010 с использованием первичного образа, расположенного на компьютере 1050, удаленная платформа абстракции и клиентская заглушка 1016 могут действовать согласованно, чтобы скопировать заданный файл первичного образа на компьютер 1010.

[0079] CreateEnclave представляет собой примерный метод для создания экземпляра анклава. Метод CreateEnclave может быть описан при помощи псевдокода:

```
HANDLE
CreateEnclave(
  _In_  PCWSTR    enclavePath,
  _In_  DWORD     flEnclaveType,
  _In_  DWORD     dwFlags,
  _In_reads_bytes_(dwInfoLength)
        PCVOID    enclaveInformation,
  _In_  DWORD     dwInfoLength,
  _Out_opt_ PDWORD    enclaveError
  )
```

[0080] Псевдокод, используемый для описания методов в настоящем документе, может использовать несколько соглашений псевдокода для определения интерфейсов API. Например, параметры функции, такие как enclavePath выше, могут быть оформлены с "_In_" или "_Out_", чтобы указывать, что параметр представляет собой параметр ввода или вывода, соответственно. "_Out_opt_" может указывать опциональный параметр вывода. Написанные заглавными буквами слова могут указывать тип данных. HANDLE может представлять собой номер, такой как 32-битный номер 32, используемый, чтобы косвенным образом ссылаться на что-то. Например, вышеуказанный способ CreateEnclave возвращает HANDLE вызывающему CreateEnclave, и этот HANDLE может представлять собой дескриптор анклава, который был создан; PCWSTR может представлять собой указатель на определенный тип текстовой строки; DWORD может представлять собой неподписанную 32-битную величину; PCVOID может представлять собой указатель на данные не-специфицированного типа; BOOL может представлять собой двоичное значение.

[0081] CreateEnclave может разрешать клиенту, такому как клиент 916, создавать анклав и загружать первичный образ в анклаве. Любая информация конфигурации анклава в этом образе может быть ассоциирована с созданным экземпляром анклава. CreateEnclave может включать в себя следующие параметры:

lpEnclaveName: может специфицировать путь к первичному образу анклава, который в реализациях может представлять собой некоторый другой тип идентификатора для идентифицирования кода и/или данных первичного образа анклава, такой как дескриптор для открытого файла, унифицированный идентификатор ресурса (URI) или идентификатор, который используется при внешнем поиске. Например, глобально уникальный идентификатор (GUID) может использоваться в качестве ключа в базу данных первичных образов. В других реализациях, этот параметр может идентифицировать область памяти, которая содержит первичный образ анклава.

flEnclaveType: может специфицировать тип анклава, подлежащего созданию (в случае, если образ анклава поддерживает несколько типов). Может быть установлен в DEFAULT в случае, если двоичный код поддерживает только один анклав, или разработчик явно специфицировал значение по умолчанию. dwFlags: может специфицировать одно или несколько предопределенных действий, которые следует предпринять при создании анклавов и загрузке первичного образа анклава.

enclaveInformation: может представлять собой дополнительные параметры ввода для конфигурации среды исполнения анклава.

lpEnclaveError: может специфицировать дополнительный параметр, чтобы возвращать код ошибки конкретной архитектуры.

[0082] После успешного завершения, CreateEnclave может возвратить дескриптор на анклав. При ошибке, может быть возвращен NULL. Другие идентификаторы (GUID, URI и т.д.) могут также возвращаться без отклонения от объема настоящего раскрытия. Для простоты, настоящая спецификация будет описывать API с использованием дескриптора. Создание анклава может быть безуспешным, например, из-за недостатка памяти анклава, недостатка поддержки специфицированного типа анклава в платформе абстракции или нативной платформе, или создание может быть безуспешным из-за явных политик конфигурации, препятствующих исполнению на системе анклава конкретного типа.

[0083] Реализации CreateEnclave и другого метода API, описанные ниже, могут исключать один или несколько описанных параметров метода. Например, касательно CreateEnclave, могут быть исключены lpEnclaveName, flEnclaveType, dwFlags и enclaveInformation, используя конкретное предопределенное значение для данного конкретного API. Аргумент lpEnclaveError может также исключаться из API, и альтернативные способы проверки на ошибки в вызове API могут быть опционально реализованы.

[0084] CreateEnclave может отвечать за загрузку всех зависимых модулей, как специфицировано в первичном образе анклава. Первичный образ анклава может представлять собой файл переносимого исполнения (PE), который специфицирует другие файлы двоичного образа, от которых зависит первичный образ. CreateEnclave может также выполнять конкретную инициализацию нативной платформы, такую как финализация измерений для аттестации, распределение структур для безопасности транспортного уровня (TLS) и/или других протоколов согласования ключей и связи и т.д. Интерфейсы 920, 922 протокола абстракции анклава (включая методы, например, для запечатывания и аттестации данных) могут применяться, когда инициализация анклава завершена.

[0085] TerminateEnclave представляет собой примерный метод для завершения анклава:

```
VOID
TerminateEnclave(
  _In_  HANDLE  hEnclave
)
```

[0086] TerminateEnclave может использоваться, чтобы разрушить анклав. В реализациях, разрушение анклава может включать в себя принуждение всех тредов анклава возвратиться на хост или завершиться, и/или освобождение памяти, ассоциированной с анклавом. Вызов TerminateEnclave на работающем анклаве может завершить его и освободить все ресурсы, ассоциированные с анклавом.

[0087] Платформа 912 абстракции анклава может включать в себя примитивы переноса управления исполнением, которые могут использоваться, например, чтобы перенести управление между анклавом и его клиентом. Примитивы переноса управления исполнения могут обеспечить возможность связи между анклавом 914 и клиентом 916 путем запуска исполнения кода в точке входа в другом компоненте. Примитивы переноса управления исполнения разрешают пересылку данных в/из анклавов, позволяя параметрам ассоциироваться с запросом переноса управления; параметры могут специфицировать индивидуальные элементы данных (сами параметры сообщаются), или параметры могут быть указателями на области памяти (сообщаются буферы, указываемые параметрами). Эти примитивы могут задействовать перенос управления, несмотря на ограничения на прямой вызов или переход между анклавом 914 и клиентом 916 из-за ограничений безопасности на контейнере анклава.

[0088] Для вызова в анклав, интерфейс 924 может включать в себя механизмы, чтобы разрешать клиенту 916 вызывать в анклав 914 через интерфейс 920. Например, интерфейс 924 может включать в себя методы GetProcAddress и CallEnclave:

```
typedef PVOID (*ENCPROC)(PVOID);
ENCPROC
GetProcAddress(
 _In_  HMODULE  hEnclave,
 _In_  LPCTSTR  lpProcName
)
BOOL
CallEnclaveIn(
 _In_  ENCPROC  pCallin,
 _In_  PVOID  pParameter,
 _Out_  PVOID  pReturn
)
```

[0089] Клиент анклава, такой как клиент 916, может вызывать в анклав, такой как анклав 914, с использованием указателя функции, возвращенного посредством GetProcAddress(). Параметр lpProcName может сопоставить функцию, экспортируемую в первичный образ анклава. Например:

```
// Call Callin function for enclave.
ENCPROC pCallin = (ENCPROC) GetProcAddress(hEnclave,
"CallinExample");
PVOID pParameter; // Pointer to memory
if (NULL != pCallin)
{
  CallEnclaveIn(pCallin, pParameter);
}
```

[0090] В других вариантах осуществления GetProcAddress, lpProcName может представлять собой другой идентификатор конкретной экспортируемой функции, такой как номер, такой как выбор из нумерации точек входа, экспортируемых из образа анклава, или другой не-текстовый идентификатор, соответствующий функции. Другие варианты осуществления CallEnclaveIn могут дополнительно брать параметр ввода, специфицирующий анклав, подлежащий вызову в анклав, например, возвращаемое дескриптором CreateEnclave.

[0091] При вызове в анклав, тред в процессе клиента может приостанавливаться, и тред анклава (с отдельным ID треда) может использоваться, чтобы обслуживать вызов в запросе. Код анклава, исполняющийся на треде анклава, может затем иметь доступ к памяти, которая была ранее доступна клиенту анклава перед вызовом в анклав. Например, клиент может поместить данные в буфер, указанный посредством pParameter, перед вызовом метода абстракции CallEnclaveIn, и затем анклав может иметь доступ к буферу, указанному посредством pParameter, при обслуживании вызова в запросе. После вызова вовне, тред исходного (клиентского) вызова может использоваться для обслуживания вызова вовне. Может поддерживаться повторный вход (например, вызов вовне в хост может снова вызывать в анклав).

[0092] Для вызова из анклава, интерфейс 922 может включать в себя способы, относящиеся к вышеописанным методам CallEnclaveIn, которые разрешают анклаву 914 вызывать вовне на клиент 916 анклава. Например, анклав 914 может вызвать вовне в любую функцию в процессе хоста конкретного типа, например типа функции ENCPROC. Указатель функции для этого может пересылаться с использованием параметров вызова в анклав.

```
BOOL
CallEnclaveOut(
 _In_  ENCPROC  pCallout,
 _In_  PVOID  pParameter,
 _Out_  PVOID  pReturn
)
// Call out to function in host process
ENCPROC pCallout = (ENCPROC) 0xF00; // address to some function in host
PVOID pParameter = // Pointer to memory
CallEnclaveOut(pCallout, pSharedMemory);
```

Интерфейс 920 может включать в себя точки входа, зарегистрированные как функция "CallinExample" выше, и интерфейс 926 может включать в себя точки входа, зарегистрированные как функции "Callout" выше. Например, в случае, где первичный образ анклава находится в формате переносимого исполняемого (PE) образа, точки входа функции в образе могут быть перечислены как точки входа "экспорта", и каждая такая экспортируемая точка входа может включать в себя текстовое имя, такое как "CallinExample," чтобы идентифицировать и отличать точки входа в этом PE образе анклава; в других реализациях точки входа функции могут быть маркированы дополнительными метаданными, такими как один бит, указывающий, что функция может представлять собой точку входа для анклава. В примере выше для вызова вовне анклава, адрес функции вызова вовне задается как 0xF00 и является только примером. Действительный адрес функции вызова вовне может определяться множеством способов. Например, адрес функции вызова вовне внутри клиента может пересылаться в анклав как параметр для функции вызова внутрь. В другом примере, адрес функции вызова вовне может быть зарегистрирован клиентом с использованием функции, такой как RegisterCallOut:

```
BOOL RegisterCallOut(
 _In_  ENCPROC  pCallout,
 _In_  LPCTSTR  lpProcName)
```

Код внутри анклава может получать адрес функции вызова вовне путем вызова комплементарной функции, такой как GetCallOut:

```
BOOL GetCallOut(
 _Out_  ENCPROC  *pCallout,
 _In_  LPCTSTR  lpProcName)
```

[0093] В других вариантах осуществления, методы CallEnclaveIn и CallEnclaveOut могут в действительности представлять собой один и тот же метод. Например, один метод CallEnclave может использоваться для вызова в анклав и вызова из анклава. В ситуациях, где клиент 916 анклава также представляет собой анклав, вызов из анклава 914 на клиент 916 будет также представлять собой вызов в анклав.

[0094] Платформа 912 абстракции может обеспечивать примитивы для запечатывания данных в анклав. Например, интерфейс 922 может предоставлять услуги на анклав 914, такие как запечатывание и распечатывание (вскрытие) данных по идентификатору. Как объяснено выше, анклав может иметь несколько вложенных идентификаторов, и данные могут быть запечатаны по любому такому идентификатору. Когда данные запечатаны по идентификатору, который соответствует набору возможных экземпляров анклава, запечатанные данные могут распечатываться любым из этого соответствующего набора экземпляров анклава. Например:

```
struct SEALING_POLICY
{
 ENCLAVE_ID_TYPE  enclaveIdType;
};
```

[0095] Для каждого значения enclaveIdType, анклав будет запечатываться по ID отображения. Возможные типы идентификатора анклава (и значения enclaveIdType) включают в себя:

```
ENCLAVE_EXACTHASH
ENCLAVE_INSTANCEHASH:    // seal using MRENCLAVE for SGX,
IMAGE HASH for VSM
ENCLAVE_IMAGEIDS:    // not supported in SGX, will use IMAGE IDS for
VSM
ENCLAVE_FAMILYID:    // will use PRODUCTID for SGX, FAMILY ID for
VSM
ENCLAVE_AUTHORID:  // will use MRSIGNER for SGX, AUTHOR ID for
VSM
```

[0096] Платформа может также применять дополнительную конфигурацию отладки (авторскую и времени исполнения) к политике запечатывания. Для разных политик отладки могут использоваться разные ключи запечатывания. Например, конфигурации отладки и выпуска могут использовать разные ключи запечатывания.

```
DWORD
EnclaveSeal(
 _In_          SEALING_POLICY sealingPolicy,
_In_reads_bytes_opt_(dwPlaintextSize) LPCVOID pPlaintext,
_In_           DWORD  dwPlaintextSize,
_In_reads_bytes_opt_(dwAuthdataSize) LPCVOID pAuthdata,
_In_           DWORD  dwAuthdataSize,
_Out_writes_bytes_to_(dwSealedtextSize) LPVOID pSealedtext,
_Inout_          DWORD  *dwSealedtextSize
)
DWORD
EnclaveUnseal(
_In_reads_bytes_opt_(dwSealedtextSize) LPCVOID pSealedtext,
_In_           DWORD  dwSealedtextSize,
_In_reads_bytes_opt_(dwAuthdataSize) LPCVOID pAuthdata,
_In_           DWORD  dwAuthdataSize,
_Out_writes_bytes_to_(dwPlaintextSize) LPCVOID pPlaintext,
_Inout_          DWORD  *dwPlaintextSize
)
```

[0097] Платформа 912 абстракции может обеспечивать примитивы для аттестации, такие как для создания отчетов и котировок аттестации и для верификации отчетов и котировок. Например:

```
DWORD
CreateReport(
  _In_reads_bytes_opt_(dwTargetInfoSize) PCVOID pTargetInfo,
  _In_              DWORD dwTargetInfoSize,
  _In_reads_bytes_opt_(dwAuthData)   PCVOID pAuthData,
  _In_              DWORD dwAuthData,
  _Out_writes_bytes_opt_(*pReportSize) PVOID pReport,
  _Inout_           PDWORD pReportSize,
  _Out_opt_         PDWORD lpEnclaveError
  )
DWORD
VerifyReport(
  _In_reads_bytes_(dwReportSize)    PCVOID pReport,
  _In_              DWORD  dwReportSize,
  _Out_opt_         LPDWORD lpEnclaveError
  )
```

[0098] VerifyReport() может использоваться анклавом для подтверждения целостности отчета и того, что отчет был сгенерирован анклавом на той же машине.

```
DWORD CreateQuote(
  _In_              GUID quoteType,
  _In_              DWORD authDataSize,
  _In_reads_bytes_opt_(authDataSize) const BYTE* authData,
  _Out_             DWORD* quoteSize,
  _Outptr_result_bytebuffer_opt_(*quoteSize) BYTE** quote
)
```

[0099] В CreateQuote, quoteType может отображаться на провайдер котировки, который может представлять собой источник доверия, чтобы сгенерировать конкретную котировку. В CreateQuote, authData может представлять собой указатель на данные, которые созданы вызывающим CreateQuote, и в формате, определенном им. Отметим, что authData не обязательно должно пониматься платформой 912 абстракции. authData может быть упаковано в результирующую котировку. Можно ожидать, что провайдеры котировки будут поддерживать это.

```
DWORD VerifyQuote(
  _In_          DWORD quoteSize,
  _In_reads_bytes_(quoteSize) const BYTE* quote,
  _Out_         DWORD* reportSize,
  _Outptr_result_bytebuffer_all_(*reportSize) BYTE** report
)
```

[0100] В дополнение к примитивам анклава, описанным выше, платформа абстракции анклава может обеспечивать: администрирование памяти (например, чтобы распределять и освобождать память, такую как память, ограниченная до анклава, или память, которая совместно используется между анклавом и его клиентом); обработку исключений (например, чтобы обрабатывать ошибки или исключения, которые возникают при исполнении кода анклава); синхронизацию треда; и криптографические функции (например, шифрование, хеш-функции и подписание).

[0101] Методы, описанные выше, могут быть реализованы на одном или нескольких вычислительных устройствах или средах, как описано ниже. Фиг. 11 изображает примерную универсальную вычислительную среду, например, которая может воплощать одно или несколько из доверенных аппаратных средств 172, доверенной платформы 736 или компьютеров 810, 910, 1010 и 1050, в которых могут быть реализованы некоторые из методов, описанных в настоящем документе. Среда 1102 вычислительной системы представляет собой только один пример подходящей вычислительной среды и не подразумевается, чтобы предлагать какое-либо ограничение на объем использования или функциональность представленного раскрытого изобретения. Вычислительная среда 1102 не должна интерпретироваться как имеющая какую-либо зависимость или требование, относящееся к любому одному или комбинации компонентов, проиллюстрированных в примерной операционной среде 1102. В некоторых вариантах осуществления, различные изображенные вычислительные элементы могут включать в себя схемы, сконфигурированные для реализации конкретных аспектов настоящего изобретения. Например, термин "схема", используемый в раскрытии, может включать в себя специализированные аппаратные компоненты, сконфигурированные, чтобы выполнять функцию(и) с помощью встроенного программного обеспечения или переключателей. В других примерных вариантах осуществления, термин "схема" может включать в себя блок обработки общего назначения, память и т.д., сконфигурированные программными инструкциями, которые воплощают логику, приводимую в действие, чтобы выполнять функцию(и). В примерных вариантах осуществления, где схема включает в себя комбинацию аппаратных средств и программного обеспечения, разработчик может написать исходный код, воплощающий логику, и исходный код может компилироваться в машиночитаемый код, который может быть обработан с помощью блока обработки общего назначения. Поскольку специалист в данной области техники может понимать, что уровень техники получил такое развитие, что существует незначительное различие между аппаратными средствами, программным обеспечением или комбинацией аппаратных средств/программного обеспечения, выбор аппаратных средств по сравнению с программным обеспечением для осуществления конкретных функций является выбором на этапе проектирования, который оставлен за разработчиком. Более конкретно, специалист в данной области техники может понимать, что программный процесс может быть трансформирован в эквивалентную структуру аппаратных средств, и структура

аппаратных средств сама может быть трансформирована в эквивалентный программный процесс. Таким образом, выбор аппаратной реализации по сравнению с реализацией программного обеспечения является выбором при проектировании, оставленным за разработчиком.

[0102] Компьютер 1102, который может включать в себя любое из мобильного устройства или смартфона, планшета, ноутбука, настольного компьютера или совокупности сетевых устройств, облачных вычислительных ресурсов и т.д., как правило, включает в себя множество считываемых компьютером носителей. Считываемые компьютером носители могут быть любыми доступными носителями, к которым можно получить доступ с помощью компьютера 1102, и включают в себя как энергозависимые, так и энергонезависимые носители, съемные и несъемные носители. Системная память 1122 включает в себя компьютерные носители хранения в форме энергозависимой и/или энергонезависимой памяти, такие как постоянная память (ROM) 1123 и память с произвольной выборкой (RAM) 1160. Базовая система ввода/вывода (BIOS) 1124, содержащая основные подпрограммы, которые способствуют переносу информации между элементами внутри компьютера 1102, например, во время запуска, обычно хранится в ROM 1123. RAM 1160 обычно содержит данные и/или программные модули, которые непосредственно доступны и/или в данный момент обрабатываются блоком обработки 1159. В качестве примера, а не ограничения, фиг. 11 иллюстрирует гипервизор 1130, операционную систему 1125, прикладные программы 1126, другие программные модули 1127, включая клиент 1165 анклава и анклав 1128.

[0103] Компьютер 1102 может также включать в себя другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители хранения. В качестве примера, фиг. 11 иллюстрирует накопитель 1138 на жестком диске, который считывает или записывает на несъемные, энергонезависимые магнитные носители, накопитель 1139 на магнитном диске, который считывает или записывает на съемный, энергонезависимый магнитный диск 1154, и накопитель 1104 на оптическом диске, который считывает или записывает на съемный, энергонезависимый оптический диск 1153, такой как CD-ROM или другие оптические носители. Другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители хранения, которые могут быть использованы в примерной операционной среде, включают в себя, но без ограничения указанным, кассеты с магнитной лентой, карты флэш-памяти, цифровые универсальные диски, цифровую видеоленту, твердотельную ROM, твердотельную RAM и тому подобное. Накопитель 1138 на жестком диске обычно соединен с системной шиной 1121 через интерфейс несъемной памяти, такой как интерфейс 1134, и накопитель 1139 на магнитном диске и накопитель 1104 на оптическом диске обычно соединены с системной шиной 1121 посредством интерфейса съемной памяти, такого как интерфейс 1135 или 1136.

[0104] Накопители и ассоциированные с ними компьютерные носители хранения, описанные выше и показанные на фиг. 11, обеспечивают хранение считываемых компьютером инструкций, структур данных, программных модулей и других данных для компьютера 1102. На фиг. 11, например, накопитель 1138 на жестком диске проиллюстрирован как хранящий операционную систему 1158, прикладные программы 1157, другие программные модули 1156, такие как приложения клиента анклава и двоичные файлы анклава, и программные данные 1155. Отметим, что эти компоненты могут либо быть такими же, либо отличаться от операционной системы 1125, прикладных программ 1126, других программных модулей 1127 и программных данных 1128. Операционная система 1158, прикладные программы 1157, другие программные

модули 1156 и программные данные 1155 обозначены другими ссылочными позициями для иллюстрации того, что, как минимум, они являются различными копиями. Пользователь может вводить команды и информацию в компьютер 1102 через устройства ввода, такие как клавиатура 1151 и указательное устройство 1152, обычно упоминаемое как мышь, трекбол или сенсорная панель. Другие устройства ввода (не показаны) могут включать в себя микрофон, джойстик, игровую панель, спутниковую антенну, сканер, сканер сетчатки глаза или тому подобное. Эти и другие устройства ввода часто соединяются с блоком 1159 обработки через интерфейс 1136 пользовательского ввода, который связан с системной шиной 1121, но могут быть соединены посредством других интерфейсов и шинных структур, таких как параллельный порт, игровой порт или универсальная последовательная шина (USB). Монитор 1142 или другой тип устройства отображения также соединен с системной шиной 1121 через интерфейс, такой как видео интерфейс 1132. В дополнение к монитору, компьютеры могут также включать в себя другие периферийные устройства вывода, такие как динамики 1144 и принтер 1143, которые могут быть подсоединены через интерфейс 1133 периферийных устройств вывода.

[0105] Компьютер 1102 может работать в сетевой среде, используя логические соединения с одним или более удаленными компьютерами, такими как удаленный компьютер 1146. Удаленный компьютер 1146 может представлять собой персональный компьютер, сервер, маршрутизатор, сетевой PC, одноранговое устройство или другой общий сетевой узел, и обычно включает в себя многие или все из элементов, описанных выше относительно компьютера 1102, хотя только устройство 1147 хранения памяти было проиллюстрировано на фиг. 11. Логические соединения, изображенные на фиг. 11, включают в себя локальную сеть (LAN) 1145 и глобальную сеть (WAN) 1149, но могут также включать в себя другие сети. Такие сетевые среды являются обычным явлением в офисах, компьютерных сетях предприятий, интранетах, Интернете и облачных вычислительных ресурсах.

[0106] При использовании в сетевой среде LAN, компьютер 1102 подключен к локальной сети 1145 через сетевой интерфейс или адаптер 1137. При использовании в сетевой среде WAN, компьютер 1102 обычно включает в себя модем 1105 или другие средства для установления связи через WAN 1149, такую как Интернет. Модем 1105, который может быть внутренним или внешним, может быть подключен к системной шине 1121 через интерфейс 1136 пользовательского ввода или другой соответствующий механизм. В сетевой среде, программные модули, изображенные по отношению к компьютеру 1102, или их части, могут храниться в удаленном устройстве хранения памяти. В качестве примера, а не ограничения, фиг. 11 иллюстрирует удаленные прикладные программы 1148, как находящиеся на устройстве 1147 памяти. Следует понимать, что показанные сетевые соединения являются примером, и могут быть использованы другие средства установления линии связи между компьютерами.

[0107] Фиг. 12 изображает примерную блок-схему последовательности операций для способа 1200 абстрагирования нативной платформы анклава. Платформа абстракции, такая как 912 на фиг. 9, может принимать запрос для платформы анклава в блоке 1202. Запрос может исходить от клиента анклава, такого клиент 914, или от анклава, такого как анклав 916.

[0108] Запрос от анклава может представлять собой запрос на выполнение примитива платформы абстракции и может включать в себя, например, запрос, чтобы: создавать отчет аттестации или котировку анклава; запечатывать данные в анклав; вызывать функцию в клиент анклава (вызывать вовне в клиент); считывать монотонный счетчик

(обеспечивать текущее значение монотонного счетчика); обеспечивать доверенное измерение времени; и распределять память, которая может совместно использоваться между анклавом и его клиентом (например, чтобы разрешить указателю на совместно используемую память пересылаться в качестве параметра при вызове в анклав и из анклава). В некоторых вариантах осуществления, все пространство виртуальной памяти клиента анклава может совместно использоваться с анклавом (и быть доступным из анклава), так что запрос на распределение совместно используемой памяти может быть реализован как запрос на распределение памяти для клиента анклава. В некоторых вариантах осуществления, способы распределения совместно используемой памяти доступны как анклаву, так и его клиенту.

[0109] Запрос от клиента анклава может представлять собой запрос выполнить примитив платформы абстракции и может включать в себя, например, запрос, чтобы: реализовывать анклав; верифицировать отчет аттестации анклава; вызывать функцию внутри анклава (вызов в анклав); и распределять память, которая может совместно использоваться между анклавом и его клиентом.

[0110] Запрос платформы абстракции может переводиться в запрос нативной платформы в операциях 1204-1208. Параметры, включенные или предполагаемые в принятом запросе, могут быть преобразованы в дополнительном этапе 1204, если это определено, например, что формат данных параметра в исходном запросе не является тем же самым, что и формат данных для этого параметра в нативной платформе. Например, если запрос от анклава или клиента включает в себя параметр, выведенный из отчета аттестации формата абстракции, такой как абстрактный идентификатор анклава, он будет преобразован в параметр, используемый в отчете аттестации нативного формата, такой как нативной идентификатор анклава.

[0111] Если определено, что соглашение о вызове нативной платформы и принятый запрос отличаются, соглашение о вызове может быть преобразовано на опциональном этапе 1206. Соглашение о вызове может быть преобразовано, например, путем переупорядочивания параметров на стеке вызова, перемещения параметров между регистрами процессора и стеком вызова и преобразования между способами сообщения условия ошибки, такими как возвращение условия ошибки и вызов обработчика исключений.

[0112] В некоторых вариантах осуществления, нативная платформа может быть идентична платформе абстракции для некоторых запросов, и в этом случае операции преобразования блоков 1204 и 1206 могут быть пропущены.

[0113] В блоке 1208, преобразованный запрос отправляется на нативную платформу, чтобы вызвать выполнение запроса нативной платформой. Например, в случае, где нативная платформа соответствует архитектуре анклава Расширений безопасности программного обеспечения (SGX) от Intel, нативная платформа может включать в себя инструкции процессора для анклавов. В этом случае, отправка запроса в блоке 1208 может включать в себя исполнение одной или нескольких инструкций процессора для анклавов. В другом примере, нативная платформа может соответствовать архитектуре анклава Виртуального безопасного режима (VSM) от Microsoft, которая может включать в себя гипервизор с гипервызовами для анклавов. Гипервызов представляет собой программное прерывание кода гипервизора, и гипервызов может вызывать изменение контекста процессора на контекст, в котором могут быть разрешены привилегированные операции. В этом примере VSM, отправка запроса в блоке 1208 может включать в себя выполнение гипервызовов в гипервизор и/или другие механизмы, чтобы переключить контекст исполнения на контекст, в котором могут быть разрешены привилегированные

операции.

[0114] Отправка запроса на нативную платформу здесь обычно обозначает выполнение запроса с использованием характеристик нативной платформы. Операция отправки запроса на нативную платформу 1208 может предусматривать множество операций с нативной платформой и может варьироваться в зависимости от запрошенной операции (или примитива), такой как создание анклава, аттестация, запечатывание данных, перенос управления или использование монотонных счетчиков и доверенного времени.

[0115] Примитив CreateEnclave может использоваться, чтобы реализовать анклав. Запрос CreateEnclave, чтобы реализовать анклав, может побудить платформу абстракции создать безопасный контейнер (например, путем распределения некоторой памяти и установления границы безопасности или изоляции для этой памяти), копировать код анклава в этот безопасный контейнер (например, из образа анклава) и конфигурировать или задействовать точки входа в код анклава (например, в соответствии с метаданными точек входа в образе анклава).

[0116] Отправка запроса CreateEnclave на нативную платформу при помощи обеспеченного анклавом гипервизора (гипервизора, который обеспечивает функции администрирования анклава, такого как VSM), может включать в себя распределение памяти и побуждение гипервызовов устанавливать таблицы страниц процессора для памяти таким образом, чтобы предотвратить доступ кода вне контейнера анклава к этой памяти. Гипервызовы создания анклава из платформы абстракции могут также побуждать гипервизор устанавливать информацию конфигурации для переноса управления в анклав в назначенных точках входа. Затем, код вне безопасного контейнера может осуществлять гипервызовы переноса управления, чтобы перенести исполнение в назначенных точках входа внутри безопасного контейнера.

[0117] Отправка запроса CreateEnclave на нативную платформу при помощи задействованного анклавом процессора (процессора с процессорными инструкциями администрирования анклава, такого как SGX), может включать в себя исполнение платформой абстракции инструкции, такой как ECREATE, чтобы информировать CPU, что определенную область памяти следует создать в качестве безопасного контейнера анклава, и исполнение инструкции, такой как EADD, чтобы добавить страницы данных в этот контейнер анклава. Специальные процессорные инструкции могут также использоваться, чтобы создавать специальные страницы в памяти для назначения точек входа в анклаве для переноса управления в анклав. Затем, код вне безопасного контейнера может исполнить инструкцию, такую как EENTER, специфицирующую одну из назначенных точек входа, чтобы перенести управление исполнением в эту точку входа анклава.

[0118] Примитив CreateReport может использоваться, чтобы создавать отчет аттестации. Запрос CreateReport создать отчет аттестации анклава может выполняться уровнем абстракции, как пояснено выше в отношении фиг. 5 и 7. При помощи задействованного анклавом гипервизора, уровень абстракции может отправлять запрос на нативную платформу путем осуществления гипервызова, который изменяет состояние исполнения, например, на контекст контроля безопасности, который имеет доступ к секретному ключу, такому как PrivAK726 согласно фиг. 7, который может использоваться, чтобы подписывать отчеты. Этот секретный ключ может быть доступным только контексту контроля безопасности, если компьютер был загружен в работоспособной конфигурации, как верифицировано на основе TCG log в TPM. Контроль безопасности может маркировать данные отчета идентификационной

информацией удостоверяемого анклава.

[0119] При помощи задействованного анклавом процессора, запрос CreateReport может быть отправлен на нативную платформу путем исполнения инструкции, такой как EREPORT, которая генерирует отчет и отправляет его на специальный анклав, который будет иметь доступ к конфиденциальному ключу для подписания отчетов.

[0120] Примитив EnclaveSeal может использоваться, чтобы запечатывать данные в анклав. Запечатывание данных в анклав зашифровывает данные таким способом или при помощи ключа, который ассоциирован с анклавом. Запрос EnclaveSeal может представлять собой запрос запечатать данные, находящиеся внутри анклава, такие как все или часть состояния анклава, в анклаве с использованием политики запечатывания. Политика запечатывания, такая как SEALING_POLICY, описанная выше, может специфицировать тип идентификационной информации анклава, который указывает, какие анклавы должны иметь возможность распечатывать (вскрывать) данные. Сами процессы запечатывания могут использовать ключ шифрования, ассоциированный с идентификационной информацией анклава, специфицированной в политике запечатывания. Затем, реализация нового анклава может быть способна вскрывать данные, если значение идентификационной информации нового анклава в специфицированном типе идентификационной информации является тем же самым, что и значение идентификационной информации анклава запечатывания в специфицированном типе идентификационной информации.

[0121] Запечатывание данных разрешает копировать секретные или конфиденциальные данные анклава безопасным образом в небезопасное хранилище, например, в память вне безопасного контейнера анклава или в постоянное хранилище, такое как жесткий диск. Когда запечатанные данные представляют собой данные состояния анклава, запечатывание разрешает анклаву быть восстановленным в предыдущее состояние, и разрешает прерывание безопасной операции анклава и дальнейшее продолжение в другом анклаве.

[0122] Чтобы восстановить состояние анклава, сначала состояние анклава сохраняется путем запечатывания его состояния в анклав. Запечатывание может выполняться путем шифрования данных состояния при помощи ключа, ассоциированного с анклавом. Затем, возможно после того, как состояние анклава изменилось, запечатанные данные состояния могут быть вскрыты в том же самом анклаве путем дешифрования запечатанных данных и затем замены текущего состояния анклава дешифрованными данными (например, путем копирования дешифрованных данных в безопасный контейнер анклава).

[0123] Чтобы прервать безопасную операцию и продолжить в другом анклаве, безопасная операция запускается путем исполнения операции, содержащей множество процессорных инструкций в первом анклаве. Когда первый анклав прерывается, состояние этого анклава может быть запечатано по идентификатору анклава, заданному в политике запечатывания, и запечатанные данные могут затем быть сохранены в незащищенном хранилище, таком как локальное или облачное постоянное хранилище. Первый анклав может затем быть (опционально) завершен или может запускать другие операции анклава. Второй анклав может быть создан или перепрофилирован, чтобы продолжить прерванную операцию путем распечатывания запечатанных данных состояния во второй анклав. Прерванная операция может быть продолжена во втором анклаве, где остановился первый анклав.

[0124] При помощи задействованного анклавом гипервизора, уровень абстракции может отправить запрос EnclaveSeal в нативную платформу путем осуществления

гипервызова. Гипервызов может изменять состояние исполнения на контекст, например, контекст контроля безопасности, который будет иметь доступ к секретному ключу запечатывания, ассоциированному с анклавом, который может использоваться, чтобы запечатывать или распечатывать данные. Ключ запечатывания может извлекаться из комбинации идентификационной информации анклава и секретного ключа платформы, доступного только контролю безопасности. Этот ключ платформы может быть доступен только контролю безопасности, когда машина загружается в работоспособной конфигурации, и конфигурация загрузки верифицируется при помощи TCG log на основе TPM. В этом варианте осуществления задействованного анклавом гипервизора, код анклава никогда не имеет доступа к ключу запечатывания.

[0125] При помощи задействованного анклавом процессора, запрос EnclaveSeal может быть отправлен на нативную платформу путем исполнения инструкции, такой как EGETKEY, чтобы получить ключ шифрования. Этот алгоритм может генерировать ключ запечатывания, который является уникальным для анклава. Ключ запечатывания может быть выведен из идентификационной информации анклава и секрета, встроенного в процессор. Код внутри анклава может затем зашифровывать данные при помощи ключа запечатывания. Данные могут быть запечатаны путем шифрования при помощи ключа запечатывания, например, кодом внутри анклава, платформой абстракции или нативной платформой. EnclaveUnseal может аналогично использовать EGETKEY, чтобы генерировать ключ распечатывания.

[0126] Запрос на перенос управления может представлять собой запрос перенести управление исполнением процессора из инструкций внутри анклава на точку входа вне анклава (например, CallEnclaveOut) или обратно из инструкций вне анклава в точку входа внутри анклава (например, CallEnclaveIn). Это может быть сделано, например, для безопасной операции базы данных. После реализации анклава базы данных, клиент анклава может запросить, чтобы анклав выполнил конкретную операцию, такую как запрос базы данных с использованием примитива CallEnclaveIn, чтобы перенести управление в точку входа внутри анклава базы данных, который будет выполнять запрос. После того, как анклав завершает запрос, результат запроса может быть возвращен (возможно, после шифрования результата) на клиент при помощи примитива CallEnclaveOut, чтобы перенести управление обратно на клиент в точке входа в клиент, который может принять результат запроса. Примитивы CallEnclaveIn и CallEnclaveOut могут брать указатель на буфер памяти, который может совместно использоваться между анклавом и его клиентом (буфер может быть считываемым, записываемым и/или исполняемым анклавом или его клиентом).

[0127] При помощи задействованного анклавом гипервизора, уровень абстракции может отправить запрос CallEnclaveIn на нативную платформу путем выполнения гипервызова. Гипервызов может изменять состояние исполнения на контекст, например, контекст контроля безопасности, который сохранит регистры CPU, восстановит ранее сохраненный набор значений регистров CPU анклава (возможно, из памяти анклава), изменит конфигурацию таблицы страниц, чтобы разрешить доступ к защищенной памяти анклава, и вызовет точку входа функции внутри анклава. Аналогично, когда платформа абстракции принимает запрос CallEnclaveOut, запрос может быть отправлен на нативную платформу посредством гипервызова, который сохранит регистры CPU анклава (возможно, путем сохранения в память анклава) и восстановит сохраненные ранее регистры CPU для клиента анклава, изменит конфигурацию таблицы страниц, чтобы предотвратить доступ к памяти анклава, и перенесет управление процессором в точку входа в клиенте анклава вне анклава.

[0128] При помощи задействованного анклавом процессора, запрос CallEnclaveIn может быть отправлен на нативную платформу путем исполнения инструкции, такой как EENTER, которая может побудить CPU восстановить набор регистров CPU анклава (возможно, из памяти анклава) и вызвать функцию (перенести управление в точку входа) внутри анклава. Примитив CallEnclaveOut может исполнять инструкцию, такую как EEXIT, которая может переносить управление на инструкции вне анклава и/или вызывать сбой, который переносит управление вне анклава.

[0129] Монотонный счетчик имеет множество применений. Например, анклав может пожелать ограничить то, насколько далеко назад его состояние может быть реверсировано. Монотонные счетчики могут использоваться, например, в качестве случайного слова, чтобы гарантировать свежесть сообщений, как обсуждалось выше в отношении фиг. 4. Монотонные счетчики обычно имеют возможность считывания и приращения, но не могут уменьшать значение. Чтобы ограничить возврат или реверсирование состояния анклава, код внутри анклава может сообщать приращение ассоциированному монотонному счетчику и затем сохранять значение счетчика совместно с внутренним состоянием анклава. Состояние и значение счетчика могут быть сохранены, например, при помощи примитива EnclaveSeal. Далее, при восстановлении состояния анклава, например с использованием примитива EnclaveUnseal, код внутри анклава может считать текущее значение монотонного счетчика и сравнивает его со значением счетчика с незапечатанным состоянием. Если значение счетчика с незапечатанным состоянием меньше, чем текущее значение счетчика, анклав может предотвратить использование незапечатанного состояния.

[0130] При помощи задействованного анклавом гипервизора, уровень абстракции может отправить запрос на нативную платформу, чтобы считать или сообщить приращение монотонному счетчику путем выполнения гипервызова, который предоставлен анклаву. Когда вызывается гипервызов, чтобы считать или сообщить приращение счетчику, процессор изменит состояние исполнения на контекст, такой как контроль безопасности, который будет верифицировать идентификационной информации анклава, выполняющего гипервызов, и затем выполнять считывание или приращение, соответственно, соответствующего монотонного счетчика, сохраненного, например, в энергонезависимом хранилище, таком как чип TPM. Альтернативно, контроль безопасности может выполнять считывание или приращение счетчика, сохраненного на удаленном доверенном сервере или наборе удаленных доверенных серверов, путем установления безопасного канала связи с таким сервером и запрашивания его выполнить считывание или приращение заданного монотонного счетчика. Удаленный доверенный сервер или серверы могут поддерживать счетчик внутри анклава, чтобы изолировать его от остальной части кода в компьютере-хосте.

[0131] При помощи задействованного анклавом процессора, запрос может быть отправлен на нативную платформу путем исполнения инструкции. При помощи такого процессора, примитивы монотонного счетчика могут быть реализованы путем считывания или приращения счетчика в хранилище энергонезависимой памяти в чипе на материнской плате компьютера. Альтернативно, эти примитивы могут также быть реализованы с использованием доверенного удаленного сервера как с задействованным анклавом гипервизором.

[0132] Фиг. 13 изображает примерную блок-схему последовательности операций для способа 1300 абстрагирования нативной платформы анклава. Платформа абстракции анклава может принимать запрос от анклава или клиента анклава в блоке 1302. В блоке 1304, платформа абстракции может определять, включает ли в себя нативная платформа

процессорные инструкции анклава, например, путем определения, соответствует ли нативная платформа SGX. Если она соответствует, то процессорные инструкции анклава исполняются в блоке 1306. В блоке 1308, платформа абстракции может определить, включает ли в себя нативная платформа гипервызовы анклава, например, путем определения, соответствует ли нативная платформа VSM. Если она соответствует, то нативная платформа выполняет гипервызовы анклава. Результаты исполнения инструкций анклава или вызова гипервызовов анклава очищаются в блоке 1312. Очистка может включать в себя, например, преобразование параметров вывода или обработку исключения процессорных инструкций анклава или гипервызовов анклава в формат или протокол интерфейса уровня абстракции. Преобразованные параметры вывода затем возвращаются на исходный запросчик (анклав или клиент) в блоке 1314.

Абстрактная идентификация анклава

[0133] Фиг. 14 изображает примерные двоичные образы анклава, используемые для создания экземпляра анклава. Экземпляр анклава может быть создан путем создания безопасного контейнера, такого как контейнер 1490 анклава, и копирования частей одного или нескольких образов анклава в этот контейнер. Контейнер 1490 анклава может быть создан со ссылкой на первичный образ 1410 анклава. Первичный образ может включать в себя ссылки на другие зависимые образы анклава. В этом примере, первичный образ 1410 включает в себя зависимость1 (Dependency1) и зависимость2 (Dependency2) в качестве ссылок на зависимые образы 1420 и 1450 анклава, соответственно. Образ 1420 включает в себя дополнительно зависимости от образов 1430 и 1440, в то время как образ 1450 зависит от образа 1460. Когда все эти образы (или их части) скопированы в контейнер 1490, результирующий анклав может включать в себя неплатформенные образы 1402, которые могут включать в себя код и данные, которые являются уникальными или индивидуальными для конкретного реализованного анклава, образа платформы 1404 абстракции и образа 1406 нативной платформы.

[0134] Каждый образ анклава, такой как первичный образ 1410, может включать в себя ID, зависимости, код, данные и подпись автора образа. В примере образа 1410, включены два ID 1410.1 и 1410.2. Эти ID могут представлять собой UUID, которые задают, например, значение абстрактного идентификатора, соответствующее значению ImageID, FamilyID или AuthorID, которые, по отдельности или вместе, могут использоваться, чтобы идентифицировать любой анклав, экземпляр которого создан при помощи этого образа анклава. Как изображено, образ 1410 имеет два ID, но возможно меньше или больше ID. Код в образе 1410 может представлять собой двоичные инструкции, исполняемые процессором компьютера, хостирующего контейнер 1490 анклава. Данные в образе 1410 могут использоваться любым кодом, загруженным в контейнер 1410. Образ 1410 может также включать в себя подпись Sig 1410, чтобы обеспечить целостность любого или всего из другого содержимого образа, таких как ID, ссылки зависимостей, код и данные. Другие образы 1420-1460 могут аналогично содержать ID, ссылки зависимостей, код, данные и подписи.

[0135] Указатель зависимости, такой как зависимость1 и зависимость 2 образа 1410, зависимость1 и зависимость 2 образа 1420 и зависимость 1 образа 1450, может быть задан множеством способов. Если образы 1410-1460 хранятся в компьютерной памяти системы, указатель зависимости может просто представлять собой адрес в памяти. Если образы анклава представляют собой файлы в файловой системе, ссылки могут представлять собой имена файлов. В некоторых вариантах осуществления, ссылки могут представлять собой логический идентификатор. Логический идентификатор может представлять собой уникальное число, такое как UUID, или может представлять

собой другие данные, такие как текстовая строка, которая иначе идентифицирует образ зависимости. Например, текстовая строка может указывать автора, источник, имя продукта, семейство продукта и/или номер версии зависимого двоичного образа. Логический идентификатор включает в себя веб- или интернет-местоположение, такое как местоположение, откуда может извлекаться копия зависимого двоичного кода.

[0136] В некоторых вариантах осуществления, файл образа анклава может быть локализован путем поиска указателя зависимости, такого как логический идентификатор, в регистре образов анклава, чтобы найти указатель на текущую версию или локальную копию отыскиваемого по ссылке образа анклава. В некоторых случаях, доверенная услуга может использоваться для разрешения указателя зависимости в идентификацию конкретного образа анклава или местоположения образа.

[0137] В некоторых вариантах осуществления, указатель зависимости может представлять собой криптографически безопасный идентификатор, такой как криптографический хеш предполагаемого зависимого двоичного образа анклава. Такой хеш может включать в себя весь зависимый двоичный код или только его часть. Часть зависимого двоичного кода, включенная в указатель зависимости, может включать в себя значения абстрактного идентификатора, такие как ID 1410.1 или ID 1420.2, и может представлять собой значения абстрактного идентификатора. Услуга разрешения для криптографически безопасного идентификатора может не обязательно быть настолько же доверенной, что и для логического идентификатора, поскольку субъект, определяющий зависимости анклава, может быть способен верифицировать, что корректное зависимый образ был найдено вычислением хеша самого зависимого двоичного кода.

[0138] Фиг. 15 изображает примерную блок-схему последовательности операций для способа 1500 выполнения операции анклава с абстрактным идентификатором анклава. Значение абстрактного идентификатора для анклава может обеспечивать основу для определения эквивалентности между двумя анклавами, которые имеют некоторое общее свойство, но не являются абсолютно идентичными. Значение идентификатора может быть включено в отчет аттестации и может быть ассоциировано с типом абстрактного идентификатора (таким ExactHash, InstanceHash, ImageID, FamilyID или AuthorID). Два анклава, которые не являются абсолютно одинаковыми, могут иметь одно и то же значение абстрактного идентификатора для типа абстрактного идентификатора. Дополнительно, идентичный код анклава, реализованный в безопасных контейнерах на двух разных нативных платформах анклава, может также иметь то же самое значение абстрактного идентификатора. Способ 1500 может выполняться, например, уровнем платформы абстракции между нативной платформой анклава и либо анклавом, либо клиентом анклава.

[0139] В блоке 1502, анклав реализуется из образа анклава, такого как первичный образ 1410 анклава на фиг. 14. Образ анклава может представлять собой первичный образ, включающий в себя код анклава, данные, список идентификаторов, список любых зависимых образов анклава и подпись. Чтобы обеспечить целостность образов анклава, образы могут быть подписаны при помощи конфиденциального ключа, который может соответствовать автору образа анклава. Список ID идентификационной информации анклава в образе анклава может соответствовать типам абстрактной идентификационной информации, таким как ImageID, FamilyID и AuthorID, каждый из которых предназначен, чтобы идентифицировать образ анклава сообща совместно с другими связанными образами анклава. Список зависимых образов анклава может ссылаться на другие образы анклава, содержащие код анклава, от которых зависит

код в первичном образе анклава. Зависимый образ анклава может быть или может не быть создан тем же самым автором, и некоторые зависимые образы анклава могут быть в общем виде ассоциированы с платформой анклава (платформой абстракции или нативной платформой), а не конкретно ассоциированы с первичным образом анклава или автором первичного образа анклава. Анклав может быть реализован путем создания безопасного контейнера анклава с использованием любой нативной платформы анклава и копированием всего или части первичного образа и любых зависимых образов анклава в безопасный контейнер.

[0140] В блоке 1503, запрашивается операция анклава, например, анклавом или клиентом анклава, совместно с типом идентификационной информации анклава. Тип идентификационной информации может задавать тип абстрактной идентификационной информации, такой как Image ID или AuthorID, и быть связан с конкретным реализованным анклавом, но не задает значение AuthorID для этого анклава. Остальная часть способа 1500, следующая за блоком 1503, описывает операции для выполнения операции (такой как аттестация, запечатывание данных или использование монотонного счетчика и т.д.) с реализованным анклавом с использованием значения идентификационной информации, полученного для этого типа идентификационной информации реализованного анклава. Идентификатор может определяться с использованием хеша поднабора образов анклава. То, какой поднабор образов анклава используется в качестве ввода в хеш, может быть основано частично на типе идентификационной информации, желательном для использования в операции анклава.

[0141] В блоке 1504, часть образа анклава, называемая здесь идентификационной частью, определяется на основе типа идентификационной информации. Идентификационная часть может включать в себя все, часть или ничего из различных двоичных образов анклава, используемых, чтобы реализовывать анклав в блоке 1502. Идентификационная часть может включать в себя все, часть или ничего из кода анклава, содержащегося в образе анклава. Идентификационная часть может также включать в себя ноль, один или несколько ID идентификационной информации, перечисленных в некодовой части включенных образов анклава. Идентификационная часть также может включать или может не включать в себя данные анклава, содержащие образы анклава. Идентификационная часть может включать в себя любую комбинацию этих различных частей образов анклава. Например, идентификационная часть может включать в себя весь код, ничего из данных и два или четыре доступных ID идентификационной информации. В опциональном блоке 1506 определяется, какие зависимые образы анклава должны быть включены в идентификационную часть, и определяется идентификационная часть каждого включенного образа.

[0142] Идентификационная часть зависимых образов может быть или может не быть той же самой, что и идентификационная часть первичного образа анклава. Например, весь код и ImageID включены в идентификационную часть первичного образа, в то время как код не может и только FamilyID зависимого образа может включаться в идентификационную часть зависимого образа.

[0143] Когда код анклава включен в идентификационную часть, части кода анклава в идентификационной части могут определяться комбинацией типа идентификационной информации и указания того, какие зависимости должны быть включены в идентификационную часть. Тип идентификационной информации InstanceHash может включать в себя, например, код анклава в первичном образе, но не в зависимых образах, в то время как тип идентификационной информации ExactHash может включать в себя код анклава во всех зависимых образах, которые не являются рассматриваемой частью

платформы анклава. Например, все зависимые образы анклава, которые не подписаны закрытым ключом автора платформы анклава, могут рассматриваться как не являющиеся частью платформы анклава. Альтернативно или дополнительно, первичный образ может включать в себя список того, какие зависимые образы анклава должны быть включены или исключены в идентификационной части для типов идентификационной информации InstanceHash или ExactHash.

[0144] ID идентификационной информации анклава, которые могут быть включены в качестве метаданных в образе анклава, могут быть включены в часть идентификационной информации образа анклава вместо кода анклава или дополнительно к нему. Например, часть идентификационной информации для типов идентификационной информации ImageID, FamilyID и AuthorID может включать в себя соответствующие метаданные ID из образа анклава. Когда типы идентификационной информации вложены или упорядочены по уровням, часть идентификационной информации для типов более низкого уровня может включать в себя данные ID для типов более высокого уровня. Например, часть идентификационной информации для ImageID может включать в себя данные ID для ImageID, FamilyID и AuthorID, в то время как часть идентификационной информации для AuthorID может включать в себя только данные ID для AuthorID.

[0145] Типы идентификационной информации, которые включают в себя код анклава, такие как InstanceHash и ExactHash, обеспечивают более высокий уровень уверенности, например, клиенту анклава посредством аттестации, что определенный код анклава работает внутри анклава. Однако идентификационная информация анклава обязательно изменится, когда изменяется любая часть идентификации кода анклава. Например, если дилемма с безопасностью или другая неполадка зафиксирована в новой версии образа анклава, результирующее значение идентификатора, основанное на новом коде, также изменится. Путем обеспечения механизма для исключения определенных частей кода анклава из вычисления хеша идентификатора, идентификатор анклава может быть отделен от изменений в исключенной части кода анклава. Например, когда код анклава одного автора зависит от кода анклава, обеспеченного платформой анклава, идентификатор анклава может быть отделен от пересмотров для зависимой платформы.

[0146] В блоке 1508, определяется значение идентификатора, которое может представлять идентификатор анклава, реализованный в блоке 1502. Значение идентификационной информации может определяться вычислением хеша по ранее определенной идентификационной части образа или образов анклава (значение идентификационной информации представляет собой вывод хеш-функции, где идентификационная часть представляет собой ввод в хеш-функцию). В некоторых вариантах осуществления, ввод в хеш-функцию будет представлять собой части исходного образа(ов) анклава, в то время как в других вариантах осуществления, ввод в хеш-функцию будет представлять собой части контейнера анклава после копирования идентификационной части образа в контейнер (и возможно дешифрования идентификационной части в случае, когда исходный образ анклава зашифрован).

[0147] В блоке 1510, целостность результирующего значения идентификационной информации может быть опционально верифицирована путем верифицирования целостности исходного образа(ов) анклава. Целостность образа анклава может быть верифицирована при помощи общедоступного ключа, соответствующего конфиденциальному ключу, используемому, чтобы подписывать образ анклава. Такая пара общедоступного/конфиденциального ключей может быть ассоциирована, например, с автором образа(ов) анклава, так что доверие в отношении результирующего значения

идентификационной информации может быть основано на доверии к автору анклава.

[0148] Наконец, в блоке 1512, операция, связанная с реализованным анклавом, может выполняться с использованием значения идентификационной информации. Например: отчет аттестации реализованного анклава может генерироваться или верифицироваться для типа идентификационной информации; данные могут быть запечатаны в или распечатаны из реализованного анклава с идентификационной информацией; и могут быть использованы монотонный счетчик или доверенное время, привязанное к реализованному анклаву и типу идентификационной информации.

[0149] Операции анклава с использованием типов идентификационной информации более высокого уровня обеспечивают возможность взаимодействий между группами возможных реализаций анклава. Аттестация для типа идентификационной информации высокого уровня может обеспечивать эквивалентность отчетов аттестации для всех анклавов с той же самой идентификационной информацией высокого уровня. Например, отчет аттестации для типа идентификационной информации AuthorID может быть эквивалентным отчету аттестации от всех анклавов, реализованных из первичного образа, содержащего те же самые метаданные AuthorID. Данные, запечатанные на тип идентификационной информации высокого уровня, могут быть распечатаны любым анклавом с тем же самым значением идентификационной информации высокого уровня. Например, данные, запечатанные в реализованный анклав с типом идентификационной информации AuthorID, могут быть распечатаны любым другим реализованным анклавом с теми же самыми метаданными AuthorID в его первичном образе анклава.

Эквивалентность идентификации анклава

[0150] Фиг. 16 изображает примерную систему с эквивалентностью абстрактной идентификационной информации анклава. Клиент 1602 анклава может осуществлять связь с первым анклавом 1612, реализованным в безопасном контейнере анклава первой нативной платформы 1616 посредством платформы 1614 абстракции, и клиент 1602 может также осуществлять связь со вторым анклавом 1622, реализованным в безопасном контейнере анклава второй нативной платформы 1626 посредством платформы 1624 абстракции. Первая нативная платформа 1616 может или не может находиться на том же самом компьютере, что и вторая нативная платформа. Клиент 1602 анклава может находиться на том же самом компьютере, что и любая из нативных платформ, или может находиться на отдельном третьем компьютере. Первая нативная платформа 1616 может быть не той же самой, что и вторая нативная платформа 1626. Например, первая нативная платформа 1616 может быть более ранней версией второй нативной платформы от того же самого производителя нативной платформы. Альтернативно, первая нативная платформа 1616 и вторая нативная платформа 1626 могут соответствовать полностью разным архитектурам анклава, таким как VSM и SGX.

[0151] Клиент анклава может безопасно определять, что анклавы являются эквивалентными, путем сравнения значений идентификационной информации, полученных из отчетов аттестации. Клиент 1602 анклава может безопасно идентифицировать каждый из анклавов путем приема отдельных отчетов аттестации от первого анклава 1612 и второго анклава 1622. Значение идентификационной информации может быть включено в (или получено из) каждый(ого) из этих отчетов аттестации. Если значения идентификационной информации являются одинаковыми, клиент 1602 анклава может иметь уверенность в том, что первый анклав 1612 и второй анклав 1622 эквивалентны в некотором смысле. Значения идентификационной информации из отчетов аттестации могут представлять собой значения абстрактной идентификационной информации, соответствующие конкретному типу абстрактной

идентификационной информации (такому как ExactHash, InstanceHash, ImageID, FamilyID или AuthorID) или хешам таких значений абстрактной идентификационной информации. В этом случае, эквивалентность может определяться, когда анклавы не абсолютно идентичны. Два анклава могут не быть абсолютно идентичными, но все еще определяться как эквивалентные, например, когда образы анклава, загруженные в контейнер анклава, представляют собой разные версии той же самой функциональности или те же самые первичные образы с разными зависимыми образами или те же самые образы анклава, загруженные в контейнеры анклава разных архитектур анклава.

[0152] Первый анклав 1612 может рассматриваться эквивалентным, но не идентичным второму анклаву 1622 для множества ситуаций. В первом примере, только поднабор кода, первоначально загруженного в контейнеры анклава, является тем же самым (например, эквивалентным для типов ExactHash или InstanceHash абстрактной идентификационной информации). Во втором примере, автор кода анклава мог включить идентичный ID в два разных двоичных образа анклава, даже хотя код в двух двоичных образах является разным (например, эквивалентным для типов идентификационной информации ImageID, FamilyID или AuthorID). В третьем примере, код в каждом анклаве является полностью тем же самым, но загружен (реализован) на различные нативные платформы. В этом третьем примере, первая нативная платформа 1616 и вторая нативная платформа 1626 могут быть произведены разными производителями, и тем самым доверие в отношении разных отчетов аттестации основывается на разных органах сертификации (см. фиг. 7, элемент 738) разных производителей. Примером, где две нативных платформы различаются, является нахождение в группе серверов или в облачном вычислении, где серверы, распределенные для обработки рабочих нагрузок первого и второго анклавов, представляют собой серверы, которые не поддерживают одну и ту же нативную платформу анклава.

[0153] В альтернативном варианте осуществления, первый анклав может представлять собой клиент второго анклава, так что блоки 1602 и 1612 комбинируются. Определение эквивалентности анклава в этом варианте осуществления может включать в себя определение, в пределах первого анклава, что значение идентификационной информации из отчета аттестации второго анклава является тем же самым, что и собственное значение идентификационной информации первого анклава (на конкретном уровне абстрактной идентификационной информации).

[0154] Фиг. 17 изображает примерную блок-схему последовательности операций для параллельной обработки с двумя эквивалентными анклавами. Процесс 1700 может выполняться, например, клиентом двух или более разных анклавов. В блоке 1702, два анклава реализованы на разных экземплярах нативной платформы, например, как изображено на фиг. 16. Анклавы могут быть реализованы клиентом анклава, задающим двоичный образ анклава (такое как первичный образ 1410 на фиг. 14) посредством метода CreateEnclave платформ 1614 и 1624 абстракции. Двоичный образ анклава, заданный для создания экземпляров двух анклавов, может быть тем же самым или другим. Отчет аттестации для каждого реализованного анклава создается в блоке 1704. Отчеты аттестации могут создаваться, например, в запросе клиента анклава или в запросе самих анклавов. Субъект, желающий доказать эквивалентность двух анклавов, такой как клиент анклава, получает копии обоих отчетов аттестации. Отчеты аттестации могут быть опционально верифицированы в блоке 1706. Например, целостность отчета может быть верифицирована путем верифицирования подписи аттестации при помощи сертификата подтверждения (такого как на фиг. 7, элемент 728) нативной платформы, которая создала отчет аттестации. Дополнительно, сертификат подтверждения может

быть верифицирован при помощи общедоступного ключа производителя нативной платформы (такого как на фиг. 7, элемент 732). Значения идентификатора (или их хеш) могут извлекаться из каждого отчета аттестации в блоке 1708, и эквивалентность двух анклавов может определяться путем верифицирования, что извлеченные значения идентификатора являются одними и теми же для каждого анклава. Эти значения идентификатора могут представлять собой значения абстрактного идентификатора (или их хеши), ассоциированные с типом идентификатора.

[0155] Когда клиент анклава доказал эквивалентность двух экземпляров анклава из операций в блоках 1708 и 1710, два анклава могут использоваться взаимозаменяемо, в соответствии с типом показанной эквивалентности. Блоки 1712-1720 изображают примерный способ использования эквивалентных анклавов для использования двух реализованных эквивалентных анклавов путем параллельной обработки. В блоках 1712 и 1716, часть набора данных ввода, такая как часть базы данных или часть файла цифрового образа, копируется в первый и второй анклав. Часть скопированного набора данных может быть идентичной или различной в соответствии с решаемой задачей обработки. Операция обработки может безопасно выполняться параллельно путем одновременного выполнения операции частично в первом анклаве в блоке 1714 и выполнения операции частично во втором анклаве в блоке 1718. Операцией может быть, например, выполнение поиска в базе данных или выполнение операции обработки образа. Первый анклав может выполнять поиск в первой половине базы данных или выполнять операцию обработки образа в отношении первой половины образа, в то время как второй анклав может выполнять поиск во второй половине базы данных или выполнять операцию обработки образа в отношении второй половины образа. Наконец, в блоке 1720, результаты параллельной обработки в первом и втором анклаве могут комбинироваться, например, путем комбинирования двух отсортированных половин базы данных или сведения двух половин образа вместе.

[0156] Фиг. 18 изображает примерную блок-схему последовательности операций для последовательной обработки при помощи двух эквивалентных анклавов. Как изображено на фиг. 18, операция анклава, такая как операция базы данных или операция обработки образа, создается безопасно в двух последовательных частях в двух отдельных анклавах. Процесс 1800 может выполняться, например, клиентом 1602 анклава согласно фиг. 16. В блоке 1802, первый анклав создается на первой нативной платформе анклава, и отчет аттестации первого анклава создается в блоке 1804. Этот первый отчет аттестации (первого анклава) может быть верифицирован в блоке 1806, например, как описано выше в отношении блока 1706 на фиг. 17. В блоке 1808, безопасная операция начинается в первом анклаве, но не завершается. Состояние первого анклава может дополнительно быть запечатано для безопасного удаления из первого анклава в блоке 1810. Например, первое состояние анклава может быть запечатано на тип идентификационной информации первого анклава. Когда состояние анклава сохранено, первый анклав может быть завершен (не изображено).

[0157] Второй анклав используется, начиная с блока 1812. В блоке 1812, второй анклав реализуется на второй нативной платформе. Как на фиг. 16 и 17, вторая нативная платформа может или не может хостироваться на том же самом компьютере, что и первая нативная платформа, и первая и вторая нативные платформы могут быть одними и теми же или разными. Отчет аттестации второй нативной платформы создается в блоке 1814, и, дополнительно, этот второй отчет аттестации может быть верифицирован в блоке 1816. Значение идентификационной информации из первого и второго отчетов аттестации может сравниваться в блоке 1818, чтобы верифицировать эквивалентность

первого и второго анклавов. В альтернативных вариантах осуществления, второй анклав может быть реализован, и эквивалентность верифицирована (блоки 1812-1818) до того, как безопасная операция начинается в первом анклаве в блоке 1808. Чтобы продолжить безопасную операцию, начатую в первом анклаве, запечатанное состояние из первого анклава может копироваться во второй анклав и распечатываться в блоке 1820. В блоке 1822, безопасная операция завершается во втором анклаве (с использованием состояния анклава, безопасно скопированного из первого анклава, если состояние было скопировано).

Распределенное запечатывание данных

[0158] Фиг. 19 представляет собой блок-схему примерной системы распределенного запечатывания данных. Запечатывание данных может распределяться по нескольким анклавам, где эти анклавы хостируются на отдельных нативных платформах анклава и/или на отдельных компьютерах. Как объяснено выше, примитивы EnclaveSeal и EnclaveUnseal абстракции могут запечатывать и распечатывать данные для анклава с использованием ключа, привязанного к нативной платформе анклава или физическому компьютеру, на котором работает анклав. Это может ограничить распечатывание только до анклавов, хостируемых на одном и том же компьютере или одном и том же экземпляре нативной платформы анклава. Фиг. 19 изображает систему распределенного запечатывания данных, где запечатывание или распечатывание данных может происходить на нативной платформе или компьютере, отличных от нативной платформы и компьютера, хостирующих анклав. Система 1900 включает в себя компьютеры 1910, 1930, 1950 с сетью 1902, соединяющей компьютеры 1910 и 1930, и сетью 1904, соединяющей компьютеры 1930 и 1950. Компьютер 1910 хостирует исходный анклав 1912, из которого могут исходить данные, подлежащие запечатыванию; компьютер 1930 хостирует анклав распределенного запечатывания (DSE) 1932 для обслуживания запросов распределенного запечатывания и распечатывания; и компьютер 1950 хостирует анклав 1952 места назначения, где распечатываются ранее запечатанные данные. Как объяснено выше в отношении фиг. 9, анклавы 1912, 1932, 1952 могут осуществлять связь с платформами 1914, 1934, 1954 абстракции, соответственно, посредством протокола абстракции анклава, и платформы 1914, 1934, 1954 абстракции могут осуществлять связь с нативными платформами 1916, 1936 и 1956, соответственно, посредством нативного протокола. В альтернативных вариантах осуществления, один или несколько анклавов 1912, 1932, 1950 могут хостироваться непосредственно на нативных платформах 1961, 1936, 1956 без промежуточной платформы абстракции. Запечатанные данные 1938 могут представлять собой данные, запечатанные в DSE 1932 с использованием ключа, ассоциированного с DSE 1932 или хостирующей его нативной платформой 1936. Запечатанные данные 1938 могут храниться в менее защищенном местоположении, например, на компьютере 1930 вне безопасного контейнера анклава DSE 1932, например, где-то в пространстве памяти компьютера 1930 или в файловой системе жесткого диска.

[0159] Распределенное запечатывание данных может включать в себя аутентификацию DSE 1930 для исходного анклава, например, путем аттестации DSE 1932 по сети 1902. Когда исходный анклав 1912 доверяет DSE 1932, исходный анклав 1912 может отправить конфиденциальные данные по безопасному каналу связи на DSE 1932 совместно с политикой запечатывания для запечатывания посредством DSE 1932. DSE 1932 может затем запечатать данные из анклава 1912 в него самого и сохранить запечатанные данные в небезопасном хранилище. Далее, анклав 1952 места назначения может запросить ранее запечатанные данные. Для распечатывания данных, DSE 1932 может

аутентифицировать анклав 1952 места назначения, например, путем аттестации по сети 1904, и верифицировать, что распечатывание для анклава 1952 места назначения разрешено в соответствии с политикой запечатывания, обеспеченной исходным анклавом 1912. DSE 1932 может распечатать ранее запечатанные данные из исходного анклава 1912 и затем отправить распечатанные данные по безопасному каналу связи на анклав 1952 места назначения. Данные анклава могут сообщаться безопасно на и от DSE 1932 путем шифрования данных анклава по сетям 1902 и 1904. Например, данные анклава, отправленные по сети 1902, могут быть зашифрованы при помощи ключа, сгенерированного во время аттестации DSE 1932, в исходный анклав 1912, и данные, отправленные по сети 1904, могут быть зашифрованы при помощи ключа, сгенерированного во время аттестации анклава 1952 места назначения, в DSE 1932. Возможны другие безопасные каналы связи, такие как шифрование при помощи общедоступного ключа места назначения, например, общедоступного ключа, ассоциированного с DSE, или общедоступного ключа, ассоциированного с анклавом места назначения.

[0160] Идентификаторы анклава, используемые в распределенном запечатывании и распечатывании, могут или не могут представлять собой абстрактные идентификаторы анклава. Например, в некоторых вариантах осуществления с уровнем платформы абстракции, политика запечатывания, такая как политика, заданная исходным анклавом и приводимая в исполнение посредством DSE, может идентифицировать разрешенные идентификаторы анклава распечатывания, где разрешенные идентификаторы анклава распечатывания представляют собой, например, список абстрактных идентификаторов анклава или список типов абстрактного идентификатора в комбинации со значениями абстрактных идентификаторов исходного анклава. В других ситуациях может использоваться неабстрактный идентификатор. Например, в некоторых вариантах осуществления, DSE может быть реализован при помощи открытого кода, так что доверие в отношении DSE представляет собой доверие в отношении знания его кода в противоположность доверию к автору его кода. В этом примере, аттестация DSE может представлять собой подписанный хеш всего открытого кода DSE, и ввод в хеш-функцию может не включать в себя значения абстрактной идентификационной информации, назначенные автором.

[0161] В некоторых вариантах осуществления, нативные платформы 1916, 1936, 1956 представляют собой отдельные нативные платформы, поскольку они хостируются на разных компьютерах 1910, 1930, 1950, даже если нативные платформы 1916, 1936, 1956 соответствуют той же самой версии той же самой архитектуры нативной платформы анклава. В других вариантах осуществления, нативные платформы 1916, 1936, 1956 могут соответствовать разным архитектурам платформы или разным версиям той же самой архитектуры нативной платформы анклава. Использование абстрактных идентификаторов в политике запечатывания может облегчать хостирование анклавов источника и места назначения на разных архитектурах нативной платформы.

[0162] В других вариантах осуществления распределенного запечатывания данных, не изображенных на фиг. 19, может не иметься трех отдельных компьютеров (таких как отдельные компьютеры 1910, 1930, 1950). Например, анклавы источника и места назначения могут находиться на одном компьютере (и возможно на одной нативной платформе), в то время как DSE находится на другом компьютере. С другой стороны, DSE может хостироваться на том же самом компьютере, что и компьютер, хостирующий исходный анклав, или компьютер, хостирующий анклав места назначения. В этих вариантах осуществления распределенного запечатывания данных, запечатывание и

распечатывание данных не является полностью локальным для одного компьютера, как описано выше в отношении примитивов абстракции EnclaveSeal и EnclaveUnseal.

[0163] Распределенное запечатывание данных может быть реализовано в уровне абстракции API, например, платформами 1914, 1934, 1954 абстракции. Например, примитивы DistributedEnclaveSeal и DistributedEnclaveUnseal аналогичны примитивам EnclaveSeal и EnclaveUnseal локального запечатывания данных, обсуждаемым выше.

```
DWORD
DistributedEnclaveSeal (
  _In_          SEALING_POLICY sealingPolicy,
  _In_reads_bytes_opt_(dwPlaintextSize) LPCVOID pPlaintext,
  _In_          DWORD  dwPlaintextSize,
  _In_reads_bytes_opt_(dwAuthdataSize) LPCVOID pAuthdata,
  _In_          DWORD  dwAuthdataSize,
  _Out_writes_bytes_to_(dwSealedtextSize) LPVOID pSealedtext,
  _Inout_       DWORD dwSealedtextSize,
      Set<EnclaveIdentity> SetOfTargetEnclaves
)
```

[0164] DistributedEnclaveSeal расширяет EnclaveSeal путем взятия дополнительного параметра SetOfTargetEnclaves, который разрешает вызов анклава, такого как анклав 1910, чтобы задавать набор идентификаторов анклава, которые авторизованы для распечатывания данных, обеспеченных посредством параметра pPlaintext. Если идентификаторы не обеспечены посредством SetOfTargetEnclaves, авторизованный идентификатор анклава по умолчанию может предполагаться как идентификатор анклава запечатывания, например ExactHash или InstanceHash анклава запечатывания.

[0165] Реализация DistributedEnclaveSeal, например, как метода платформы 1914 абстракции на компьютере исходного анклава, может включать в себя установление безопасного канала связи с DSE, например, путем шифрования сообщения по сети 1902. Ключ(и) для этого шифрования может, например, генерироваться во время процесса аттестации, как описано выше, или может представлять собой любой общедоступный ключ, ассоциированный с DSE 1932.

[0166] DistributedEnclaveSeal может быть дополнительно обобщен путем взятия дополнительного KeyForData параметра (не показан в прототипе функции DistributedEnclaveSeal выше). В некоторых вариантах осуществления, несколько наборов данных могут сохраняться запечатанными одновременно для одного анклава или одного идентификатора анклава. В этом случае, KeyForData разрешает спецификацию того, какой набор данных запечатывается. KeyForData может представлять собой любой вид идентификатора данных, такой как строка, число или набор параметров. В некоторых вариантах осуществления, KeyForData может представлять собой параметр ввода в DistributedEnclaveSeal и может генерироваться анклавом запечатывания, эффективно позволяя анклаву запечатывания именовать набор данных. В других вариантах осуществления, KeyForData может представлять собой параметр вывода, где DSE генерирует идентификатор KeyForData, когда данные запечатываются.

```
DWORD
DistributedEnclaveUnseal(
  _In_reads_bytes_opt_(dwSealedtextSize) LPCVOID pSealedtext,
  _In_              DWORD  wSealedtextSize,
_In_reads_bytes_opt_(dwAuthdataSize) LPCVOID pAuthdata,
_In_              DWORD  dwAuthdataSize,
_Out_writes_bytes_to_(dwPlaintextSize) LPCVOID pPlaintext,
_Inout_           DWORD  dwPlaintextSize
            Key   KeyForData,
                  EnclaveIdentity Identity
)
```

[0167] DistributedEnclaveUnseal может быть реализован в платформе 1954 абстракции и может работать в ответ на запрос из анклава 1952 места назначения. DistributedEnclaveUnseal может устанавливать безопасный канал связи с DSE 1932, например, путем шифрования сообщений при помощи ключа, сгенерированного во время аттестации анклава 1952 места назначения для DSE 1932, или путем шифрования сообщений, отправленных на анклав места назначения, при помощи общедоступного ключа анклава места назначения. DSE может верифицировать идентификатор анклава запрашивания (места назначения), например, путем аттестации, распечатать запрошенные запечатанные данные и безопасно отправить распечатанные данные на анклав запрашивания. В вариантах осуществления, где анклав запрашивания имеет несколько идентификаторов, конкретный идентификатор может быть указан в параметре идентификации. В вариантах осуществления, где несколько наборов данных анклава хранятся для одного идентификатора анклава, параметр KeyForData может указывать, какой набор запечатанных данных (для указанного идентификатора) запрашивается, с использованием того же самого значения KeyForData, использованного в DistributedEnclaveSeal, когда набор данных запечатывался.

[0168] В некоторых вариантах осуществления, идентификаторы анклавов, авторизованных, чтобы распечатывать данные, могут быть указаны (например, в параметре SetOfTargetEnclaves) общедоступными ключами целевых авторизованных целенаправленно выбранных анклавов. В этом варианте осуществления, аттестация анклава места назначения для DSE может не требоваться, но распечатанные данные могут затем предоставляться только как зашифрованные с использованием одного из указанных общедоступных ключей. Полагая, что только целенаправленно выбранные анклавы имеют доступ к соответствующим конфиденциальным ключам для дешифрования, только целенаправленно выбранные анклавы будут иметь доступ к распечатанным данным.

[0169] В вариантах осуществления, не изображенных на фиг. 19, функции анклава 1932 распределенного запечатывания (DSE) могут сами быть распределены по нескольким DSE. Например, функциональность DSE может быть распределена по нескольким DSE на нескольких компьютерах для избыточности и устойчивости к отказам. В этом примере, любой дублированный DSE может обслуживать запрос запечатать или распечатать. Отметим, что запечатанные данные 1938, как только они запечатаны/зашифрованы, могут безопасно храниться в любом месте, включая

дублирование по облачным серверам хранения.

[0170] Распределенное запечатывание данных может разрешить перемещение рабочих нагрузок анклава между компьютерами. Например, данные исходного анклава, запечатанные посредством DSE, могут представлять собой данные состояния исходного анклава на первом облачном сервере, которые могут загружаться в анклав места назначения на втором облачном сервере после распечатывания. Это может делаться аналогично тому, как описано выше в отношении фиг. 18. Безопасная операция может начинать исполнение в исходном анклаве. Далее, возможно после того, как исполнение в исходном анклаве прервано, состояние исходного анклава может быть запечатано на DSE и затем распечатано для анклава места назначения, когда анклав места назначения готов продолжить безопасную операцию, которая была начата в исходном анклаве.

[0171] Фиг. 20 представляет собой примерную блок-схему последовательности операций для распределенного запечатывания и распечатывания данных, как может выполняться анклавом запечатывания или DSE. Блоки 2002-2006 соответствуют распределенному запечатыванию данных, в то время как блоки 2008-2010 соответствуют распределенному распечатыванию данных. В ответ на запрос запечатать набор данных анклава, исходящий из исходного анклава, анклав запечатывания (или DSE) может выполнить аттестацию самого себя для исходного анклава, путем отправки отчета или котировки аттестации на исходный анклав в блоке 2002. Исходный анклав может верифицировать идентификационную информацию анклава запечатывания как подлинного и доверенного анклава запечатывания, путем проверки значения и подписи идентификатора в отчете аттестации анклава запечатывания. В блоке 2004, анклав запечатывания принимает разрешенный список и данные анклава, подлежащие запечатыванию. Они могут быть приняты посредством безопасного канала, как описано выше в отношении фиг. 19. В опциональном блоке 2006, анклав запечатывания может запечатать данные исходного анклава в него самого, например, если данные хранятся вне безопасного контейнера анклава запечатывания, например, в файловой системе компьютера. Чтобы распечатать данные для анклава места назначения, анклав места назначения может выполнить аттестацию себя самого для анклава запечатывания, например, путем предоставления отчета или котировки аттестации, в блоке 2008. В блоке 2010, идентификационная информация анклава места назначения может быть верифицирована, например, путем проверки отчета аттестации анклава места назначения. В блоке 2012, анклав запечатывания определяет, разрешено ли анклаву места назначения распечатывать данные из исходного анклава, путем верифицирования того, что аутентифицированная идентификационная информация анклава места назначения включена в разрешенный список, принятый с данными. Как только разрешение было подтверждено, данные анклава могут быть распечатаны, если они были запечатаны, и затем отправлены на анклав места назначения посредством безопасного канала в блоке 2014.

Анклав хранилища ключей

[0172] Хранилища ключей могут быть реализованы с анклавами. Хранилище ключей безопасно хранит ключи, такие как ключи системы шифрования для шифрования и дешифрования данных, для клиентов хранилища ключей. Хранилище ключей может дополнительно выполнять операции с ключом, такие как шифрование и дешифрование данных, подписание данных и получение новых ключей из существующего ключа. Хранилище ключей, при реализации как анклав, может обеспечивать высоконадежное хранение и обработку с секретными ключами шифрования. Дополнительно, аттестация

программного обеспечения анклава хранилища ключей может обеспечивать высокие уровни уверенности клиенту хранилища, что он осуществляет связь с хранилищем аутентичных и доверенных ключей. Высоконадежные системы могут быть построены на анклаве хранилища ключей с хранящимся в хранилище (запертым) ключом, причем ключ, хранящийся внутри хранилища ключей, никогда не выпускается любому клиенту вне хранилища ключей, и в некоторых случаях запертый в хранилище ключ может только существовать как хранящийся внутри (или возможно запечатанный) анклава хранилища ключей.

[0173] Фиг. 21 представляет собой блок-схему примерного анклава хранилища ключей. Анклав 2122 представляет собой хранилище ключей внутри безопасного контейнера анклава второй нативной платформы 2126 анклава. В примере на фиг. 21, клиент 2112 анклава 2122 хранилища ключей также представляет собой анклав и хостируется внутри безопасного контейнера анклава первой нативной платформы 2116 анклава. Анклавы 2112, 2122 могут взаимодействовать с их соответственными нативными платформами 2116, 2126 посредством соответственных платформ 2114, 2124 абстракции анклава. В других вариантах осуществления, одна или обе платформы 2114, 2124 абстракции могут не существовать там, где анклавы 2112 и/или 2122 взаимодействуют напрямую с нативными платформами 2116, 2126.

[0174] Анклав 2122 хранилища ключей может осуществлять связь с клиентом 2112 хранилища посредством канала 2150 связи. В некоторых вариантах осуществления, канал 2112 связи может представлять собой безопасный канал связи, обеспечивающий уверенность в конфиденциальности, целостности и/или свежести сообщений, отправляемых по каналу 2150 связи. Конфиденциальность и целостность такого безопасного канала связи могут быть установлены, например, при помощи шифрования и подписей, как на фиг. 2 и 3, с применением совместно используемых ключей, генерируемых как часть процесса аттестации, как на фиг. 5 и 6.

[0175] Аттестация программного обеспечения обеспечивает безопасность отчасти путем обеспечения уверенности в подлинности субъекта на другой стороне канала связи. Путем аттестации анклава 2122 хранилища ключей для клиента хранилища, клиент может получить уверенность в том, что анклав 2122 хранилища ключей представляет собой того, за кого себя выдает, до отправки секрета, такого как ключ или другие данные открытого текста, на хранилище ключей. Обратное также истинно для клиентов, которые также представляют собой анклавы, как изображено на фиг. 21. Путем аттестации анклава 2112 хранилища для анклава 2122 хранилища ключей, хранилище ключей может получить уверенность в том, что клиент представляет собой того, за кого себя выдает, до раскрытия секрета, такого как ключ или другие данные открытого текста, клиенту.

[0176] Системы хранилища ключей с хранящимися в защищенном хранилище ключами и полученными (выведенными) ключами, в частности, где ключи шифрования выводятся из запертого в хранилище ключа, могут использоваться, чтобы создавать систему безопасности, которая является гибкой и очень надежной. Функция вывода ключей, которые могут или не могут быть открытыми, может использоваться, чтобы генерировать множество ключей из первого ключа. Первый ключ (корневой секрет) может быть запертым в хранилище для наивысшего уровня безопасности, и ключи, выведенные из первого ключа, могут использоваться в целях шифрования. Если выведенный ключ скомпрометирован, новый выведенный ключ может генерироваться в существующей системе без необходимости доступа к хранилищу ключей или изменения хранилища ключей, хранящего первый ключ.

[0177] Примерный анклав хранилища ключей (KVE) представляет собой облачную систему хранилища ключей, которая обеспечивает хранение, генерацию, получение, распределение, шифрование, дешифрование и подписи ключей с использованием анклавов. KVE может быть идентифицирован своим точным хешем (хешем содержимого его безопасного контейнера) или произвольным идентификатором, назначенным его создателем или ассоциированным с ним. В последнем случае, анклав может быть подписан закрытым ключом его создателя во избежание столкновений и нарушений безопасности из-за конфликтов идентификатора.

[0178] Клиент хранилища ключей может взаимодействовать с системой хранилища ключей с использованием следующих примерных примитивов. Примерный прототип функции StoreKey представляет собой:

StoreKey([in] Keyname, [in] KeyType, [in] KeyValue, [in] Policy)

StoreKey хранит данный ключ в хранилище ключей и ассоциирует его с данным именем. Тип ключа также ассоциирован с ключом и ограничивает то, что может делаться с ключом. Например, некоторые ключи должны использоваться только для шифрования, другие для подписей и т.д. И конкретные ключи могут использоваться только с конкретными криптографическими алгоритмами. Политика может задавать политики, чтобы дополнительно ограничивать использование ключа. Например, она может задавать набор идентификаторов анклава, которым разрешено извлекать ключ и/или использовать ключ. Она может также задавать временные свойства, например, что ключ должен быть разрушен в определенную дату, или что скорость операций с использованием ключа должна быть ограничена.

[0179] Примерный прототип функции GenerateKey представляет собой:

GenerateKey([in] keyName, [in] keyType, [in] Policy)

[0180] GenerateKey генерирует новый ключ определенного типа и сохраняет его внутри хранилища ключей, т.е. ключ никогда не покидает хранилище ключей.

[0181] Примерный прототип функции GetKey представляет собой:

GetKey([in] KeyName, [out] KeyValue)

[0182] GetKey извлекает ключ, хранящийся внутри хранилища ключей. Эти примитивы обычно реализуются на безопасном канале связи, и код, который вызывает примитив, обычно исполняется в доверенной среде. В таком контексте, часто допустимо извлекать ключ из хранилища ключей.

[0183] Примерный прототип функции DeleteKey представляет собой:

DeleteKey([in] keyName)

[0184] DeleteKey удаляет ключ из хранилища ключей.

[0185] Примерный прототип функции DeriveKey представляет собой:

DeriveKey([in] newKeyName, [in] KeyName, [in] kdfIdentifier, [in] AdditionalData)

[0186] DeriveKey использует криптографическую функцию получения (вывода) ключа (KDF), идентифицированную посредством kdfIdentifier, чтобы вывести новый ключ на основе ключа, идентифицированного посредством keyName, и данных, пересланных в AdditionalData.

[0187] Примерный прототип функции Encrypt представляет собой:

Encrypt([in] KeyName, [in] algorithm, [in] data, [out]encryptedData)

[0188] Encrypt зашифровывает данные при помощи ключа, идентифицированного посредством keyName, с использованием запрошенного алгоритма.

[0189] Примерный прототип функции Decrypt представляет собой:

Decrypt([in] KeyName, [in] algorithm, [in] encrytedData, [out]data)

[0190] Decrypt дешифрует данные при помощи ключа, идентифицированного

посредством keyName, с использованием запрошенного алгоритма.

[0191] Примерный прототип функции Sign представляет собой:

Sign([in] KeyName, [in] algorithm, [in] data, [out] signature)

[0192] Sign подписывает данные при помощи ключа, идентифицированного посредством keyName, с использованием запрошенного алгоритма.

[0193] Примерный прототип функции VerifySignature представляет собой:

VerifySignature([in]KeyName, [in] algorithm, [in] signature, [out} bool signatureIsCorrect)

[0194] VerifySignature верифицирует подпись при помощи ключа, идентифицированного посредством keyName, с использованием запрошенного алгоритма.

[0195] Все из названных выше примитивов хранилища ключей могут быть реализованы путем установления безопасного канала с KVE. Канал может быть установлен с использованием аттестации и выполнения обмена ключами Диффи-Хеллмана, как описано выше в отношении фиг. 5 и 6. После того, как канал связи установлен, запрос безопасно отправляется по каналу, и ответ считывается из канала. Канал может обеспечивать гарантии конфиденциальности и целостности обмениваемых данных.

[0196] В другом варианте осуществления, когда KVE работает впервые, он генерирует пару общедоступного/конфиденциального ключей, и он генерирует котировку для общедоступного ключа. Затем он выписывает котировку и общедоступный ключ, в то же время сохраняя конфиденциальный ключ внутри анклава. Общедоступный ключ и котировка могут затем распределяться по всем системам/кодам, которые желают использовать хранилище ключей. В этом случае, реализация вышеуказанных примитивов верифицирует котировку, чтобы удостовериться, что она ведет диалог с подлинным KVE, и затем дешифрует запросы с использованием общедоступного ключа KVE. В качестве части запроса, реализация примитивов может включать в себя ключ для шифрования и защиты целостности результатов, отправленных из KVE. Этот вариант осуществления может обеспечивать безопасный канал двусторонней связи без аттестации.

[0197] Фиг. 22 представляет собой примерную блок-схему последовательности операций для некоторых операций хранилища ключей анклава. Процесс 2200 начинается в блоке 2202 путем безопасного хранения, в анклаве хранилища ключей, ключа, используемого в системе шифрования. Ключ может использоваться, например, чтобы зашифровывать или дешифровывать данные, генерировать криптографическую подпись, или может только использоваться как корневой ключ, из которого следует извлекать другие ключи. Ключ может безопасно храниться в анклаве хранилища ключей путем, например, хранения ключа в пространстве памяти безопасного контейнера анклава. В других вариантах осуществления, ключ может храниться безопасно вне безопасного контейнера анклава путем запечатывания данных ключа в анклав хранилища ключей или может храниться безопасно путем удаленного запечатывания при помощи анклава распределенного запечатывания, как описано в отношении фиг. 19 и 20.

[0198] В блоке 2204, анклав хранилища ключей выполняет процесс аттестации для аттестации идентификационной информации анклава хранилища ключей для клиента хранилища. Это может давать клиенту уверенность, что хранилище ключей не является самозванцем и ему можно доверять секреты, такие как ключ или данные, подлежащие шифрованию. Аттестация анклава хранилища ключей может включать в себя отправку, на клиент хранилища, отчета аттестации или котировки аттестации анклава хранилища ключей. Клиент хранилища ключей может затем верифицировать целостность отчета

аттестации путем верифицирования подписи в отчете аттестации при помощи общедоступного ключа, ассоциированного с нативной платформой анклава анклава хранилища ключей. Например, отчет аттестации хранилища 2122 ключей может быть сгенерирован второй нативной платформой 2126, и клиент 2112 хранилища может верифицировать подпись в отчете с использованием общедоступного ключа, ассоциированного со второй нативной платформой 2126. Этот процесс аттестации может также генерировать ключи, используемые для безопасного канала связи между клиентом хранилища и анклавом хранилища ключей, например, как показано на фиг. 5 и 6. Отчет аттестации может включать в себя идентификационную информацию анклава хранилища ключей, которая может определяться различными способами, как описано выше, например, в отношении фиг. 14 и 15. Идентификационная информация может, например, быть основана на хеше всего содержимого безопасного контейнера анклава хранилища ключей, хеше только уникального идентификатора, назначенного автором/создателем анклава хранилища ключей, или хеше комбинации части содержимого контейнера и уникального идентификатора.

[0199] Некоторые операции хранилища ключей анклава могут также требовать уверенности в подлинности клиента хранилища. Например, дешифрование данных или разглашение ключа (например, при помощи примитивов Decrypt или GetKey) может требовать такой уверенности. В этих ситуациях, если клиент хранилища также представляет собой анклав, дополнительный блок 2208 включает в себя процесс аттестации для верифицирования, анклавом хранилища ключей, идентификационной информации клиента хранилища. Процесс аттестации блока 2208 может включать в себя прием, в анклаве хранилища ключей, отчета или котировки аттестации клиента хранилища.

[0200] В дополнительном блоке 2210, безопасный канал связи может быть установлен между хранилищем ключей и анклавом хранилища ключей. Безопасная связь может требоваться для пересылки секретов между клиентом хранилища и анклавом хранилища ключей, таких как ключи или данные, подлежащие шифрованию. Процесс аттестации блока 2004 или 2008 может генерировать ключи, которые могут использоваться, чтобы создавать безопасный канал связи между клиентом хранилища и анклавом хранилища ключей, например, как показано на фиг. 5 и 6. С другой стороны, любой известный общедоступный ключ места назначения сообщения может использоваться, чтобы безопасно отправлять сообщение.

[0201] В блоке 2212, операция ключа, такая как операция одного из примитивов хранилища ключей, описанных выше, может выполняться внутри анклава хранилища ключей. Во время этой операции, данные ключа могут храниться только в адресном пространстве безопасного контейнера анклава хранилища ключей. Примерные примитивы включают в себя DeriveKey, Decrypt, Sign и другие.

[0202] Процесс 2200 предполагает, что анклав хранилища ключей уже знает ключ. Отметим, что для некоторых операций или примитивов анклава хранилища ключей, таких как StoreKey или GenerateKey, порядок операций может отличаться от изображенного в процессе 2200. Например, для GenerateKey, операция генерации ключа (как в блоке 2212) будет происходить до операции безопасного хранения блока 2202. Такой порядок операций изображен на фиг. 23, блоки 2302-2308.

[0203] Фиг. 23 представляет собой примерную блок-схему последовательности операций для создания и использования анклава хранилища ключей с запертым в хранилище ключом. В блоках 2302-2308 процесса 2300, новый ключ выводится в анклаве хранилища ключей. В блоках 2310-2316, заново выведенный ключ используется, чтобы

выполнять операцию дешифрования. Это представляет собой примерное использование запертого в хранилище ключа, причем все операции ключа выполняются при помощи анклава хранилища ключей, и ключ никогда не предоставляются клиенту хранилища. Дополнительно, новый ключ в этом примере никогда не может существовать вне анклава хранилища ключей, поскольку он был создан (выведен) из самого анклава хранилища ключей, и никогда не предоставлялся анклаву хранилища ключей из клиента хранилища или откуда-либо еще. Для некоторых вариантов осуществления и политик использования ключа, запертый в хранилище ключ может быть недолговечным в том, что он никогда не покидает безопасный контейнер анклава хранилища ключей, даже после запечатывания ключа в анклав хранилища ключей. Подобный недолговечный ключ, такой как может иметь место с выведенным ключом, используемым, чтобы временно защищать канал связи, может переставать существовать где-либо, когда контейнер анклава хранилища ключей разрушен или завершен. Хотя процесс на фиг. 23 иллюстрирует то, как может использоваться запертый в хранилище ключ, процесс на фиг. 23 может также использоваться с ключом, который не заперт в хранилище, например, если политика использования ключа разрешала возвратить ключ на клиент, который запросил его создание.

[0204] В блоке 2302, анклав хранилища ключей выполняет аттестацию себя для клиента хранилища. Это может потребоваться клиенту, поскольку клиент будет предоставлять секрет, подлежащий шифрованию, в блоке 2312. В блоке 2304, анклав хранилища ключей может принимать, например, от клиента хранилища, указание политики использования ключа. Указание может, например, представлять собой структуру данных, задающую политику, или может представлять собой идентификатор, подлежащий использованию с регистром политик использования ключа. Политика использования ключа сама может указывать, что этот ключ никогда не должен быть предоставлен какому-либо клиенту хранилища. В блоке 2306, новый ключ выводится из ранее известного корневого ключа, например, при помощи примитива DeriveKey, описанного выше. Запрос (не изображен) вывести новый ключ может быть принят анклавом хранилища ключей, например, от клиента хранилища. В блоке 2308, заново выведенный ключ может храниться безопасно в соответствии с принятой политикой использования ключа.

[0205] Клиент хранилища может выполнить аттестацию себя по отношению к анклаву хранилища ключей в блоке 2310. Процесс аттестации может включать в себя прием, в анклаве хранилища ключей, отчета или котировки аттестации клиента хранилища. Принятая политика использования ключа может ограничивать некоторые или все использования нового ключа до запросов от запросчиков, которые аутентифицированы посредством аттестации программного обеспечения. В блоках 2312-2316, операция дешифрования, такая как для примитива Decrypt, описанного выше, выполняется с использованием ключа, выведенного в блоке 2306. В других вариантах осуществления, другие операции могут выполняться при помощи запертого в хранилище ключа, такие как шифрование, подписание, верифицирование подписи и вывод другого нового ключа из ключа, выведенного в блоке 2306 (вывод ключа второго поколения из корневого ключа). В блоке 2312, буфер зашифрованных данных принимается от клиента хранилища. Принятые зашифрованные данные дешифруются при помощи выведенного ключа в блоке 1314, и результирующие дешифрованные данные (в буфере дешифрованных данных) отправляются на клиент хранилища посредством безопасного канала связи в блоке 2316.

[0206] В варианте осуществления, способ идентификации анклава выполняется

вычислительным устройством, содержащим процессор и память, способ содержит:

[0207] прием типа идентификационной информации и запроса на операцию, связанную с реализованным анклавом;

[0208] определение значения идентификационной информации для анклава на основе типа идентификационной информации и информации, полученной из образа анклава, из которого был реализован анклав; и

[0209] выполнение операции с значением идентификационной информации.

[0210] В варианте осуществления, способ дополнительно содержит:

[0211] верифицирование целостности значения идентификационной информации путем верифицирования подписи в образе анклава с общедоступным ключом, связанным с автором образа анклава.

[0212] В варианте осуществления, значение идентификационной информации определяется как вывод хеш-функции, и (способ) дополнительно содержит:

[0213] определение, основываясь, по меньшей мере частично, на типе идентификационной информации, идентификационной части образа анклава; и

[0214] вычисление хеш-функции по идентификационной части.

[0215] В варианте осуществления, образ анклава включает в себя ссылки на дополнительные зависимые образы анклава, и дополнительно содержит:

[0216] определение, основываясь, по меньшей мере частично, на типе идентификационной информации, какие из дополнительных зависимых образов анклава включены как ввод в хеш-функцию.

[0217] В варианте осуществления, идентификационная часть включает в себя одно или несколько из: двоичного кода, скопированного в безопасный контейнер анклава во время реализации реализуемого анклава, и одного или нескольких идентификаторов, которые не являются исполняемым кодом.

[0218] В варианте осуществления:

[0219] Значение идентификационной информации определяется как вывод хеш-функции;

[0220] анклав был реализован с множеством образов анклава; и

[0221] тип идентификационной информации указывает, какие из множества образов анклава включены, по меньшей мере частично, в хеш-функцию, чтобы определять значение идентификационной информации.

[0222] В варианте осуществления:

[0223] операция включает в себя генерирование отчета аттестации с значением идентификационной информации.

[0224] В варианте осуществления:

[0225] операция включает в себя запечатывание данных в анклав путем запечатывания данных с значением идентификационной информации.

[0226] В варианте осуществления, операция включает в себя реализацию монотонного счетчика, причем монотонный счетчик идентифицируется значением идентификационной информации.

[0227] В варианте осуществления, операция включает в себя осуществление измерения доверенного времени, причем доверенное время определяется на основе значения идентификационной информации.

[0228] В варианте осуществления, система содержит по меньшей мере процессор и память, хранящую инструкции, которые, при исполнении системой, вызывают по меньшей мере:

[0229] прием типа идентификационной информации и запроса на операцию, связанную

с реализованным анклавом;

[0230] определение значения идентификационной информации для анклава на основе типа идентификационной информации и информации, полученной из образа анклава, из которого был реализован анклав; и

[0231] выполнение операции с значением идентификационной информации.

[0232] В варианте осуществления, инструкции дополнительно вызывают по меньшей мере:

[0233] верифицирование целостности значения идентификационной информации путем верифицирования подписи в образе анклава с общедоступным ключом, связанным с автором образа анклава.

[0234] В варианте осуществления, значение идентификационной информации определяется как вывод хеш-функции, и причем инструкции дополнительно вызывают по меньшей мере:

[0235] определение, основываясь, по меньшей мере частично, на типе идентификационной информации, идентификационной части образа анклава; и

[0236] вычисление хеш-функции по идентификационной части.

[0237] В варианте осуществления, образ анклава включает в себя ссылки на дополнительные зависимые образы анклава, причем инструкции дополнительно вызывают по меньшей мере:

[0238] определение, основываясь, по меньшей мере частично, на типе идентификационной информации, какие из дополнительных зависимых образов анклава включены во ввод в хеш-функцию.

[0239] В варианте осуществления, идентификационная часть включает в себя одно или несколько из: двоичного кода, скопированного в безопасный контейнер анклава во время реализации реализуемого анклава, и одного или нескольких идентификаторов, которые не являются исполняемым кодом.

[0240] В варианте осуществления, значение идентификационной информации определяется как вывод хеш-функции;

[0241] анклав был реализован с множеством образов анклава; и

[0242] тип идентификационной информации указывает, какие из множества образов анклава включены, по меньшей мере частично, в хеш-функцию, чтобы определить значение идентификационной информации.

[0243] В варианте осуществления, операция включает в себя генерирование отчета аттестации с значением идентификационной информации.

[0244] В варианте осуществления, операция включает в себя запечатывание данных в анклав путем запечатывания данных с значением идентификационной информации.

[0245] В варианте осуществления, операция включает в себя реализацию монотонного счетчика, причем монотонный счетчик идентифицирован значением идентификационной информации.

[0246] В варианте осуществления, операция включает в себя осуществление измерения доверенного времени, причем доверенное время определяется на основе значения идентификационной информации.

[0247] В варианте осуществления, способ идентификации анклава содержит:

[0248] прием типа идентификационной информации и запроса на операцию, связанную с реализованным анклавом;

[0249] определение значения идентификационной информации для анклава на основе типа идентификационной информации и данных, хранящихся в контейнере анклава; и

[0250] выполнение операции с значением идентификационной информации.

[0251] Способ идентификации анклава по п. 21, причем:

[0252] данные, хранящиеся в контейнере анклава, включают в себя множество значений идентификационной информации, и

[0253] значение идентификационной информации определяется путем выбора среди множества значений идентификационной информации на основе типа идентификационной информации.

[0254] Каждый из процессов, способов и алгоритмов, описанных в предыдущих разделах, может быть воплощен и полностью или частично автоматизирован посредством программных модулей, исполняемых одним или несколькими компьютерами или компьютерными процессорами. Модули кода может быть сохранены на любом типе не-временного считываемого компьютерного носителя или компьютерного устройства хранения, такого как накопители на жестких дисках, твердотельная память, оптический диск и/или тому подобные. Процессы и алгоритмы могут быть реализованы частично или полностью в ориентированных на приложение схемах. Результаты описанных процессов и этапов процесса могут быть сохранены, постоянно или иным образом, в любом типе не-временного компьютерного хранилища, такого как, например, энергозависимое или энергонезависимое хранилище. Различные признаки и процессы, описанные выше, могут быть использованы независимо друг от друга или могут быть объединены различным образом. Все возможные комбинации и подкомбинации предназначены, чтобы входить в объем настоящего раскрытия. Кроме того, некоторые способы или блоки процессов могут быть опущены в некоторых реализациях. Способы и процессы, описанные в данном документе, также не ограничены какой-либо определенной последовательностью, а блоки или состояния, относящиеся к ним, могут быть выполнены в других последовательностях, которые являются подходящими. Например, описанные блоки или состояния могут быть выполнены в порядке, ином, чем тот, который конкретно раскрыт, или множество блоков или состояний могут быть объединены в одном блоке или состоянии. Примерные блоки или состояния могут выполняться последовательно, параллельно или каким-либо другим способом. Блоки или состояния могут быть добавлены или удалены из раскрытых примерных вариантов осуществления. Примерные системы и компоненты, описанные в данном документе, могут быть сконфигурированы по-другому, чем описано. Например, элементы могут быть добавлены, удалены или переупорядочены по сравнению с раскрытыми примерными вариантами осуществления.

[0255] Следует также иметь в виду, что различные элементы показаны как хранящиеся в памяти или хранилище во время использования, и что эти элементы или их части могут переноситься между памятью и другими устройствами хранения для целей управления памятью и целостности данных. Альтернативно, в других вариантах осуществления, некоторые или все из программных модулей и/или систем могут исполняться в памяти на другом устройстве и взаимодействовать с проиллюстрированными вычислительными системами через межкомпьютерную коммуникацию. Кроме того, в некоторых вариантах осуществления, некоторые или все из систем и/или модулей могут быть реализованы или предоставлены другими способами, например, по меньшей мере частично во встроенном программном обеспечении и/или аппаратных средствах, включая, но без ограничения указанным, одну или несколько специализированных интегральных схем (ASIC), стандартные интегральные схемы, контроллеры (например, путем исполнения соответствующих инструкций и включая микроконтроллеры и/или встроенные контроллеры), программируемые вентильные матрицы (FPGA), сложные программируемые логические

устройства (CPLD) и т.д. Некоторые или все из модулей, систем и структур данных также могут быть сохранены (например, как программные инструкции или структурированные данные) на считываемом компьютером носителе, таком как жесткий диск, память, сеть или портативный медиа-продукт для чтения с помощью соответствующего привода или через соответствующее соединение. Для целей данного описания и формулы изобретения, фраза "считываемый компьютером носитель хранения" и ее производные не включают в себя волны, сигналы и/или другие временные (переходные) и/или неосязаемые коммуникационные среды. Системы, модули и структуры данных также могут быть переданы как генерируемые сигналы данных (например, как часть несущей волны или другой аналоговый или цифровой распространяющийся сигнал) на множестве считываемых компьютером сред передачи, включая беспроводные и проводные/кабельные среды, и могут принимать различные формы (например, как часть одиночного или мультиплексированного аналогового сигнала или как множество дискретных цифровых пакетов или кадров). Такие компьютерные программные продукты могут также принимать другие формы в других вариантах осуществления. Соответственно, настоящее раскрытие может быть осуществлено на практике с другими конфигурациями компьютерных систем.

[0256] Условная терминология, используемая в данном описании, в частности, такие термины как "быть способным", "был бы способен", "мог бы", "может", "например" и т.п., если специально не указано иное или понимаемое иначе в используемом контексте, как правило, предназначены для передачи того, что некоторые варианты осуществления включают в себя, в то время как другие варианты осуществления не включают в себя определенные признаки, элементы и/или этапы. Таким образом, такие условные термины, как правило, не подразумевает, что признаки, элементы и/или этапы в любом случае требуются для одного или более вариантов осуществления, или что один или более вариантов осуществления обязательно включают в себя логику для принятия решения, при наличии или без авторского ввода или подсказки, включены ли эти признаки, элементы и/или этапы или должны выполняться в любом конкретном варианте осуществления. Термины "содержащий", "включающий", "имеющий" и тому подобные являются синонимами и используются инклюзивно, открытым образом и не исключают дополнительные элементы, признаки, действия, операции и так далее. Кроме того, термин "или" используется в инклюзивном смысле (а не в исключительном смысле), так что когда он используется, например, для соединения списка элементов, термин "или" означает один, некоторые или все элементы в списке.

[0257] Хотя были описаны некоторые примерные варианты осуществления, эти варианты осуществления были представлены только в качестве примера и не предназначены для ограничения объема раскрытых здесь изобретений. Таким образом, ничто в приведенном выше описании не подразумевает, что любой конкретный признак, характеристика, этап, модуль или блок является необходимым или незаменимым. В самом деле, новые способы и системы, описанные здесь, могут быть воплощены в различных других формах; кроме того, различные пропуски, замены и изменения в форме способов и систем, описанных здесь, могут быть сделаны без отклонения от сущности раскрытых здесь изобретений. Прилагаемая формула изобретения и ее эквиваленты предназначены для охвата таких форм или модификаций как входящих в пределы объема и сущности некоторых из раскрытых здесь изобретений.

(57) Формула изобретения

1. Способ идентификации анклава, выполняемый вычислительным устройством,

содержащим процессор и память, причем способ содержит:

прием типа идентификационной информации и запроса на операцию, связанную с анклавом, экземпляр которого создан, при этом образ анклава, из которого был создан экземпляр анклава, включает в себя ссылки на дополнительные зависимые образы анклава;

определение того, какие из этих дополнительных образов анклава должны быть включены, по меньшей мере отчасти, в качестве ввода в хеш-функцию, на основе, по меньшей мере отчасти, того, что каждый дополнительный образ анклава, который должен быть включен, по меньшей мере отчасти, в качестве ввода в хеш-функцию, включается, по меньшей мере отчасти, в идентификационную часть образа анклава, которая основывается, по меньшей мере отчасти, на типе идентификационной информации;

вычисление хеш-функции на основе, по меньшей мере отчасти, типа идентификационной информации и информации, извлеченной из образа анклава, чтобы обеспечить, в качестве вывода хеш-функции, значение идентификационной информации для анклава, причем целостность значения идентификационной информации является верифицируемой путем верифицирования подписи в образе анклава с помощью общедоступного ключа, связанного с автором образа анклава; и

выполнение упомянутой операции с этим значением идентификационной информации, чтобы обеспечить безопасность в отношении анклава.

2. Способ по п.1, дополнительно содержащий этап, на котором определяют, основываясь, по меньшей мере отчасти, на типе идентификационной информации, идентификационную часть образа анклава, при этом при упомянутом вычислении хеш-функции хеш-функцию вычисляют по этой идентификационной части.

3. Способ по п.2, в котором упомянутая идентификационная часть включает в себя одно или более из: (a) двоичного кода, скопированного в безопасный контейнер анклава во время создания экземпляра анклава, и (b) одного или более идентификаторов, которые не являются исполняемым кодом.

4. Способ по п.1, в котором экземпляр анклава был создан с помощью множества образов анклава и тип идентификационной информации указывает, какие из этого множества образов анклава включаются, по меньшей мере частично, в хеш-функцию, чтобы определять значение идентификационной информации.

5. Способ по п.1, в котором упомянутая операция включает в себя генерирование отчета аттестации с упомянутым значением идентификационной информации.

6. Способ по п.1, в котором упомянутая операция включает в себя запечатывание данных в анклав путем запечатывания данных с упомянутым значением идентификационной информации.

7. Способ по п.1, в котором упомянутая операция включает в себя приращение монотонного счетчика, причем монотонный счетчик идентифицируется упомянутым значением идентификационной информации.

8. Способ по п.1, в котором упомянутая операция включает в себя осуществление измерения доверенного времени, причем доверенное время определяется на основе упомянутого значения идентификационной информации.

9. Способ по п.1, в котором образ анклава включает в себя множество идентификаторов, соответствующих множеству соответственных типов абстрактной идентификационной информации, причем каждый из множества идентификаторов идентифицирует соответственную группу экземпляров анклава, соответствующих этому соответственному типу абстрактной идентификационной информации.

10. Вычислительная система, сконфигурированная для идентификации анклава, причем вычислительная система содержит по меньшей мере процессор и память, хранящую инструкции, которые при их исполнении системой предписывают выполнение по меньшей мере:

приема типа идентификационной информации и запроса на операцию, связанную с анклавом, экземпляр которого создан, при этом тип идентификационной информации используется для идентификации множества анклавов, имеющих по меньшей мере одно общее свойство, каковое множество анклавов включает в себя анклав, экземпляр которого создан;

вычисления хеш-функции на основе, по меньшей мере отчасти, типа идентификационной информации и информации, извлеченной из образа анклава, из которого был создан экземпляр анклава, чтобы обеспечить, в качестве вывода хеш-функции, значение идентификационной информации для анклава, при этом образ анклава включает в себя множество ссылок на множество соответственных дополнительных зависимых образов анклава, причем целостность значения идентификационной информации является верифицируемой путем верифицирования подписи в образе анклава с помощью общедоступного ключа, связанного с автором образа анклава, при этом тип идентификационной информации указывает то, какие из этих дополнительных образов анклава включаются в качестве ввода в хеш-функцию; и

выполнения упомянутой операции с этим значением идентификационной информации, чтобы обеспечить безопасность в отношении анклава.

11. Система по п.10, в которой инструкции предписывают выполнение по меньшей мере:

определения, основываясь, по меньшей мере отчасти, на типе идентификационной информации, идентификационной части образа анклава; и

вычисления хеш-функции по этой идентификационной части.

12. Система по п.10, при этом упомянутая идентификационная часть включает в себя одно или более из: (a) двоичного кода, скопированного в безопасный контейнер анклава во время создания экземпляра анклава, и (b) одного или более идентификаторов, которые не являются исполняемым кодом.

13. Система по п.10, при этом экземпляр анклава был создан с помощью множества образов анклава и тип идентификационной информации указывает, какие из этого множества образов анклава включаются, по меньшей мере частично, в хеш-функцию, чтобы определять значение идентификационной информации.

14. Система по п.10, в которой упомянутая операция включает в себя генерирование отчета аттестации с упомянутым значением идентификационной информации.

15. Система по п.10, в которой упомянутая операция включает в себя запечатывание данных в анклав путем запечатывания данных с упомянутым значением идентификационной информации.

16. Система по п.10, в которой упомянутая операция включает в себя приращение монотонного счетчика, причем монотонный счетчик идентифицируется упомянутым значением идентификационной информации.

17. Система по п.10, в которой упомянутая операция включает в себя осуществление измерения доверенного времени, причем доверенное время определяется на основе упомянутого значения идентификационной информации.

18. Способ идентификации анклава, содержащий:

прием типа идентификационной информации и запроса на операцию, связанную с

анклавом, экземпляр которого создан, при этом тип идентификационной информации используется для идентификации множества анклавов, имеющих по меньшей мере одно общее свойство, каковое множество анклавов включает в себя анклав, экземпляр которого создан;

вычисление хеш-функции на основе, по меньшей мере отчасти, типа идентификационной информации и данных, хранящихся в контейнере анклава, чтобы обеспечить, в качестве вывода хеш-функции, значение идентификационной информации для анклава, при этом тип идентификационной информации указывает то, какие из множества образов анклава, которые были использованы для создания экземпляра анклава, должны быть включены, по меньшей мере отчасти, в хеш-функцию, причем данные, хранящиеся в контейнере анклава, включают в себя множество значений идентификационной информации, которые идентифицируют анклав, при этом упомянутое значение идентификационной информации определяют посредством выбора среди этого множества значений идентификационной информации на основе типа идентификационной информации, причем целостность значения идентификационной информации является верифицируемой путем верифицирования подписи в образе анклава с помощью общедоступного ключа, связанного с автором образа анклава; и

выполнение упомянутой операции с этим значением идентификационной информации, чтобы обеспечить безопасность в отношении анклава.

19. Способ по п.18, в котором упомянутая операция включает в себя запечатывание данных в анклав путем запечатывания данных с упомянутым значением идентификационной информации.

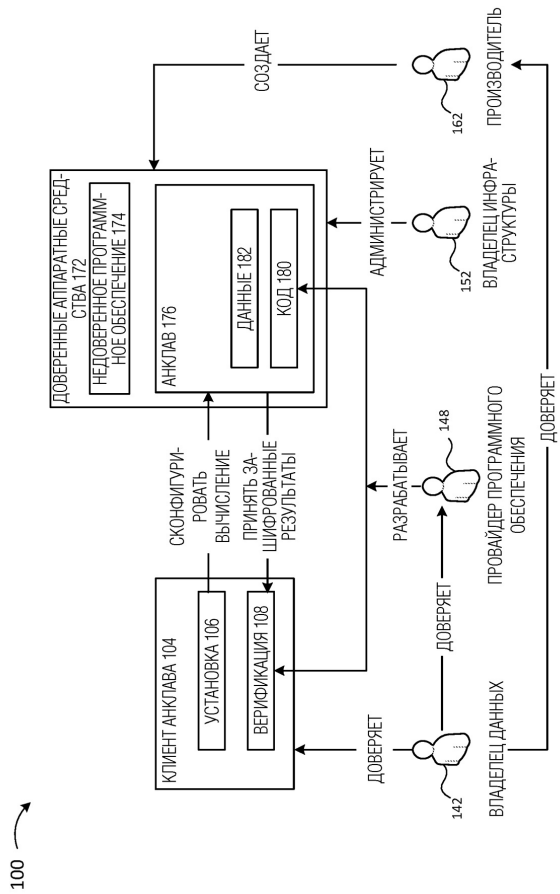
20. Способ по п.18, в котором упомянутая операция включает в себя приращение монотонного счетчика, причем монотонный счетчик идентифицируется упомянутым значением идентификационной информации.

21. Способ по п.18, в котором упомянутая операция включает в себя осуществление измерения доверенного времени, причем доверенное время определяется на основе упомянутого значения идентификационной информации.

22. Способ по п.18, в котором упомянутая операция включает в себя генерирование отчета аттестации с упомянутым значением идентификационной информации.

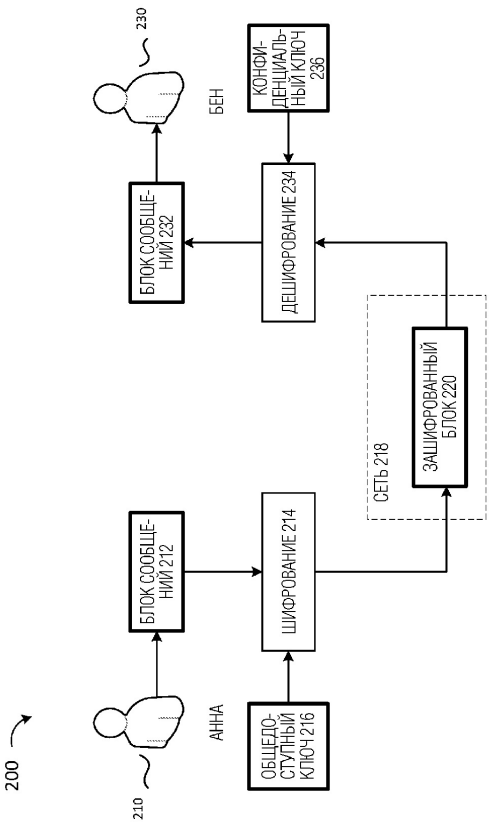
1

1/23


100
КЛИЕНТ АНКЛАВА 104
УСТАНОВКА 106
ВЕРИФИКАЦИЯ 108
ДОВЕРЯЕТ
ВЛАДЕЛЕЦ ДАННЫХ
142
СКОНФИГУРИ-РОВАТЬ ВЫЧИСЛЕНИЕ
ПРИНЯТЬ ЗА-ШИФРОВАННЫЕ РЕЗУЛЬТАТЫ
ДОВЕРЕННЫЕ АППАРАТНЫЕ СРЕД-СТВА 172
НЕДОВЕРЕННОЕ ПРОГРАММ-НОЕ ОБЕСПЕЧЕНИЕ 174
АНКЛАВ 176
ДАННЫЕ 182
КОД 180
РАЗРАБАТЫВАЕТ
148
ПРОВАЙДЕР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ДОВЕРЯЕТ
АДМИНИСТРИРУЕТ
152
ВЛАДЕЛЕЦ ИНФРА-СТРУКТУРЫ
СОЗДАЕТ
162
ПРОИЗВОДИТЕЛЬ

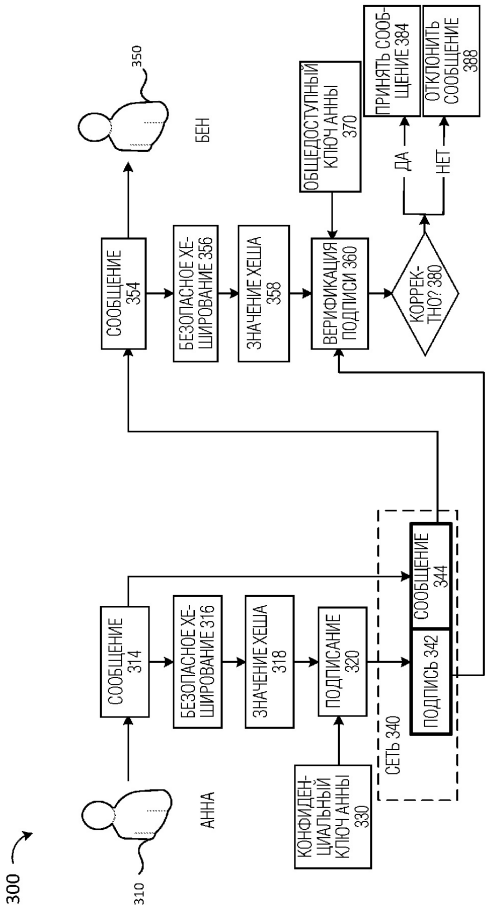
ФИГ. 1

2


200
210
АННА
ОБЩЕДОСТУПНЫЙ КЛЮЧ 216
БЛОК СООБЩЕНИЙ 212
ШИФРОВАНИЕ 214
СЕТЬ 218
ЗАШИФРОВАННЫЙ БЛОК 220
ДЕШИФРОВАНИЕ 234
КОНФИДЕНЦИАЛЬНЫЙ КЛЮЧ 236
БЛОК СООБЩЕНИЙ 232
БЕН
230

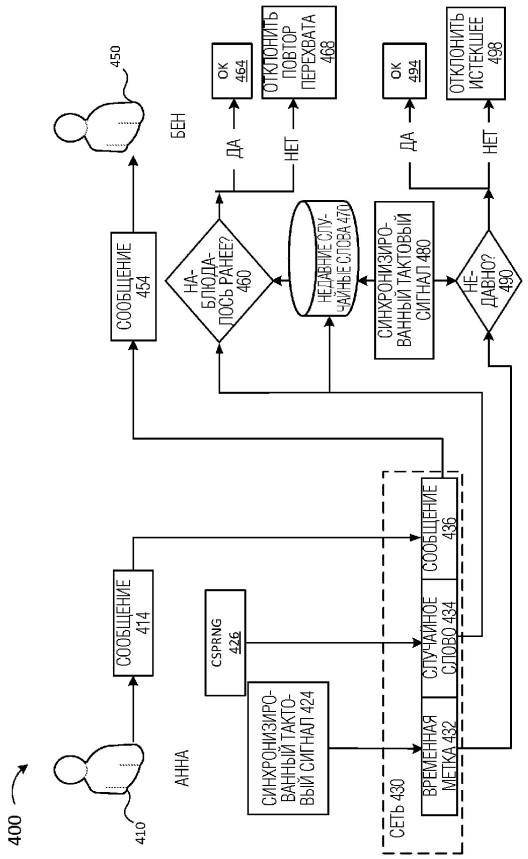
ФИГ. 2

3


300
310
АННА
СООБЩЕНИЕ 314
БЕЗОПАСНОЕ ХЕШИРОВАНИЕ 316
ЗНАЧЕНИЕ ХЕША 318
ПОДПИСАНИЕ 320
КОНФИДЕНЦИАЛЬНЫЙ КЛЮЧ АННЫ 330
СЕТЬ 340
ПОДПИСЬ 342
СООБЩЕНИЕ 344
350
БЕН
СООБЩЕНИЕ 354
БЕЗОПАСНОЕ ХЕШИРОВАНИЕ 356
ЗНАЧЕНИЕ ХЕША 358
ВЕРИФИКАЦИЯ ПОДПИСИ 360
ОБЩЕДОСТУПНЫЙ КЛЮЧ АННЫ 370
КОРРЕКТНО? 380
ДА
НЕТ
ПРИНЯТЬ СООБЩЕНИЕ 384
ОТКЛОНИТЬ СООБЩЕНИЕ 388

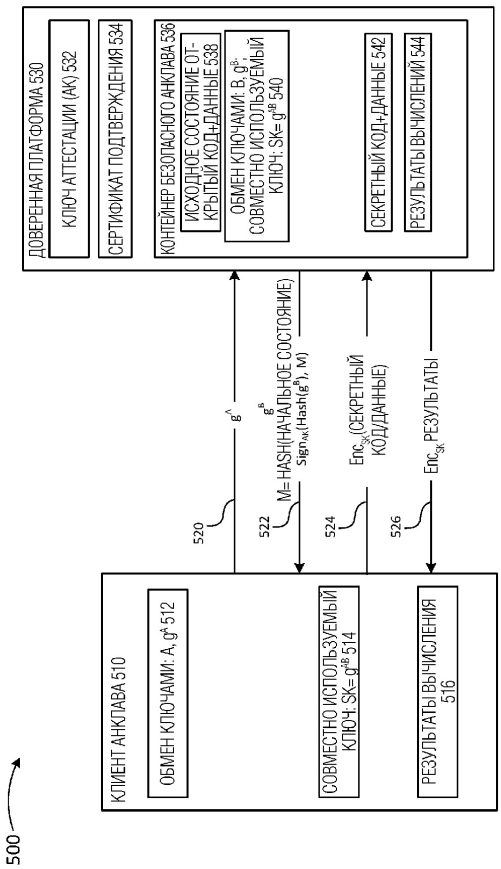
ФИГ. 3

4


400
410
АННА
СООБЩЕНИЕ 414
СИНХРОНИЗИРОВАННЫЙ ТАКТОВЫЙ СИГНАЛ 424
CSPRNG 426
СЕТЬ 430
ВРЕМЕННАЯ МЕТКА 432
СЛУЧАЙНОЕ СЛОВО 434
СООБЩЕНИЕ 436
СООБЩЕНИЕ 454
450
БЕН
НАБЛЮДАЛОСЬ РАНЕЕ? 460
ДА
НЕТ
ОК 464
ОТКЛОНИТЬ ПОВТОР ПЕРЕХВАТА 468
НЕДАВНИЕ СЛУЧАЙНЫЕ СЛОВА 470
СИНХРОНИЗИРОВАННЫЙ ТАКТОВЫЙ СИГНАЛ 480
НЕДАВНО? 490
ДА
НЕТ
ОК 494
ОТКЛОНИТЬ ИСТЕКШЕЕ 498

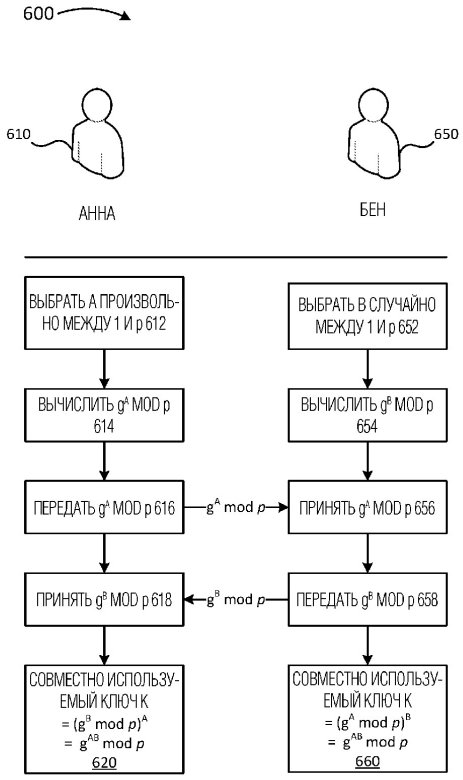
ФИГ. 4

5


ДОВЕРЕННАЯ ПЛАТФОРМА 530
КЛЮЧ АТТЕСТАЦИИ (АК) 532
СЕРТИФИКАТ ПОДТВЕРЖДЕНИЯ 534
КОНТЕЙНЕР БЕЗОПАСНОГО АНКЛАВА 536
ИСХОДНОЕ СОСТОЯНИЕ ОТКРЫТЫЙ КОД+ДАННЫЕ 538
ОБМЕН КЛЮЧАМИ: B, g^b; СОВМЕСТНО ИСПОЛЬЗУЕМЫЙ КЛЮЧ: $SK=g^{ab}$ 540
СЕКРЕТНЫЙ КОД+ДАННЫЕ 542
РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ 544
КЛИЕНТ АНКЛАВА 510
ОБМЕН КЛЮЧАМИ: A, g^a 512
СОВМЕСТНО ИСПОЛЬЗУЕМЫЙ КЛЮЧ: $SK=g^{ab}$ 514
РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЯ 516
520
522
524
526
g^a
g^b
M= HASH(НАЧАЛЬНОЕ СОСТОЯНИЕ)
$Sign_{AK}(Hash(g^b), M)$
Enc_{SK}(СЕКРЕТНЫЙ КОД/ДАННЫЕ)
Enc_{SK} РЕЗУЛЬТАТЫ
500

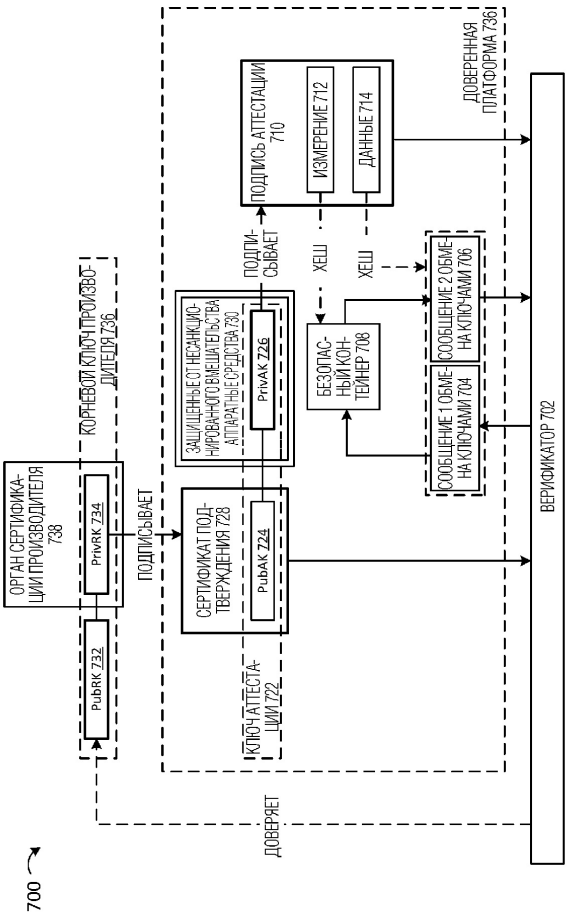
ФИГ. 5

6

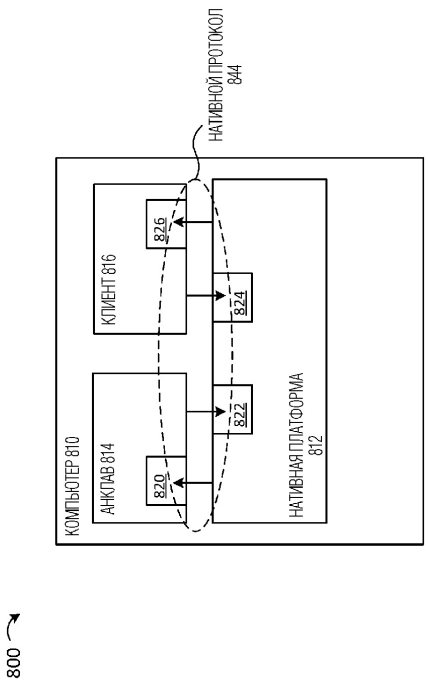

600
610
650
АННА
БЕН
ВЫБРАТЬ A ПРОИЗВОЛЬНО МЕЖДУ 1 И p 612
ВЫБРАТЬ B СЛУЧАЙНО МЕЖДУ 1 И p 652
ВЫЧИСЛИТЬ g^A MOD p 614
ВЫЧИСЛИТЬ g^B MOD p 654
ПЕРЕДАТЬ g^A MOD p 616
g^A mod p
ПРИНЯТЬ g^A MOD p 656
ПРИНЯТЬ g^B MOD p 618
g^B mod p
ПЕРЕДАТЬ g^B MOD p 658
СОВМЕСТНО ИСПОЛЬЗУЕМЫЙ КЛЮЧ K = $(g^B \bmod p)^A$ = $g^{AB} \bmod p$ 620
СОВМЕСТНО ИСПОЛЬЗУЕМЫЙ КЛЮЧ K = $(g^A \bmod p)^B$ = $g^{AB} \bmod p$ 660

ФИГ. 6

7

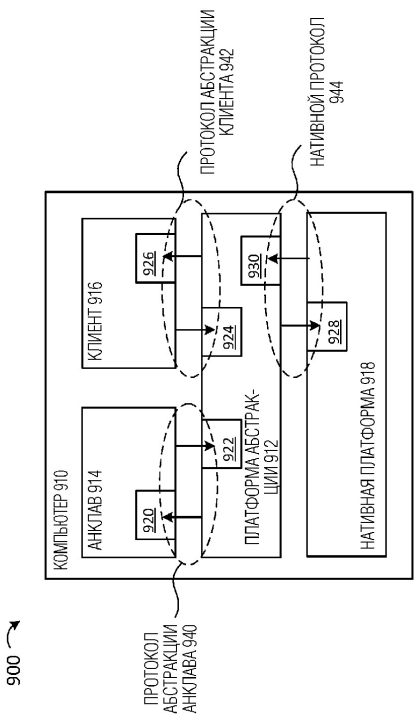

700
ОРГАН СЕРТИФИКАЦИИ ПРОИЗВОДИТЕЛЯ 738
PrivRK 734
PubRK 732
КОРНЕВОЙ КЛЮЧ ПРОИЗВОДИТЕЛЯ 736
ПОДПИСЫВАЕТ
СЕРТИФИКАТ ПОДТВЕРЖДЕНИЯ 728
PubAK 724
КЛЮЧ АТТЕСТАЦИИ 722
ЗАЩИЩЕННЫЕ ОТ НЕСАНКЦИОНИРОВАННОГО ВМЕШАТЕЛЬСТВА АППАРАТНЫЕ СРЕДСТВА 730
PrivAK 726
ПОДПИСЫВАЕТ
ПОДПИСЬ АТТЕСТАЦИИ 710
ИЗМЕРЕНИЕ 712
ДАННЫЕ 714
ДОВЕРЕННАЯ ПЛАТФОРМА 736
ХЕШ
ХЕШ
БЕЗОПАСНЫЙ КОНТЕЙНЕР 708
СООБЩЕНИЕ 1 ОБМЕНА КЛЮЧАМИ 704
СООБЩЕНИЕ 2 ОБМЕНА КЛЮЧАМИ 706
ВЕРИФИКАТОР 702
ДОВЕРЯЕТ

ФИГ. 7

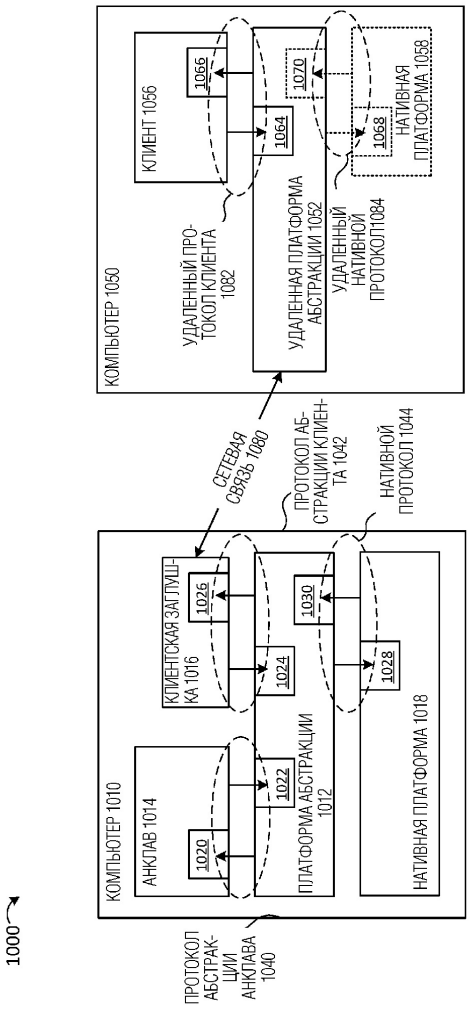

800
КОМПЬЮТЕР 810
АНКЛАВ 814
КЛИЕНТ 816
НАТИВНАЯ ПЛАТФОРМА 812
НАТИВНОЙ ПРОТОКОЛ 844
820
822
824
826

ФИГ. 8

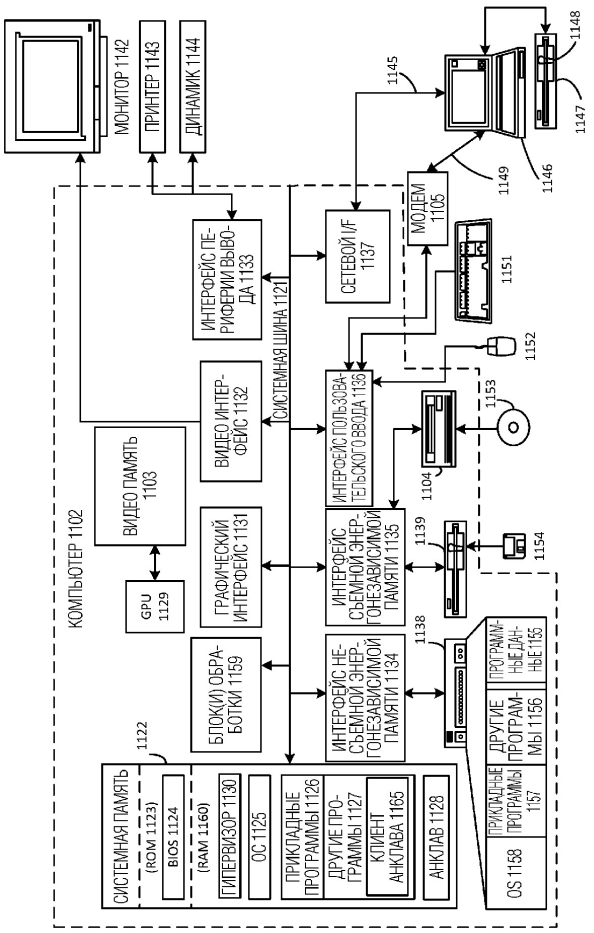
9


900
КОМПЬЮТЕР 910
АНКПАВ 914
920
КЛИЕНТ 916
926
922
924
ПЛАТФОРМА АБСТРАКЦИИ 912
930
928
НАТИВНАЯ ПЛАТФОРМА 918
ПРОТОКОЛ АБСТРАКЦИИ АНКПАВА 940
ПРОТОКОЛ АБСТРАКЦИИ КЛИЕНТА 942
НАТИВНОЙ ПРОТОКОЛ 944

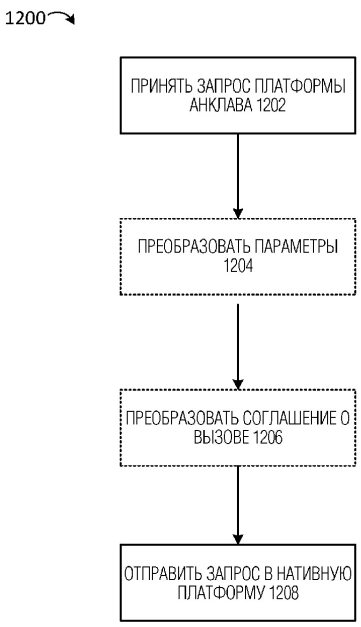
ФИГ. 9


1000
КОМПЬЮТЕР 1010
АНКЛАВ 1014
1020
КЛИЕНТСКАЯ ЗАГЛУШКА 1016
1026
1024
ПЛАТФОРМА АБСТРАКЦИИ 1012
1022
1030
НАТИВНАЯ ПЛАТФОРМА 1018
1028
ПРОТОКОЛ АБСТРАКЦИИ АНКЛАВА 1040
ПРОТОКОЛ АБСТРАКЦИИ КЛИЕНТА 1042
НАТИВНОЙ ПРОТОКОЛ 1044
СЕТЕВАЯ СВЯЗЬ 1080
КОМПЬЮТЕР 1050
КЛИЕНТ 1056
1066
1064
УДАЛЕННЫЙ ПРОТОКОЛ КЛИЕНТА 1082
УДАЛЕННАЯ ПЛАТФОРМА АБСТРАКЦИИ 1052
1070
1068
НАТИВНАЯ ПЛАТФОРМА 1058
УДАЛЕННЫЙ НАТИВНОЙ ПРОТОКОЛ 1084

ФИГ. 10

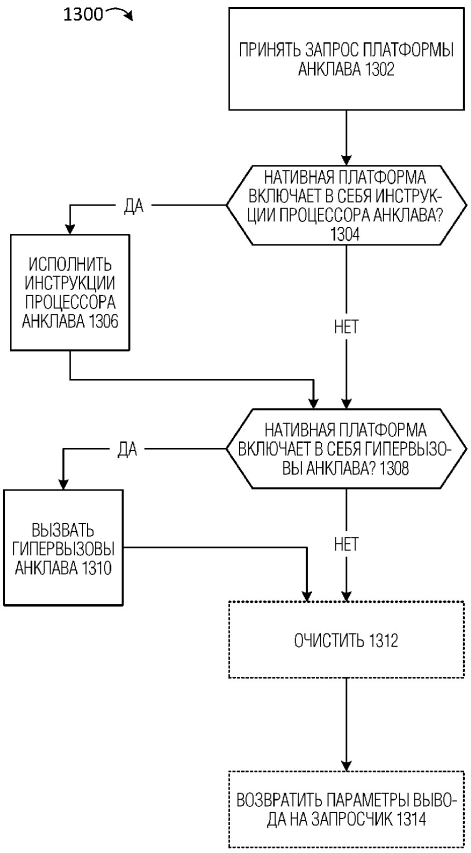

МОНИТОР 1142
ПРИНТЕР 1143
ДИНАМИК 1144
1145
1148
1147
1149
1146
МОДЕМ 1105
СЕТЕВОЙ I/F 1137
ИНТЕРФЕЙС ПЕРИФЕРИЙ ВЫВОДА 1133
1151
1152
1153
1104
1139
1154
1138
ВИДЕО ИНТЕРФЕЙС 1132
ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА 1135
СИСТЕМНАЯ ШИНА 1121
КОМПЬЮТЕР 1102
ВИДЕО ПАМЯТЬ 1103
GPU 1129
ГРАФИЧЕСКИЙ ИНТЕРФЕЙС 1131
ИНТЕРФЕЙС СЪЕМНОЙ ЭНЕРГОНЕЗАВИСИМОЙ ПАМЯТИ 1135
БЛОК(И) ОБРАБОТКИ 1159
ИНТЕРФЕЙС НЕСЪЕМНОЙ ЭНЕРГОНЕЗАВИСИМОЙ ПАМЯТИ 1134
1122
СИСТЕМНАЯ ПАМЯТЬ
(ROM 1123)
BIOS 1124
(RAM 1160)
ГИПЕРВИЗОР 1130
ОС 1125
ПРИКЛАДНЫЕ ПРОГРАММЫ 1126
ДРУГИЕ ПРОГРАММЫ 1127
КЛИЕНТ АНКЛАВА 1165
АНКЛАВ 1128
OS 1158
ПРИКЛАДНЫЕ ПРОГРАММЫ 1157
ДРУГИЕ ПРОГРАММЫ 1156
ПРОГРАММНЫЕ ДАННЫЕ 1155

ФИГ. 11

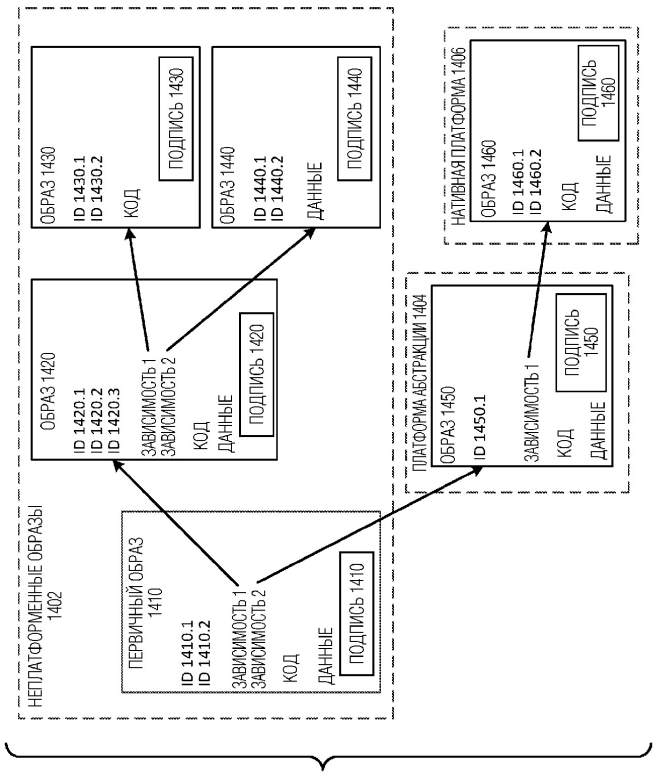

1200
ПРИНЯТЬ ЗАПРОС ПЛАТФОРМЫ АНКЛАВА 1202
ПРЕОБРАЗОВАТЬ ПАРАМЕТРЫ 1204
ПРЕОБРАЗОВАТЬ СОГЛАШЕНИЕ О ВЫЗОВЕ 1206
ОТПРАВИТЬ ЗАПРОС В НАТИВНУЮ ПЛАТФОРМУ 1208

ФИГ. 12

13

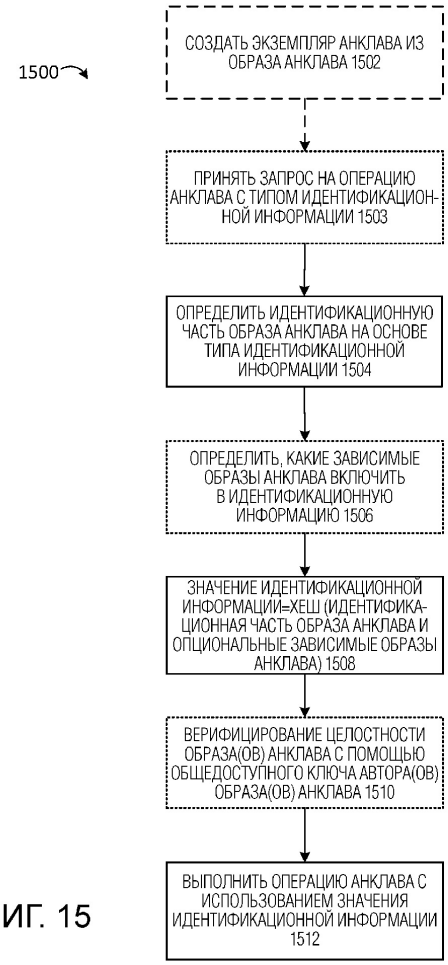

1300
ПРИНЯТЬ ЗАПРОС ПЛАТФОРМЫ АНКЛАВА 1302
НАТИВНАЯ ПЛАТФОРМА ВКЛЮЧАЕТ В СЕБЯ ИНСТРУКЦИИ ПРОЦЕССОРА АНКЛАВА? 1304
ДА
НЕТ
ИСПОЛНИТЬ ИНСТРУКЦИИ ПРОЦЕССОРА АНКЛАВА 1306
НАТИВНАЯ ПЛАТФОРМА ВКЛЮЧАЕТ В СЕБЯ ГИПЕРВЫЗОВЫ АНКЛАВА? 1308
ДА
НЕТ
ВЫЗВАТЬ ГИПЕРВЫЗОВЫ АНКЛАВА 1310
ОЧИСТИТЬ 1312
ВОЗВРАТИТЬ ПАРАМЕТРЫ ВЫВОДА НА ЗАПРОСЧИК 1314

ФИГ. 13


НЕПЛАТФОРМЕННЫЕ ОБРАЗЫ
1402
ПЕРВИЧНЫЙ ОБРАЗ
1410
ID 1410.1
ID 1410.2
ЗАВИСИМОСТЬ 1
ЗАВИСИМОСТЬ 2
КОД
ДАННЫЕ
ПОДПИСЬ 1410
ОБРАЗ 1420
ID 1420.1
ID 1420.2
ID 1420.3
ЗАВИСИМОСТЬ 1
ЗАВИСИМОСТЬ 2
КОД
ДАННЫЕ
ПОДПИСЬ 1420
ОБРАЗ 1430
ID 1430.1
ID 1430.2
КОД
ПОДПИСЬ 1430
ОБРАЗ 1440
ID 1440.1
ID 1440.2
ДАННЫЕ
ПОДПИСЬ 1440
ПЛАТФОРМА АБСТРАКЦИИ 1404
ОБРАЗ 1450
ID 1450.1
ЗАВИСИМОСТЬ 1
КОД
ДАННЫЕ
ПОДПИСЬ 1450
НАТИВНАЯ ПЛАТФОРМА 1406
ОБРАЗ 1460
ID 1460.1
ID 1460.2
КОД
ДАННЫЕ
ПОДПИСЬ 1460
КОНТЕЙНЕР АНКЛАВА
1490
НЕПЛАТФОРМЕННЫЕ ОБРАЗЫ
• ОБРАЗ 1410
• ОБРАЗ 1420
• ОБРАЗ 1430
• ОБРАЗ 1440
ОБРАЗЫ АБСТРАКТНОЙ ПЛАТФОРМЫ
• ОБРАЗ 1450
ОБРАЗЫ НАТИВНОЙ ПЛАТФОРМЫ
• ОБРАЗ 1460

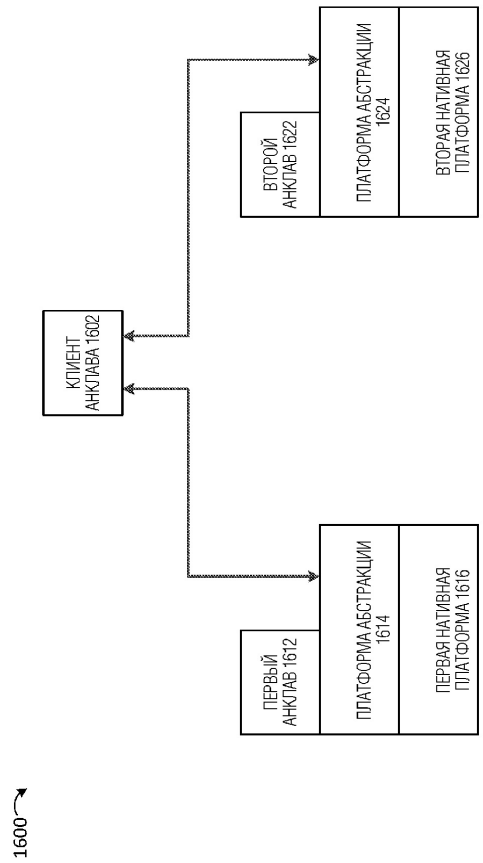
ФИГ. 14

15

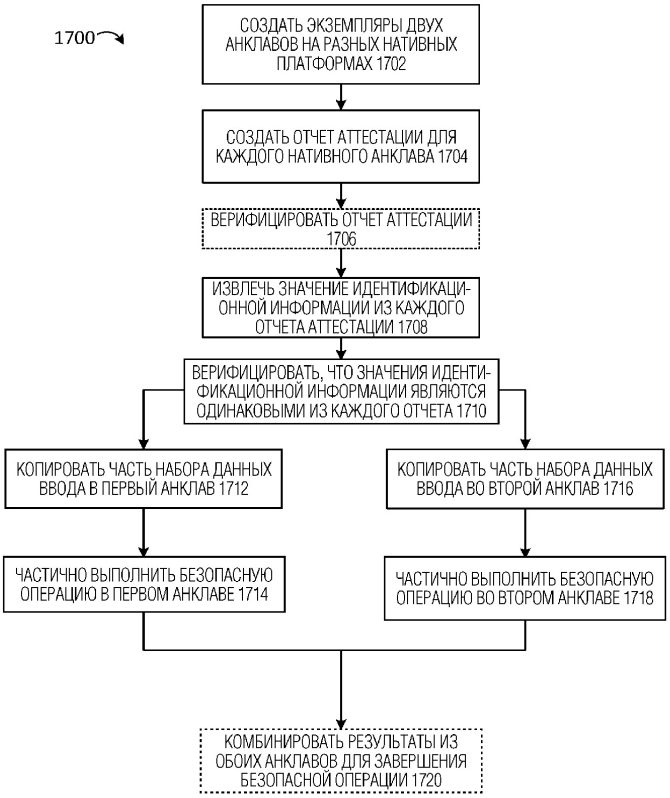

1500
СОЗДАТЬ ЭКЗЕМПЛЯР АНКЛАВА ИЗ ОБРАЗА АНКЛАВА 1502
ПРИНЯТЬ ЗАПРОС НА ОПЕРАЦИЮ АНКЛАВА С ТИПОМ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ 1503
ОПРЕДЕЛИТЬ ИДЕНТИФИКАЦИОННУЮ ЧАСТЬ ОБРАЗА АНКЛАВА НА ОСНОВЕ ТИПА ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ 1504
ОПРЕДЕЛИТЬ, КАКИЕ ЗАВИСИМЫЕ ОБРАЗЫ АНКЛАВА ВКЛЮЧИТЬ В ИДЕНТИФИКАЦИОННУЮ ИНФОРМАЦИЮ 1506
ЗНАЧЕНИЕ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ=ХЕШ (ИДЕНТИФИКАЦИОННАЯ ЧАСТЬ ОБРАЗА АНКЛАВА И ОПЦИОНАЛЬНЫЕ ЗАВИСИМЫЕ ОБРАЗЫ АНКЛАВА) 1508
ВЕРИФИЦИРОВАНИЕ ЦЕЛОСТНОСТИ ОБРАЗА(ОВ) АНКЛАВА С ПОМОЩЬЮ ОБЩЕДОСТУПНОГО КЛЮЧА АВТОРА(ОВ) ОБРАЗА(ОВ) АНКЛАВА 1510
ВЫПОЛНИТЬ ОПЕРАЦИЮ АНКЛАВА С ИСПОЛЬЗОВАНИЕМ ЗНАЧЕНИЯ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ 1512

ФИГ. 15

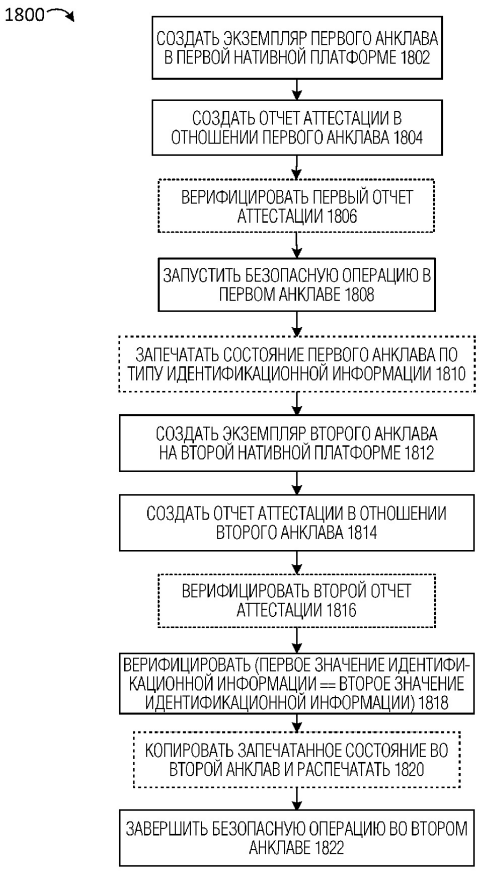
16


КЛИЕНТ АНКЛАВА 1602
ПЕРВЫЙ АНКЛАВ 1612
ПЛАТФОРМА АБСТРАКЦИИ 1614
ПЕРВАЯ НАТИВНАЯ ПЛАТФОРМА 1616
ВТОРОЙ АНКЛАВ 1622
ПЛАТФОРМА АБСТРАКЦИИ 1624
ВТОРАЯ НАТИВНАЯ ПЛАТФОРМА 1626
1600

ФИГ. 16

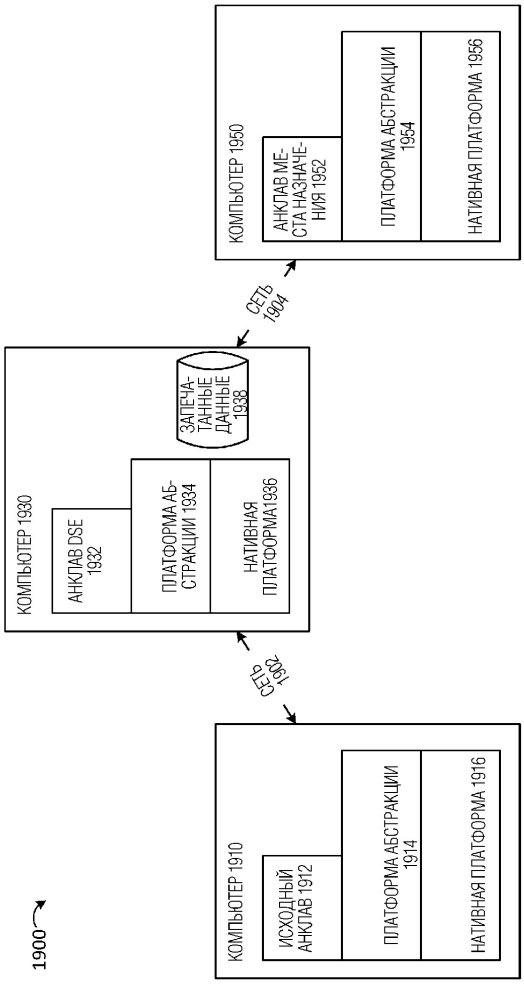

1700
СОЗДАТЬ ЭКЗЕМПЛЯРЫ ДВУХ АНКЛАВОВ НА РАЗНЫХ НАТИВНЫХ ПЛАТФОРМАХ 1702
СОЗДАТЬ ОТЧЕТ АТТЕСТАЦИИ ДЛЯ КАЖДОГО НАТИВНОГО АНКЛАВА 1704
ВЕРИФИЦИРОВАТЬ ОТЧЕТ АТТЕСТАЦИИ 1706
ИЗВЛЕЧЬ ЗНАЧЕНИЕ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ ИЗ КАЖДОГО ОТЧЕТА АТТЕСТАЦИИ 1708
ВЕРИФИЦИРОВАТЬ, ЧТО ЗНАЧЕНИЯ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ ЯВЛЯЮТСЯ ОДИНАКОВЫМИ ИЗ КАЖДОГО ОТЧЕТА 1710
КОПИРОВАТЬ ЧАСТЬ НАБОРА ДАННЫХ ВВОДА В ПЕРВЫЙ АНКЛАВ 1712
КОПИРОВАТЬ ЧАСТЬ НАБОРА ДАННЫХ ВВОДА ВО ВТОРОЙ АНКЛАВ 1716
ЧАСТИЧНО ВЫПОЛНИТЬ БЕЗОПАСНУЮ ОПЕРАЦИЮ В ПЕРВОМ АНКЛАВЕ 1714
ЧАСТИЧНО ВЫПОЛНИТЬ БЕЗОПАСНУЮ ОПЕРАЦИЮ ВО ВТОРОМ АНКЛАВЕ 1718
КОМБИНИРОВАТЬ РЕЗУЛЬТАТЫ ИЗ ОБОИХ АНКЛАВОВ ДЛЯ ЗАВЕРШЕНИЯ БЕЗОПАСНОЙ ОПЕРАЦИИ 1720

ФИГ. 17

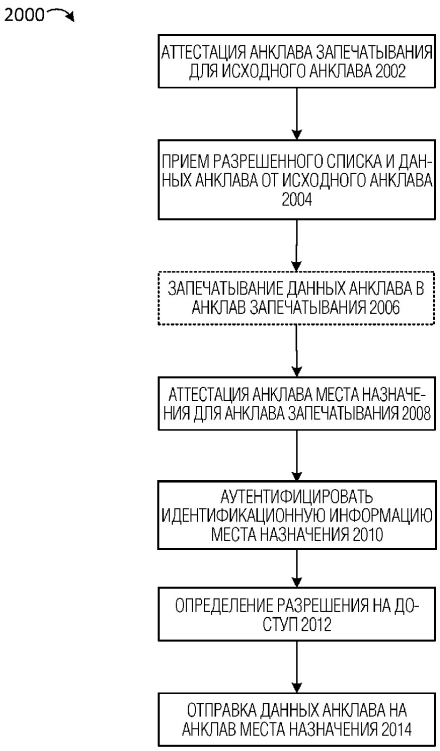

1800
СОЗДАТЬ ЭКЗЕМПЛЯР ПЕРВОГО АНКЛАВА В ПЕРВОЙ НАТИВНОЙ ПЛАТФОРМЕ 1802
СОЗДАТЬ ОТЧЕТ АТТЕСТАЦИИ В ОТНОШЕНИИ ПЕРВОГО АНКЛАВА 1804
ВЕРИФИЦИРОВАТЬ ПЕРВЫЙ ОТЧЕТ АТТЕСТАЦИИ 1806
ЗАПУСТИТЬ БЕЗОПАСНУЮ ОПЕРАЦИЮ В ПЕРВОМ АНКЛАВЕ 1808
ЗАПЕЧАТАТЬ СОСТОЯНИЕ ПЕРВОГО АНКЛАВА ПО ТИПУ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ 1810
СОЗДАТЬ ЭКЗЕМПЛЯР ВТОРОГО АНКЛАВА НА ВТОРОЙ НАТИВНОЙ ПЛАТФОРМЕ 1812
СОЗДАТЬ ОТЧЕТ АТТЕСТАЦИИ В ОТНОШЕНИИ ВТОРОГО АНКЛАВА 1814
ВЕРИФИЦИРОВАТЬ ВТОРОЙ ОТЧЕТ АТТЕСТАЦИИ 1816
ВЕРИФИЦИРОВАТЬ (ПЕРВОЕ ЗНАЧЕНИЕ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ == ВТОРОЕ ЗНАЧЕНИЕ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ) 1818
КОПИРОВАТЬ ЗАПЕЧАТАННОЕ СОСТОЯНИЕ ВО ВТОРОЙ АНКЛАВ И РАСПЕЧАТАТЬ 1820
ЗАВЕРШИТЬ БЕЗОПАСНУЮ ОПЕРАЦИЮ ВО ВТОРОМ АНКЛАВЕ 1822

ФИГ. 18

19

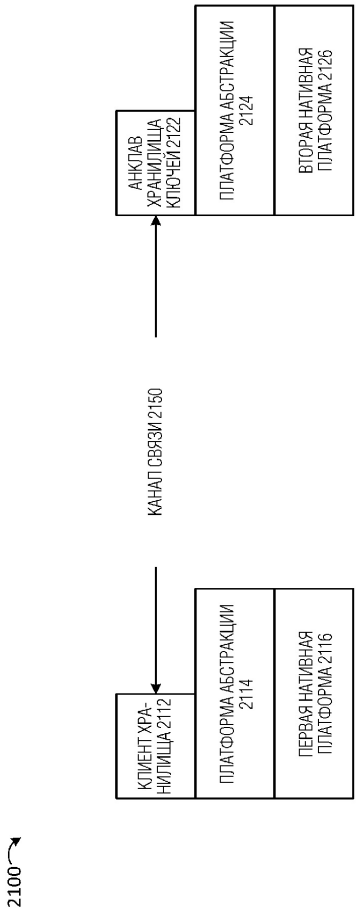

1900
КОМПЬЮТЕР 1910
ИСХОДНЫЙ АНКЛАВ 1912
ПЛАТФОРМА АБСТРАКЦИИ 1914
НАТИВНАЯ ПЛАТФОРМА 1916
СЕТЬ 1902
КОМПЬЮТЕР 1930
АНКЛАВ DSE 1932
ПЛАТФОРМА АБСТРАКЦИИ 1934
НАТИВНАЯ ПЛАТФОРМА1936
ЗАПЕЧАТАННЫЕ ДАННЫЕ 1938
СЕТЬ 1904
КОМПЬЮТЕР 1950
АНКЛАВ МЕСТА НАЗНАЧЕНИЯ 1952
ПЛАТФОРМА АБСТРАКЦИИ 1954
НАТИВНАЯ ПЛАТФОРМА 1956

ФИГ. 19

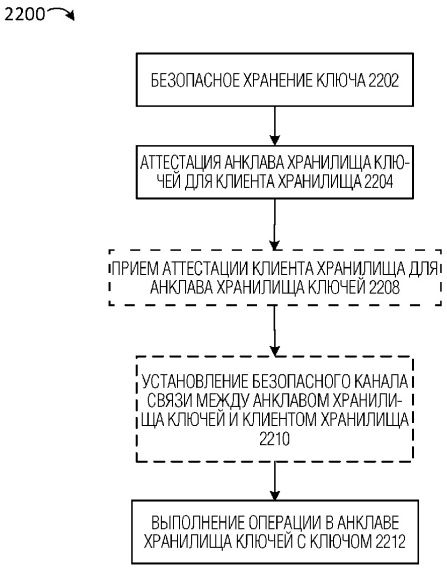

2000
АТТЕСТАЦИЯ АНКЛАВА ЗАПЕЧАТЫВАНИЯ ДЛЯ ИСХОДНОГО АНКЛАВА 2002
ПРИЕМ РАЗРЕШЕННОГО СПИСКА И ДАННЫХ АНКЛАВА ОТ ИСХОДНОГО АНКЛАВА 2004
ЗАПЕЧАТЫВАНИЕ ДАННЫХ АНКЛАВА В АНКЛАВ ЗАПЕЧАТЫВАНИЯ 2006
АТТЕСТАЦИЯ АНКЛАВА МЕСТА НАЗНАЧЕНИЯ ДЛЯ АНКЛАВА ЗАПЕЧАТЫВАНИЯ 2008
АУТЕНТИФИЦИРОВАТЬ ИДЕНТИФИКАЦИОННУЮ ИНФОРМАЦИЮ МЕСТА НАЗНАЧЕНИЯ 2010
ОПРЕДЕЛЕНИЕ РАЗРЕШЕНИЯ НА ДОСТУП 2012
ОТПРАВКА ДАННЫХ АНКЛАВА НА АНКЛАВ МЕСТА НАЗНАЧЕНИЯ 2014

ФИГ. 20

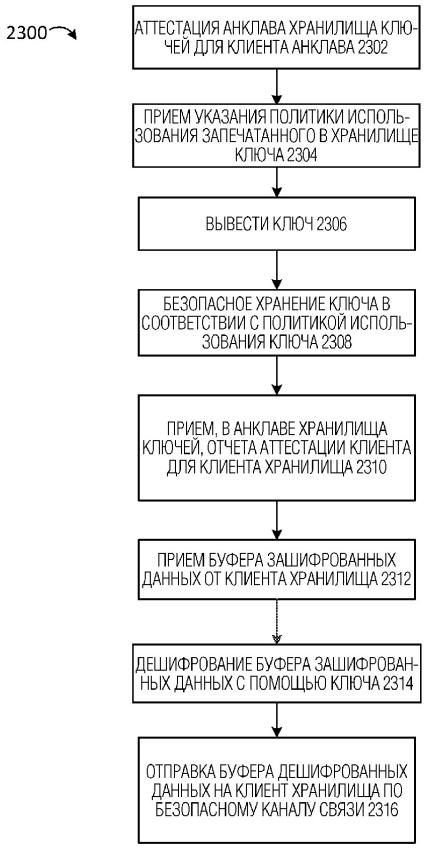
21


2100
КЛИЕНТ ХРАНИЛИЩА 2112
ПЛАТФОРМА АБСТРАКЦИИ 2114
ПЕРВАЯ НАТИВНАЯ ПЛАТФОРМА 2116
КАНАЛ СВЯЗИ 2150
АНКЛАВ ХРАНИЛИЩА КЛЮЧЕЙ 2122
ПЛАТФОРМА АБСТРАКЦИИ 2124
ВТОРАЯ НАТИВНАЯ ПЛАТФОРМА 2126

ФИГ. 21


2200
БЕЗОПАСНОЕ ХРАНЕНИЕ КЛЮЧА 2202
АТТЕСТАЦИЯ АНКЛАВА ХРАНИЛИЩА КЛЮЧЕЙ ДЛЯ КЛИЕНТА ХРАНИЛИЩА 2204
ПРИЕМ АТТЕСТАЦИИ КЛИЕНТА ХРАНИЛИЩА ДЛЯ АНКЛАВА ХРАНИЛИЩА КЛЮЧЕЙ 2208
УСТАНОВЛЕНИЕ БЕЗОПАСНОГО КАНАЛА СВЯЗИ МЕЖДУ АНКЛАВОМ ХРАНИЛИЩА КЛЮЧЕЙ И КЛИЕНТОМ ХРАНИЛИЩА 2210
ВЫПОЛНЕНИЕ ОПЕРАЦИИ В АНКЛАВЕ ХРАНИЛИЩА КЛЮЧЕЙ С КЛЮЧОМ 2212

ФИГ. 22


2300
АТТЕСТАЦИЯ АНКЛАВА ХРАНИЛИЩА КЛЮЧЕЙ ДЛЯ КЛИЕНТА АНКЛАВА 2302
ПРИЕМ УКАЗАНИЯ ПОЛИТИКИ ИСПОЛЬЗОВАНИЯ ЗАПЕЧАТАННОГО В ХРАНИЛИЩЕ КЛЮЧА 2304
ВЫВЕСТИ КЛЮЧ 2306
БЕЗОПАСНОЕ ХРАНЕНИЕ КЛЮЧА В СООТВЕТСТВИИ С ПОЛИТИКОЙ ИСПОЛЬЗОВАНИЯ КЛЮЧА 2308
ПРИЕМ, В АНКЛАВЕ ХРАНИЛИЩА КЛЮЧЕЙ, ОТЧЕТА АТТЕСТАЦИИ КЛИЕНТА ДЛЯ КЛИЕНТА ХРАНИЛИЩА 2310
ПРИЕМ БУФЕРА ЗАШИФРОВАННЫХ ДАННЫХ ОТ КЛИЕНТА ХРАНИЛИЩА 2312
ДЕШИФРОВАНИЕ БУФЕРА ЗАШИФРОВАННЫХ ДАННЫХ С ПОМОЩЬЮ КЛЮЧА 2314
ОТПРАВКА БУФЕРА ДЕШИФРОВАННЫХ ДАННЫХ НА КЛИЕНТ ХРАНИЛИЩА ПО БЕЗОПАСНОМУ КАНАЛУ СВЯЗИ 2316

ФИГ. 23