РОССИЙСКАЯ ФЕДЕРАЦИЯ



ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 595 924**(13) **C2**

(51) МПК
G09C 1/00 (2006.01)
H04L 9/32 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2014106490/08, 14.08.2012**

(24) Дата начала отсчета срока действия патента:
14.08.2012

Приоритет(ы):
(30) Конвенционный приоритет:
29.08.2011 JP 2011-185947;
02.03.2012 JP 2012-046687

(43) Дата публикации заявки: **27.08.2015** Бюл. № **24**

(45) Опубликовано: **27.08.2016** Бюл. № **24**

(56) Список документов, цитированных в отчете о поиске: **US 7961876 B2, 14.06.2011. US 2009/0010428 A1, 08.01.2009. US 2009/0279693 A1, 12.11.2009. WO 2010/024248 A1, 04.03.2010. US 2008/0013716 A1, 17.01.2008. US 2010/0266122 A1, 21.10.2010.**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **20.02.2014**

(86) Заявка PCT:
JP 2012/070675 (14.08.2012)

(87) Публикация заявки PCT:
WO 2013/031533 (07.03.2013)

Адрес для переписки:
109012, Москва, ул. Ильинка, 5/2, ООО "Союзпатент"

(72) Автор(ы):
САКУМОТО Койти (JP),
СИРАЙ Тайдзо (JP),
ХИВАТАРИ Харунага (JP),
КАМИО Кадзуя (JP)

(73) Патентообладатель(и):
СОНИ КОРПОРЕЙШН (JP)

RU 2595924 C2

RU 2595924 C2

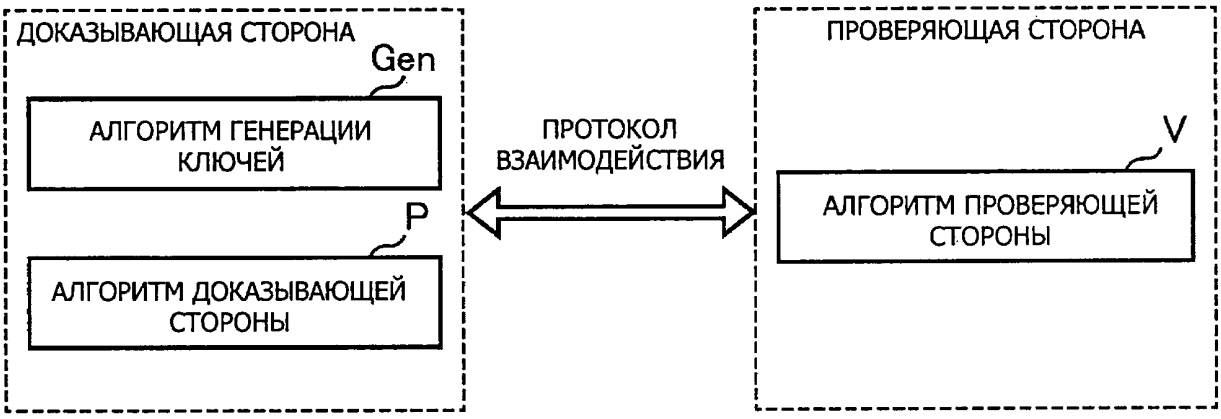
(54) УСТРОЙСТВО ОБРАБОТКИ ИНФОРМАЦИИ, СПОСОБ ОБРАБОТКИ ИНФОРМАЦИИ И ПРОГРАММА

(57) Реферат:

Изобретение относится к технологии обработки информации и технологии связи. Технический результат - обеспечение эффективной безопасности электронных документов. Устройство обработки информации, содержащее модуль генерации чисел, выполненный с возможностью генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков от нескольких переменных, и модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков от нескольких переменных,

составляющими элементами которых является указанная пара многочленов F множества порядков от нескольких переменных. 9 н. и 10 з.п. ф-лы, 28 ил.

АЛГОРИТМ АУТЕНТИФИКАЦИИ С ОТКРЫТЫМ КЛЮЧОМ



Фиг. 1

RU 2595924 C2

RU 2595924 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 595 924**(13) **C2**

(51) Int. Cl.
G09C 1/00 (2006.01)
H04L 9/32 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2014106490/08, 14.08.2012**

(24) Effective date for property rights:
14.08.2012

Priority:
(30) Convention priority:
29.08.2011 JP 2011-185947;
02.03.2012 JP 2012-046687

(43) Application published: **27.08.2015** Bull. № **24**

(45) Date of publication: **27.08.2016** Bull. № **24**

(85) Commencement of national phase: **20.02.2014**

(86) PCT application:
JP 2012/070675 (14.08.2012)

(87) PCT publication:
WO 2013/031533 (07.03.2013)

Mail address:
109012, Moskva, ul. Ilinka, 5/2, OOO "Sojuzpatent"

(72) Inventor(s):
SAKUMOTO Kojti (JP),
SIRAJ Tajdzo (JP),
KHIVATARI KHarunaga (JP),
KAMIO Kadzuya (JP)

(73) Proprietor(s):
SONI KORPOREJSHN (JP)

RU 2595924 C2

(54) INFORMATION PROCESSING APPARATUS, INFORMATION PROCESSING METHOD AND PROGRAM

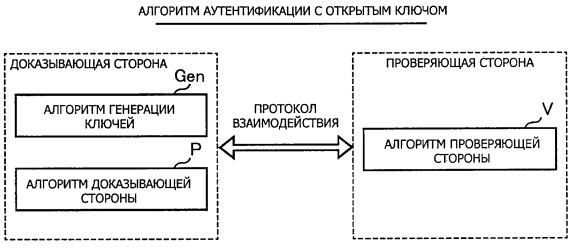
(57) Abstract:

FIELD: information technology.

SUBSTANCE: invention relates to information processing technology and communication technology. Information processing device, comprising generating module numbers, configured to generate numbers, used in coefficients of members of a pair of polynomials F = (f_1, …, f_m) of multiple orders from several variables, using preset function on basis of information shared by objects, performing algorithm of authentication with open key or algorithm of digital signature, which applies open key, containing above pair of polynomials F multiple orders from several variables and module purpose configured to assign numbers generated by module for generating numbers, coefficients of polynomials of multiple orders from several variables, components of which is above pair of polynomials F multiple orders from several variables.

EFFECT: technical result is provision of efficient security of electronic documents.

19 cl, 28 dwg



Фиг. 1

RU 2595924 C2

Область техники, к которой относится изобретение

Настоящее изобретение относится к устройству обработки информации, способу обработки информации и программе.

Уровень техники

По мере быстрого развития технологий обработки информации и технологий связи, происходит интенсивный процесс преобразования документов в цифровую форму независимо от того, являются ли эти документы открытыми для общего пользования или частными. В связи с переводом таких документов в цифровую форму многие - как отдельные люди, так и компании, в значительной степени заинтересованы в обеспечении и управлении безопасностью электронных документов. В связи с увеличением такой заинтересованности сейчас ведутся активные исследования мер противодействия несанкционированному вмешательству, перехвату, взлому и фальсификации электронных документов в различных областях. Что касается перехвата и "прослушивания" электронных документов, безопасность обеспечивается, например, посредством шифрования электронных документов. Далее, что касается фальсификации электронных документов, безопасность обеспечивается, например, посредством использования цифровых подписей. Однако если способы шифрования или цифровой подписи, которые предполагается использовать, не обладают высокой степенью устойчивости против взлома, достаточный уровень безопасности не обеспечивается.

Цифровая подпись используется для идентификации автора электронного документа. Соответственно, эта цифровая подпись должна быть такой, чтобы ее мог генерировать только автор электронного документа. Если какая-либо злонамеренная третья сторона сможет генерировать такую же самую цифровую подпись, такая третья сторона сможет выдать себя за автора электронного документа. Иными словами, электронный документ может быть фальсифицирован такой злонамеренной третьей стороной. Были высказаны самые разнообразные точки зрения относительно обеспечения защиты такой цифровой подписи с целью предотвращения подобной фальсификации. Сегодня широко используются такие алгоритмы цифровой подписи, как, например, RSA-алгоритм подписи и DSA-алгоритм подписи.

Указанный RSA-алгоритм подписи принимает за основу обеспечения безопасности принцип «трудности разложения большого составного числа на простые множители» (в последующем, проблема разложения на простые множители). Кроме того, DSA-алгоритм подписи принимает за основу обеспечения безопасности принцип «трудности решения проблемы дискретного логарифмирования». Эти принципы базируются на том факте, что не существуют алгоритмы, позволяющие эффективно решать проблему разложения на простые множители и проблему дискретного логарифмирования с использованием классического компьютера. Иными словами, упомянутые выше трудности предполагают вычислительные сложности для классического компьютера. Однако считается, что решения проблемы разложения на простые множители и проблемы дискретного логарифмирования могут быть эффективно вычислены с использованием квантового компьютера.

Аналогично RSA-алгоритму подписи и DSA-алгоритму подписи многие другие применяемые сегодня алгоритмы цифровой подписи и алгоритмы аутентификации с открытым ключом также опираются на трудности решения проблемы разложения на простые множители и проблемы дискретного логарифмирования как на основу обеспечения безопасности. Таким образом, если квантовые компьютеры войдут в сферу практического применения, безопасность таких алгоритмов цифровой подписи и алгоритмов аутентификации с открытым ключом больше обеспечена не будет.

Соответственно появляется потребность в реализации новых алгоритмов цифровой подписи и алгоритмов аутентификации с открытым ключом, где основой безопасности будут являться проблемы, отличные от таких проблем, как проблема разложения на простые множители и проблема дискретного логарифмирования, которые могут быть легко решены с помощью квантового компьютера. В качестве задачи, решить которую квантовому компьютеру будет нелегко, можно указать задачу, относящуюся к многочленам от нескольких переменных.

Например, в качестве алгоритмов цифровой подписи, использующих проблему многочленов от нескольких переменных в качестве основы безопасности, известны такие алгоритмы, как алгоритмы на основе криптографии Мацумото-Имаи (Matsumoto-Imai (MI)), криптографии с использованием уравнений скрытого поля (Hidden Field Equation (HFE)), алгоритмы подписи типа Oil-Vinegar (OV) signature и криптографии с использованием ручного преобразования (Tamed Transformation Method (TTM)). Например, алгоритм цифровой подписи на основе HFE-криптографии рассмотрен в следующей непатентной литературе 1 и 2.

Список литературы

Непатентная литература

Непатентная литература 1: Жак Патарен, "Асимметричная криптография со скрытым одночленом" (Jacques Patarin, Asymmetric Cryptography with a Hidden Monomial, CRYPTO 1996, pp. 45-60).

Непатентная литература 2: Патарен Ж., Куртуа Н. и Губин Л., "КВАРЦ, Цифровые подписи длиной 128 бит", (Patarin, J., Courtois, N., and Goubin, L., QUARTZ, 128-Bit Long Digital Signatures, In Naccache, D., Ed.). Topics in Cryptology - CT-RSA 2001 (San Francisco, CA, USA, April 2001), vol. 2020 of Lecture Notes in Computer Science, Springer-Verlag., pp. 282-297.

Раскрытие изобретения

Техническая проблема

Как указано выше, задача многочленов от нескольких переменных представляет собой пример так называемой НП-трудной (NP-hard) проблемы, которую трудно решить даже при использовании квантового компьютера. Обычно алгоритм аутентификации с открытым ключом, использующий проблему многочлена от нескольких переменных, описываемую уравнениями HFE или аналогичным способом, использует уравнение множества порядков с несколькими переменными с потайным входом. Например, используют уравнение $F(x_1,\ldots,x_n)=y$ множества порядков с несколькими переменными $x_1, \ldots, x_n$ и линейные преобразования A и B, управляемые секретным образом. В таком случае, уравнение F множества порядков с несколькими переменными и линейные преобразования A и B являются потайными входами.

Объект, знающий потайные входы F, A и B, может решить уравнение $B(F(A(x_1,\ldots,x_n)))=y'$ относительно переменных $x_1, \ldots, x_n$. С другой стороны, уравнение $B(F(A(x_1,\ldots,x_n)))=y'$ относительно переменных xi,..., Хп не может быть решено субъектом, который не знает потайных входов F, A и B. Используя этот механизм, можно реализовать алгоритм аутентификации с открытым ключом и алгоритм цифровой подписи, использующие трудности решения уравнения множества порядков с несколькими переменными в качестве основы для обеспечения безопасности.

Как описано выше, для реализации алгоритма аутентификации с открытым ключом или алгоритма цифровой подписи необходимо подготовить специальную систему уравнений множества порядков с несколькими переменными, удовлетворяющую

условию $B(F(A(x_1,\ldots,x_n)))=y$. Далее, в момент генерации подписи необходимо решить систему уравнений F множества порядков с несколькими переменными. По этой причине совокупность доступных систем уравнений F множества порядков с несколькими переменными до сих пор ограничивалась относительно легко решаемыми системами уравнений. Таким образом, ранее в известных схемах использовались только системы $B(F(A(x_1,\ldots,x_n)))=y$ уравнений множества порядков с несколькими переменными на основе сочетания трех функций (потайных входов) B, F и A, которые можно относительно решить, вследствие чего трудно обеспечить достаточную степень безопасности.

Соответственно, в свете изложенных выше обстоятельств, авторы настоящего изобретения разработали эффективные и в высокой степени защищенные алгоритм аутентификации с открытым ключом и алгоритм цифровой подписи с использованием систем уравнений множества порядков с несколькими переменными, для которых отсутствуют или присутствуют в недостаточном количестве средства для эффективного решения (потайные входы) (Коити Сакумото, Тайдзо Сираи и Харунага Хиватари, «Алгоритмы идентификации с открытым ключом на основе квадратичных полиномов от нескольких переменных» (Koichi Sakumoto, Taizo Shirai and Harunaga Hiwatari, "Public-Key Identification Schemes Based on Multivariate Quadratic Polynomials", CRYPTO 2011, LNCS 6841, pp. 706 to 723,2011.)).

Однако многочлены от нескольких переменных, используемые в алгоритме аутентификации открытого ключа и алгоритме цифровой подписи, имеют много коэффициентов. Таким образом, для использования многочлена от нескольких переменных в качестве открытого ключа необходимо определить способ эффективного назначения коэффициентов для доказывающей стороны и для проверяющей стороны (верификатора). Предлагаемая технология разработана для создания нового и усовершенствованного устройства обработки информации, нового и усовершенствованного способа обработки информации и новой и усовершенствованной программы, способных эффективно подставлять коэффициенты в многочлен от нескольких переменных.

Решение проблемы

Согласно одному из вариантов настоящего изобретения предложено устройство обработки информации, содержащее модуль генерации чисел, выполненный с возможностью генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными. Модуль назначения группирует коэффициенты членов, имеющих идентичное сочетание переменных, из совокупности коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является пара многочленов F множества порядков с несколькими переменными, и выполняет процедуру назначения в блоках групп.

Согласно другому варианту настоящего изобретения предложено устройство обработки информации, содержащее модуль генерации чисел, выполненный с возможностью генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными. Модуль назначения группирует матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, при этом модуль назначения выполняет процедуру назначения в блоках групп.

Согласно другому варианту настоящего изобретения предложено устройство обработки информации, содержащее модуль генерации чисел, выполненный с возможностью генерации чисел, количество которых идентично количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, заранее определенной между доказывающей стороной и проверяющей стороной.

Согласно другому варианту настоящего изобретения предложен способ обработки информации, содержащий этап генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и этап назначения генерируемых чисел коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными. На этапе назначения чисел группируют коэффициенты членов, имеющих идентичное сочетание переменных, из совокупности коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, и выполняют процедуру назначения в блоках групп.

Согласно другому варианту настоящего изобретения предложен способ обработки

информации, содержащий этап генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и этап назначения генерируемых чисел коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными входит в составляющие элементы. На этапе назначения чисел группируют матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, и выполняют процедуру назначения в блоках групп.

Согласно другому варианту настоящего изобретения предложен способ обработки информации, содержащий этап генерации чисел, количество которых идентично количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и этап назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, заранее определенной между доказывающей стороной и проверяющей стороной.

Согласно другому варианту настоящего изобретения предложена программа, вызывающая выполнение компьютером функции генерации чисел для генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и функции назначения для назначения чисел, генерируемых посредством функции генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными. Функция назначения группирует коэффициенты членов, имеющих идентичное сочетание переменных, из совокупности коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, и выполняет процедуру назначения в блоках групп.

Согласно другому варианту настоящего изобретения предложена программа, вызывающая выполнение компьютером функции генерации чисел для генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм

аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и функции назначения для назначения чисел, генерируемых посредством функции генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными. Функция назначения группирует матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, при этом функция назначения выполняет процедуру назначения в блоках групп.

Согласно другому варианту настоящего изобретения предложена программа, вызывающая выполнение компьютером функции генерации чисел для генерации количества чисел, идентичного количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными, и функции назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, заранее определенной между доказывающей стороной и проверяющей стороной.

Согласно другому варианту настоящего изобретения предложен компьютерный носитель записи, выполненный с возможностью записи описанной выше программы.

Полезные результаты изобретения

Согласно настоящему изобретению, как описано выше, можно эффективно подставлять коэффициенты многочлена от нескольких переменных.

Краткое описание чертежей

Фиг. 1 представляет пояснительную схему для описания структуры алгоритма аутентификации с открытым ключом.

Фиг. 2 представляет пояснительную схему для описания структуры алгоритма цифровой подписи.

Фиг. 3 представляет пояснительную схему для описания структуры алгоритма, относящейся к n-проходной схеме аутентификации с открытым ключом.

Фиг. 4 представляет пояснительную схему для описания эффективного 3-проходного алгоритма аутентификации с открытым ключом.

Фиг. 5 представляет пояснительную схему для описания параллелизации эффективного 3-проходного алгоритма аутентификации с открытым ключом.

Фиг. 6 представляет пояснительную схему для описания эффективного 3-проходного алгоритма аутентификации с открытым ключом.

Фиг. 7 представляет пояснительную схему для описания примера структуры эффективного 5-проходного алгоритма аутентификации с открытым ключом.

Фиг. 8 представляет пояснительную схему для описания параллелизации эффективного 5-проходного алгоритма аутентификации с открытым ключом.

Фиг. 9 представляет пояснительную схему для описания параллелизации эффективного 5-проходного алгоритма аутентификации с открытым ключом.

Фиг. 10 представляет пояснительную схему для описания способа модификации эффективного 3-проходного алгоритма аутентификации с открытым ключом для преобразования его в алгоритм цифровой подписи.

Фиг. 11 представляет пояснительную схему для описания способа модификации эффективного 5-проходного алгоритма аутентификации с открытым ключом для преобразования его в алгоритм цифровой подписи.

Фиг. 12 представляет пояснительную схему для описания примера структуры хэш-функции.

Фиг. 13 представляет пояснительную схему для описания способа верификации подписи (нормальный способ установки), относящегося к алгоритму цифровой подписи на основе 3-проходной схеме.

Фиг. 14 представляет пояснительную схему для описания способа верификации подписи (способ уменьшения объема памяти), относящегося к алгоритму цифровой подписи на основе 3-проходной схемы.

Фиг. 15 представляет пояснительную схему для описания способа верификации подписи (нормальный способ установки), относящегося к алгоритму цифровой подписи на основе 5-проходной схеме.

Фиг. 16 представляет пояснительную схему для описания способа верификации подписи (способ уменьшения объема памяти), относящегося к алгоритму цифровой подписи на основе 5-проходной схемы.

Фиг. 17 представляет пояснительную схему для описания способа (способ #1 выделения) выделения троичных случайных чисел из двоичных случайных чисел.

Фиг. 18 представляет пояснительную схему для описания способа (способ #2 выделения) выделения троичных случайных чисел из двоичных случайных чисел.

Фиг. 19 представляет пояснительную схему для описания способа (способ #3 выделения) выделения троичных случайных чисел из двоичных случайных чисел.

Фиг. 20 представляет пояснительную схему для описания способа (способ #3 выделения) выделения троичных случайных чисел из двоичных случайных чисел.

Фиг. 21 представляет пояснительную схему для описания способа (способ #3 выделения) выделения троичных случайных чисел из двоичных случайных чисел.

Фиг. 22 представляет пояснительную схему для описания способа структурирования данных (способ #1 структурирования) для эффективной подстановки коэффициентов многочленов от нескольких переменных.

Фиг. 23 представляет пояснительную схему для описания способа структурирования данных (способ #1 структурирования) для эффективной подстановки коэффициентов многочленов от нескольких переменных.

Фиг. 24 представляет пояснительную схему для описания способа структурирования данных (способ #1 структурирования) для эффективной подстановки коэффициентов многочленов от нескольких переменных.

Фиг. 25 представляет пояснительную схему для описания способа структурирования данных (способ #1 структурирования) для эффективной подстановки коэффициентов многочленов от нескольких переменных.

Фиг. 26 представляет пояснительную схему для описания способа структурирования данных (способ #1 структурирования) для эффективной подстановки коэффициентов многочленов от нескольких переменных.

Фиг. 27 представляет пояснительную схему для описания способа структурирования данных (способ #2 структурирования) для эффективной подстановки коэффициентов многочленов от нескольких переменных.

Фиг. 28 представляет пояснительную схему для описания примера конфигурации аппаратуры в устройстве для обработки информации, способном исполнять алгоритм согласно каждому из вариантов настоящего изобретения.

Осуществление изобретения

Далее, предпочтительные варианты настоящего изобретения будут описаны подробно со ссылками на прилагаемые чертежи. Отметим, что в настоящем описании и на чертежах элементам, имеющим по существу такие же функции, присвоены одинаковые позиционные обозначения, а повторные пояснения опущены.

Порядок описания

Здесь будет коротко изложен порядок описания вариантов настоящего изобретения, которое будет приведено ниже. Сначала, структура алгоритма аутентификации с открытым ключом будет рассмотрена со ссылками на Фиг. 1. Далее, структура алгоритма цифровой подписи будет рассмотрена со ссылками на Фиг. 2. Затем, n-проходный алгоритм аутентификации с открытым ключом будет рассмотрен со ссылками на Фиг. 3.

Далее, 3-проходный алгоритм аутентификации с открытым ключом будет рассмотрен со ссылками на Фиг. 4-6. Затем, 5-проходный алгоритм аутентификации с открытым ключом будет рассмотрен со ссылками на Фиг. 7-9. Далее, способ модификации эффективных 3-проходных и 5-проходных алгоритмов аутентификации с открытым ключом для преобразования этих алгоритмов в алгоритмы цифровой подписи будет рассмотрен со ссылками на Фиг. 10-11.

Затем, способы уменьшения объемов памяти, необходимых для верификации подписи во время выполнения алгоритмов цифровой подписи, относящихся к описанным здесь вариантам, будут рассмотрены со ссылками на Фиг. 12-16. Далее, способы эффективного выделения троичных случайных чисел из двоичных случайных чисел будут рассмотрены со ссылками Фиг. 17-21. Затем, способы эффективной подстановки коэффициентов многочленов от нескольких переменных будут рассмотрены со ссылками Фиг. 22-27. Далее, пример конфигурации аппаратуры в устройстве для обработки информации, способном реализовать каждый алгоритм согласно вариантам настоящего изобретения будет рассмотрен со ссылками Фиг. 28. Наконец, будет кратко изложено резюме технических принципов вариантов настоящего изобретения и преимуществ, обеспечиваемых этими техническими принципами.

Содержание

1. Введение

1-1: Алгоритм аутентификации с открытым ключом

1-2: Алгоритмы цифровой подписи

1-3: N-проходный алгоритм аутентификации с открытым ключом

2. Структуры 3-проходных алгоритмов аутентификации с открытым ключом

2-1: Пример конкретной структуры алгоритма

2-2: Пример структуры параллелизованного алгоритма

2-3: Пример структуры алгоритма на основе кубического многочлена от нескольких переменных

3: Структура 5-проходного алгоритма аутентификации с открытым ключом

3-1: Пример конкретной структуры алгоритма

3-2: Пример структуры параллелизованного алгоритма

3-3: Пример структуры алгоритма на основе кубического многочлена от нескольких переменных

4: Модификация алгоритма цифровой подписи

4-1: Модификация 3-проходного алгоритма аутентификации с открытым ключом и преобразование его в алгоритм цифровой подписи
4-2: Модификация 5-проходного алгоритма аутентификации с открытым ключом и преобразование его в алгоритм цифровой подписи
5: Способ уменьшения объема памяти, необходимого для верификации подписи
5-1: Структура хэш-функции
5-2: Пример применения 3-проходного алгоритма цифровой подписи
5-3: Пример применения 5-проходного алгоритма цифровой подписи
6: Способ выделения последовательности троичных случайных чисел из последовательности двоичных случайных чисел
6-1: Способ #1 выделения (2-битовое группирование)
6-2: Способ #2 выделения (нет группирования)
6-3: Способ #3 выделения (k-битовое группирование)
6-3-1: Базовая структура
6-3-2: Дополнительный способ выделения
7: Способ эффективной подстановки коэффициентов в многочлены от нескольких переменных
7-1: Базовое определение
7-2: Структурирование данных
7-2-1: Способ #1 структурирования
7-2-2: Способ #2 структурирования
7-2-3: Способ #3 структурирования
8: Пример конфигурации аппаратуры
9: Резюме

1. Введение

Предложенные здесь варианты относятся к алгоритму аутентификации с открытым ключом и алгоритму цифровой подписи, безопасность которых основана на трудности решения системы уравнений множества порядков с несколькими переменными. Однако предложенные здесь варианты отличаются от известных способов, таких как алгоритмы цифровой подписи на основе уравнений HFE, и относятся к алгоритму аутентификации с открытым ключом и алгоритму цифровой подписи, использующих системы уравнений множества порядков с несколькими переменными, для которых не хватает средств и способов эффективного решения (функций с потайными ходами). Сначала будут кратко обобщены алгоритмы аутентификации с открытым ключом, алгоритмы цифровой подписи и n-проходные алгоритмы аутентификации с открытым ключом.

1-1: Алгоритм аутентификации с открытым ключом

Сначала, обзор алгоритмов аутентификации с открытым ключом будет рассмотрен со ссылками на Фиг. 1. Фиг. 1 представляет пояснительную схему для описания структуры алгоритма аутентификации с открытым ключом.

Аутентификация с открытым ключом используется, когда человек (доказывающая сторона) подтверждает другому человеку (проверяющей стороне), что это он сам, с использованием открытого ключа pk и секретного ключа sk. Например, открытый ключ pkд доказывающей стороны A делают известным проверяющей стороне B. С другой стороны, доказывающая сторона A сохраняет свой секретный ключ sk_A в секрете. Согласно принципу алгоритма аутентификации с открытым ключом человек, который знает секретный ключ sk_A, соответствующий открытому ключу pk_A, рассматривается как сама доказывающая сторона A.

Для того чтобы доказывающая сторона A доказала проверяющей стороне B, что

она действительно является доказывающей стороной A, с использованием процедуры алгоритма аутентификации с открытым ключом, эта доказывающая сторона A предоставляет проверяющей стороне B доказательство, что она знает секретный ключ sk_A, соответствующий открытому ключу pk_A. Тогда проверяющей стороне B представляют доказательство, указывающее, что доказывающая сторона A знает секретный ключ sk_A, и если проверяющая сторона B способна подтвердить это доказательство, достоверность доказывающей стороны A (факт, что она действительно является доказывающей стороной A) считается и является доказанной.

Однако установление процедуры аутентификации с открытым ключом требует выполнения следующих условий для обеспечения безопасности.

Первое условие формулируется так: «уменьшить насколько это возможно вероятность фальсификации со стороны фальсификатора, не имеющего секретного ключа sk, во время выполнения интерактивного протокола». Состояние, когда это первое условие выполняется, называется «корректность» (soundness). Другими словами, корректность означает, что «фальсификатор, с большой вероятностью не обладающий секретным ключом sk, ничего не фальсифицировал во время выполнения интерактивного протокола». Второе условие формулируется так: «даже при выполнении интерактивного протокола совсем не происходит утечки информации о секретном ключе sk_A доказывающей стороны A к проверяющей стороне B». Состояние, когда это второе условие выполняется, называется «нулевое разглашение» (zero knowledge).

Обеспечение безопасности с использованием алгоритма аутентификации с открытым ключом включает применение интерактивного протокола (далее - протокол взаимодействия), обладающего и корректностью, и нулевым разглашением. Если, гипотетически, процедура аутентификации осуществляется с использованием протокола взаимодействия, не обладающего корректностью и нулевым разглашением, появится конечная (ненулевая) вероятность фальшивой верификации и конечная вероятность разглашения информации секретного ключа, так что в результате достоверность доказывающей стороны не будет доказана, даже если сама процедура будет завершена успешно. Следовательно, вопрос о том, как обеспечить корректность и нулевое разглашение сеансового протокола, является очень важным.

Модель

В модели алгоритма аутентификации с открытым ключом присутствуют два субъекта - а именно доказывающая сторона и проверяющая сторона, как показано на Фиг. 1. Доказывающая сторона генерирует пару из открытого ключа pk и секретного ключа sk с использованием алгоритма Gen генерации ключей. Затем доказывающая сторона выполняет протокол взаимодействия с проверяющей стороной с использованием пары из секретного ключа sk и открытого ключа pk, сформированных посредством алгоритма Gen генерации ключей. В этот момент доказывающая сторона выполняет протокол взаимодействия с использованием алгоритма P доказывающей стороны. Как описано выше, в рамках протокола взаимодействия доказывающая сторона предоставляет проверяющей стороне доказательства, что она обладает секретным ключом sk, с использованием алгоритма P доказывающей стороны.

С другой стороны, проверяющая сторона выполняет протокол взаимодействия с использованием алгоритма V проверяющей стороны и тем самым проверяет, обладает ли доказывающая сторона секретным ключом, соответствующим открытому ключу, который опубликовала доказывающая сторона. Иными словами, проверяющая сторона представляет собой субъект, проверяющий, обладает ли доказывающая сторона

секретным ключом, соответствующим открытому ключу. Как описано здесь, модель алгоритма аутентификации с открытым ключом конфигурирована из двух субъектов, а именно - доказывающей стороны и проверяющей стороны, и трех алгоритмов, а именно - алгоритма Gen генерации ключей, алгоритма P доказывающей стороны и алгоритма V проверяющей стороны.

Кроме того, термины «доказывающая сторона» ("prover") и «проверяющая сторона» ("verifier") используются в последующем описании, но они везде обозначают строго «субъекты». Таким образом, субъект, выполняющий алгоритм Gen генерации ключей и алгоритм P доказывающей стороны, представляет собой устройство для обработки информации, соответствующее субъекту «доказывающая сторона». Аналогично, субъект, выполняющий алгоритм V проверяющей стороны, представляет собой устройство для обработки информации. Пример аппаратной конфигурации устройства для обработки информации показан на Фиг. 28, например. Иными словами, алгоритм Gen генерации ключей, алгоритм P доказывающей стороны и алгоритм V проверяющей стороны реализуются центральным процессором CPU 902 на основе программы, записанной в постоянном запоминающем устройстве (ПЗУ ROM) 904, в запоминающем устройстве с произвольной выборкой (ЗУПВ RAM) 906, в модуле 920 памяти, на сменном носителе 928 записи или на аналогичном носителе.

Алгоритм Gen генерации ключей

Указанный алгоритм Gen генерации ключей, используется доказывающей стороной. Этот алгоритм Gen генерации ключей представляет собой алгоритм, генерирующий пару и открытого ключа pk и секретного ключа sk, уникальную для соответствующей доказывающей стороны. Открытый ключ pk, сформированный алгоритмом Gen генерации ключей, публикуется. После этого опубликованный открытый ключ pk используется проверяющей стороной. С другой стороны, доказывающая сторона сохраняет секретность сформированного алгоритмом Gen генерации ключей секретного ключа sk. Секретный ключ sk, с которым доказывающая сторона обращается секретно, используется для доказательства проверяющей стороне, что доказывающая сторона обладает секретным ключом sk, соответствующим открытому ключу pk.

Формализованно, алгоритм Gen генерации ключей представлен формулой (1) ниже в качестве алгоритма, получающего на вход параметр защищенности 1λ (λ - целое число, равное 0 или больше) и генерирующего на выходе секретный ключ sk и открытый ключ pk.

$$(sk, pk) \leftarrow Gen(1^{\lambda}) \quad \cdots (1)$$

Алгоритм P доказывающей стороны

Рассматриваемый алгоритм P доказывающей стороны используется доказывающей стороной. Алгоритм P доказывающей стороны представляет собой алгоритм для доказывания проверяющей стороне, что доказывающая сторона обладает секретным ключом sk, соответствующим открытому ключу pk. Другими словами, алгоритм P доказывающей стороны представляет собой алгоритм, использующий открытый ключ pk и секретный ключ sk в качестве входных данных и выполняющий протокол взаимодействия.

Алгоритм V проверяющей стороны

Рассматриваемый алгоритм V проверяющей стороны используется проверяющей стороной. Алгоритм V проверяющей стороны представляет собой алгоритм для

проверки, обладает ли доказывающая сторона секретным ключом sk, соответствующим открытому ключу pk, во время выполнения сеансового протокола. Алгоритм V проверяющей стороны представляет собой алгоритм, который получает на вход открытый ключ pk и генерирует на выходе 0 или 1 (1 бит) в соответствии с результатами выполнения сеансового протокола. В этот момент, проверяющая сторона принимает решение, что доказывающая сторона является недостоверной, если алгоритм V проверяющей стороны генерирует на выходе 0, и признает достоверность доказывающей стороны, если алгоритм V проверяющей стороны генерирует на выходе 1. Формализованно алгоритм V проверяющей стороны представлен следующей формулой (2).

$$0/1 \leftarrow V(pk) \quad \cdots (2)$$

Как описано выше, реализация значимой аутентификации с открытым ключом включает обладание протоколом взаимодействия, удовлетворяющим двум условиям - корректности и нулевого разглашения. Однако доказательство того факта, что доказывающая сторона владеет секретным ключом sk включает выполнение этой доказывающей стороной процедуры, зависящей от этого секретного ключа sk, и затем сообщение проверяющей стороне результата, после чего проверяющая сторона осуществляет верификацию на основе содержания полученного сообщения. Для обеспечения корректности выполняют процедуру, зависящую от секретного ключа sk. В то же время нельзя передавать проверяющей стороне никакой информации о секретном ключе sk. По этой причине, упомянутые выше алгоритм Gen генерации ключей, алгоритм P доказывающей стороны и алгоритм V проверяющей стороны грамотно построены так, чтобы удовлетворять этим требованиям.

Изложенное выше, таким образом, суммирует принципы алгоритма аутентификации с открытым ключом

1-2: Алгоритмы цифровой подписи

Далее, обзор алгоритмов цифровой подписи будет рассмотрен со ссылками на Фиг. 2. Фиг. 2 представляет пояснительную схему для обзора алгоритмов цифровой подписи.

В отличие от бумажных документов, нет необходимости физически подписывать или скреплять печатью данные в цифровой форме. По этой причине, доказательство достоверности создателя данных в цифровой форме предусматривает использование электронного механизма, создающего эффекты, аналогичные физическому подписанию бумажного документа или скреплению документа печатью. Этот механизм представляет собой цифровые подписи. Цифровая подпись обозначает механизм, ассоциирующий конкретные данные с данными подписи, известными только создателю данных, передающий данные подписи получателю и проверяющий эти данные подписи на стороне получателя.

Модель

Как показано на Фиг. 2, в модели алгоритма цифровой подписи присутствуют два субъекта - подписывающая сторона и проверяющая сторона. Кроме того, модель алгоритма цифровой подписи построена из трех алгоритмов: алгоритма Gen генерации ключей, алгоритма Sig генерации подписи и алгоритма Ver верификации подписи.

Подписывающая сторона использует алгоритм Gen генерации ключей для формирования пары из ключа sk подписи и проверочного ключа pk, уникальной для этой подписывающей стороны. Подписывающая сторона использует также алгоритм Sig генерации подписи с целью генерации цифровой подписи q для присоединения к

сообщению M. Другими словами, подписывающая сторона является субъектом, прикрепляющим цифровую подпись к сообщению M. В то же время, проверяющая сторона использует алгоритм Ver верификации подписи для проверки цифровой подписи, прикрепленной к сообщению M. Другими словами, проверяющая сторона - это субъект, проверяющий цифровую подпись q с целью подтвердить, является ли создатель сообщения M подписывающей стороной.

Отметим, что где бы термины «подписывающая сторона» ("signer") и «проверяющая сторона» ("verifier") ни использовались в последующем описании, они в конечном счете обозначают субъектов. Следовательно, агент, выполняющий алгоритм Gen генерации ключей и алгоритм Sig генерации подписи, представляет собой устройство для обработки информации, соответствующее субъекту «подписывающая сторона». Аналогично, агент, выполняющий алгоритм Ver верификации подписи, представляет собой устройство для обработки информации. Вариант аппаратной конфигурации устройства для обработки информации показан на Фиг. 28, например. Иными словами, алгоритм Gen генерации ключей, алгоритм Sig генерации подписи и алгоритм Ver верификации подписи реализуются устройством, таким как центральный процессор CPU 902, на основе программы, записанной в постоянном запоминающем устройстве (ПЗУ ROM) 904, в запоминающем устройстве с произвольной выборкой (ЗУПВ RAM) 906, в модуле 920 памяти или на сменном носителе 928 записи.

Алгоритм Gen генерации ключей

Указанный алгоритм Gen генерации ключей, используется подписывающей стороной. Этот алгоритм Gen генерации ключей представляет собой алгоритм, генерирующий пару из ключа sk подписи и проверочного ключа pk, уникальную для соответствующей подписывающей стороны. Проверочный ключ pk, сформированный алгоритмом Gen генерации ключей, делается общедоступным (публикуется). В то же время, подписывающая сторона сохраняет секретность ключа sk подписи, сформированного алгоритмом Gen генерации ключей. Этот ключ sk подписи затем используется для генерации цифровой подписи q с целью прикрепления ее к сообщению M. Например, алгоритм Gen генерации ключей получает на вход параметр 1^p защищенности (где p - целое число, равное или больше 0), и передает на выход ключ sk подписи и проверочный ключ pk. В таком случае алгоритм Gen генерации ключей может быть формализован и выражен следующей формулой (3).

$$(sk, pk) \leftarrow Gen(1^{\lambda}) \quad \cdots (3)$$

Алгоритм Sig генерации подписи

Указанный алгоритм Sig генерации подписи используется подписывающей стороной. Алгоритм Sig генерации подписи представляет собой алгоритм для генерации цифровой подписи q с целью прикрепления ее к сообщению M. Алгоритм Sig генерации подписи представляет собой алгоритм, получающий ключ sk подписи и сообщение M в качестве входных данных и передающий на выход цифровую подпись q. Формализованно указанный алгоритм Sig генерации подписи представлен следующей формулой (4).

$$\sigma \leftarrow Sig(sk, M) \quad \cdots (4)$$

Алгоритм Ver верификации подписи

Указанный алгоритм Ver верификации подписи используется проверяющей стороной. Алгоритм Ver верификации подписи представляет собой алгоритм для проверки, является ли цифровая подпись q действительной и цифровой подписью для сообщения M. Этот алгоритм Ver верификации подписи представляет собой алгоритм, получающий проверочный ключ pk подписывающей стороны, сообщение M и цифровую подпись q в качестве входных данных и передающий на выход 0 или 1 (1 бит). Формализованно указанный алгоритм Ver верификации подписи представлен следующей формулой (5). В этот момент, проверяющая сторона принимает решение, что цифровая подпись q является недостоверной, если алгоритм Ver верификации подписи генерирует на выходе 0 (если проверочный ключ pk отклоняет сообщение M и цифровую подпись q), и признает достоверность цифровой подписи q, если алгоритм Ver верификации подписи генерирует на выходе 1 (если проверочный ключ pk принимает сообщение M и цифровую подпись q).

$$0/1 \leftarrow Ver(pk, M, \sigma) \quad \cdots (5)$$

Изложенное выше, таким образом, суммирует принципы алгоритма цифровой подписи.

1-3: N-проходный алгоритм аутентификации с открытым ключом

Далее, n-проходныи алгоритм аутентификации с открытым ключом будет рассмотрен со ссылками на Фиг. 3. Фиг. 3 представляет поясняющую схему, иллюстрирующую пример n-проходного алгоритма аутентификации с открытым ключом.

Как описано выше, алгоритм аутентификации с открытым ключом представляет собой алгоритм аутентификации, доказывающий проверяющей стороне, что доказывающая сторона обладает секретным ключом sk, соответствующим открытому ключу pk, во время выполнения протокола взаимодействия. Кроме того, протокол взаимодействия должен удовлетворять двум условиям - корректности и нулевого разглашения. По этой причине во время выполнения протокола взаимодействия обе стороны - доказывающая сторона и проверяющая сторона, обмениваются информацией n раз в ходе осуществления соответствующих процедур, как показано на Фиг. 3.

В случае n-проходного алгоритма аутентификации с открытым ключом доказывающая сторона выполняет процедуру с использованием алгоритма P доказывающей стороны (операция #1) и передает информацию T_1 проверяющей стороне. После этого проверяющая сторона выполняет процедуру с использованием алгоритма V проверяющей стороны (операция #2) и передает информацию T_2 доказывающей стороне. Такое выполнение процедур и передачу информации T_k осуществляют последовательно для k=3-n, и, наконец, осуществляет процедуру (операция #n+1). Передача и прием информации таким способом n раз называется, таким образом, «n-проходной» схемой аутентификации с открытым ключом.

Изложенное выше, таким образом, описывает n-проходный алгоритм аутентификации с открытым ключом

2. Структуры 3-проходных алгоритмов аутентификации с открытым ключом

Далее будут рассмотрены алгоритмы, относящиеся к классу 3-проходных алгоритмов аутентификации с открытым ключом. Отметим, что в последующем описании 3-проходный алгоритм аутентификации с открытым ключом может быть также назван в некоторых случаях «3-проходной схемой» или «3-проходным алгоритмом».

2-1: Пример конкретной структуры алгоритма (Фиг. 4)

Сначала, пример конкретной структуры алгоритма, относящейся к 3-проходной схеме, будет рассмотрен со ссылками на Фиг. 4. Фиг. 4 представляет пояснительную схему для описания конкретной структуры алгоритма, относящейся к 3-проходной схеме. Здесь будет описан случай использования пары квадратичных многочленов ($f_1(x), \ldots, f_m(x)$) в качестве части открытого ключа pk. Здесь предполагается, что квадратичный многочлен $f_i(x)$ представлен следующей формулой (6). Кроме того, вектор $(x_1, \ldots, x_n)$ обозначен x, а пара квадратичных многочленов $(f_1(x), \ldots, f_m(x))$ от нескольких переменных обозначены как многочлены F(x) от нескольких переменных.

$$f_i(x_1, \cdots, x_n) = \sum_{j,k} a_{ijk} x_j x_k + \sum_j b_{ij} x_j$$

… (6)

Кроме того, указанная пара квадратичных многочленов $(f_1(x), \ldots, f_m(x))$ может быть представлена следующей формулой (7). В дополнение к этому, $A_1, \ldots, A_m$ представляет собой матрицу n × n. Далее, каждый из $b_1, \ldots, b_m$ представляет собой вектор n × 1.

$$F(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix} = \begin{pmatrix} x^T A_1 x + b_1^T x \\ \vdots \\ x^T A_m x + b_m^T x \end{pmatrix}$$

… (7)

При использовании этих формул многочлен F может быть выражен следующими формулой (8) и формулой (9). На основе следующей формулы (10) можно легко подтвердить, что это выражение удовлетворяется.

$$F(x+y) = F(x) + F(y) + G(x,y)$$

… (8)

$$G(x,y) = \begin{pmatrix} y^T (A_1^T + A_1) x \\ \vdots \\ y^T (A_m^T + A_m) x \end{pmatrix}$$

… (9)

$$f_l(x+y) = (x+y)^T A_l(x+y) + b_l^T(x+y)$$
$$= x^T A_l x + x^T A_l y + y^T A_l x + y^T A_l y + b_l^T x + b_l^T y$$
$$= f_l(x) + f_l(y) + x^T A_l y + y^T A_l x$$
$$= f_l(x) + f_l(y) + x^T (A_l^T)^T y + y^T A_l x$$
$$= f_l(x) + f_l(y) + (A_l^T x)^T y + y^T A_l x$$
$$= f_l(x) + f_l(y) + y^T (A_l^T x) + y^T A_l x$$
$$= f_l(x) + f_l(y) + y^T (A_l^T + A_l)x$$

... (10)

Если F(x+y) разбить на первую часть, зависящую только от x, вторую часть, зависящую только от y, и третью часть, зависящую и от x, и от y, тогда член G(x, y), соответствующий третьей части, становится билинейным относительно x и y. В последующем этот член G(x, y) будет также называться билинейным членом. Использование этого свойства позволяет построить эффективный алгоритм.

Например, используйте вектор t_0, являющийся элементом множества K^n, и вектор eo, являющийся элементом множества K^m, для представления многочлена $F_1(x)$ от нескольких переменных, служащего для маскирования многочленов F(x+r) от нескольких переменных, в виде $F_1(x)=G(x, t_0)+e_0$. В этом случае сумма многочленов $F(x+r_0)$ и G(x) от нескольких переменных, представлена формулой (11) ниже. Здесь, если $t_1=r_0+t_0$, $e_1 =F(r_0)+e_0$, многочлен $F_2(x)=F(x+r_0)+F_1(x)$ от нескольких переменных может быть представлен вектором t_1, являющимся элементом множества K^n, и вектором e_1, являющимся элементом множества K^m. По этой причине, когда задан многочлен $F_1(x) =G(x, t_0)+e_0$, многочлены F_1 и F_2 могут быть выражены с использованием вектора из множества K^n и вектора из множества K^m, и, следовательно, можно реализовать эффективный алгоритм, которому для связи и обмена информацией необходим лишь небольшой объем данных.

$$F(x+r_0) + F_1(x)$$
$$= F(x) + F(r_0) + G(x, r_0) + G(x, t_0) + e_0$$
$$= F(x) + G(x, r_0 + t_0) + F(r_0) + e_0$$

... (11)

Кроме того, из многочлена F_2 (или F_1) совсем нет утечки информации относительно r_0. Например, даже если даны e_1 и t_1 (или e_0 и t_0), информация относительно го остается совсем неизвестной до тех пор, пока неизвестны e_0 и t_0 (или e_1 и t_1). Соответственно, обеспечивается нулевое разглашение. Далее будут описаны алгоритмы, относящиеся

к классу 3-проходных алгоритмов и построенные на основе изложенной выше логики. Алгоритм по 3-проходной схеме, который будет описан здесь ниже, построен из алгоритма Gen генерации ключей, алгоритма P доказывающей стороны и алгоритма V проверяющей стороны.

Алгоритм Gen генерации ключей

Алгоритм Gen генерации ключей формирует m многочленов $f_1(x_1, \ldots, xn), \ldots, f_m(x_1, \ldots, x_n)$ от нескольких переменных, определенных в кольце k, и вектор $s=(s_1, \ldots, s_n)$, являющийся элементом множества K^n. Далее алгоритм Gen генерации ключей вычисляет $y=(y1, \ldots, y_m) \leftarrow (f_1(s), \ldots, f_m(s))$. Кроме того, алгоритм Gen генерации ключей задает $(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n), y)$ в открытом ключе pk и задает s в качестве секретного ключа. В последующем, вектор $(x_1, \ldots, x_n)$ обозначен x, а пара многочленов $(f_1(x), \ldots, f_m(x))$ от нескольких переменных обозначена как F(x).

Алгоритм P доказывающей стороны, алгоритм V проверяющей стороны

В дальнейшем, процедура, выполняемая алгоритмом P доказывающей стороны, и процедура, выполняемая алгоритмом V проверяющей стороны, в ходе работы по протоколу взаимодействия будут описаны со ссылками на Фиг. 4. Во время исполнения протокола взаимодействия доказывающая сторона совсем не допускает утечки информации о секретном ключе s и сообщает проверяющей стороне, что «она сама знает величину s, удовлетворяющую уравнению y=F(s)». С другой стороны, проверяющая сторона осуществляет проверку, действительно ли доказывающая сторона знает величину s, удовлетворяющую уравнению y=F(s). Предполагается, что открытый ключ pk, сформированный алгоритмом Gen генерации ключей, публикуется. Кроме того, предполагается, что доказывающая сторона сохраняет секретность своего секретного ключа s. В дальнейшем описание будет дано со ссылками на логическую схему, изображенную на Фиг. 4.

Операция #1:

Как показано на Фиг. 4, алгоритм P доказывающей стороны сначала случайным образом генерирует вектор r_0, t_0, являющийся элементом множества K^n, и вектор e_0, являющийся элементом множества K^m. После этого алгоритм P доказывающей стороны вычисляет $r_1 \leftarrow -s-r_0$. Эти расчеты эквивалентны маскированию секретного ключа s с использованием вектора r_0. Кроме того алгоритм P доказывающей стороны вычисляет $t_1 \leftarrow -r_0-t_0$. После этого алгоритм P доказывающей стороны вычисляет $e_1 \leftarrow -F(r_0) -e_0$.

Операция #1 (продолжение):

После этого алгоритм P доказывающей стороны вычисляет $c_0 \leftarrow -H(r_1, G(t_0, r_1)+e_0)$. После этого алгоритм P доказывающей стороны вычисляет $c_1 \leftarrow -H(t_0, e_0)$. После этого алгоритм P доказывающей стороны вычисляет $c2 \leftarrow -H(t_1, e_1)$. Сообщение (c_0, c_1, c_2), генерируемое в ходе операции #1 посылают алгоритму V проверяющей стороны.

Операция #2:

После приема сообщения (c_0, c_1, c_2) алгоритм V проверяющей стороны выбирает, какую схему использовать, из совокупности трех схем верификации. Например, алгоритм V проверяющей стороны может выбрать числовую величину из совокупности трех числовых величин {0, 1, 2}, представляющих схемы верификации, и задает выбранную числовую величину в качестве параметра Запрос (challenge) Ch. Этот запрос Ch передают алгоритму P доказывающей стороны.

Операция #3:

По получении запроса Ch алгоритм P доказывающей стороны генерирует ответ Rsp для передачи алгоритму V проверяющей стороны в ответ на принятый запрос Ch. Если Ch=0, алгоритм P доказывающей стороны генерирует ответ $Rsp=(r_0, t_1, e_1)$. Если Ch=1, алгоритм P доказывающей стороны генерирует ответ $Rsp=(r_1, t_0, e_0)$. Если Ch=2, алгоритм P доказывающей стороны генерирует ответ $Rsp=(r_1, t_1, e_1)$. Ответ Rsp, генерируемый в ходе операции #3, посылают алгоритму V проверяющей стороны.

Операция #4:

После приема ответа Rsp алгоритм V проверяющей стороны выполняет следующую процедуру верификации с использованием принятого ответа Rsp.

Если Ch=0, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_1=H(r_0-t_1, F(r_0)-e_1)$. Кроме того, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_2=H(t_1, e_1)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Если Ch=1, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_0=H(r_1, G(t_0, r_1)+e_0)$. Кроме того, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_1=H(t_0, e_0)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Если Ch=2, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_0=H(r_1, y-F(r_1)-G(t_1, r_1)-e_1)$. Кроме того, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_2=H(t_1, e_1)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Выше был описан пример структуры эфективного алгоритма, относящегося к 3-проходной схеме.

2-2: Пример структуры параллелизованного алгоритма (Фиг. 5)

Далее, способ параллелизации алгоритма, относящегося к 3-проходной схеме, показанной на Фиг. 4, будет описан со ссылками на Фиг. 5. Однако дальнейшее описание структуры алгоритма Gen генерации ключей будет опущено.

На деле, применение описанного выше сеансового протокола делает возможным сохранение вероятности успешной фальсификации на уровне 2/3 или меньше. Следовательно, выполнение этого сеансового протокола дважды делает возможным сохранение вероятности успешной фальсификации на уровне $(2/3)^2$ или меньше. Более того, если рассматриваемый сеансовый протокол выполнить N раз, вероятность успешной фальсификации становится не больше $(2/3)^N$, а если в качестве N задать достаточно большое число (N=140, например), вероятность успешной фальсификации становится пренебрежимо малой.

В качестве способов многократного выполнения протокола взаимодействия можно представить себе последовательный способ, в соответствии с которым последовательно и многократно производят операции обмена сообщениями, передачи запроса и получения ответа, и параллельный способ, когда большим числом сообщений, запросов

и ответов обмениваются в единственном цикле обмена, например. Кроме того, можно также представить себе гибридный способ, сочетающий последовательный способ и параллельный способ. Здесь алгоритмы, выполняющие описанный выше протокол взаимодействия, относящийся к 3-проходной схеме, параллельно (далее называемые параллелизованными алгоритмами) будут теперь описаны со ссылками на Фиг. 5.

Операция #1:

Как показано на Фиг. 5, алгоритм P доказывающей стороны сначала выполняет следующие процессы с (1) по (6) для i=1-N.

Процесс (1): Алгоритм P доказывающей стороны генерирует векторы r_{0i}, t_{0i}, являющиеся элементами множества K^n, и вектор e_{0i}, являющийся элементом множества K^m.

Процесс (2): Алгоритм P доказывающей стороны вычисляет r_{1i}<-s-r_{0i}. Эти расчеты эквивалентны маскированию секретного ключа s с использованием вектора r_{0i}. Кроме того алгоритм P доказывающей стороны вычисляет t_{1i}<-r_{0i}+t_{0i}.

Процесс (3): Алгоритм P доказывающей стороны вычисляет e_{1i}<-F(r_{0i})-e_{0i}.

Процесс (4): Алгоритм P доказывающей стороны вычисляет c_{0i}<-H(r_{1i}, G(r_{1i}, t_{0i})+e_{0i}).

Процесс (5): Алгоритм P доказывающей стороны вычисляет c_{1i}<-H(t_{0i}, e_{0i}).

Процесс (6): Алгоритм P доказывающей стороны вычисляет c_{2i}<-H(t_{1i}, e_{1i}).

Операция #1 (продолжение):

После выполнения описанных выше процессов с (1) по (6) для i=1-N, алгоритм P доказывающей стороны вычисляет Cmt<-H(c_{01}, c_{11}, c_{21}, …, c_{0N}, c_{1N}, c_{2N}). Хэш-величину Cmt, генерируемую в ходе операции #1, передают алгоритму V проверяющей стороны. Таким путем сообщение (c_{01}, c_{11}, c_{21}, …, c_{0N}, c_{1N}, c_{2N}) преобразуют в хэш-величину, передаваемую алгоритму V проверяющей стороны, тем самым позволяя уменьшить объем обмена информацией в процессе связи.

Операция #2:

После приема хэш-величины Cmt алгоритм V проверяющей стороны выбирает схему верификации для использования из совокупности трех схем верификации для каждого из i=1-N. Например, алгоритм V проверяющей стороны может, для каждого из i=1-N, выбрать числовую величину из совокупности трех цифровых величин {0, 1, 2}, представляющих схемы верификации, и задать выбранную числовую величину в качестве параметра запроса Ch_i. Параметры запросов Ch_1, …, Ch_N передают алгоритму P доказывающей стороны.

Операция #3:

По получении запросов Ch_1, …, Ch_N алгоритм P доказывающей стороны генерирует ответы Rsp_1, …, Rsp_N для передачи алгоритму V проверяющей стороны в ответ на каждый из принятых запросов Ch_1, …, Ch_N. Если Ch_i=0, алгоритм P доказывающей стороны генерирует ответ Rsp_i=(r_{0i}, t_{1i}, e_{1i}, c_{0i}). Если Ch_i=1, алгоритм P доказывающей стороны генерирует ответ Rsp_i=(r_{1i}, t_{0i}, e_{0i}, c_{2i}). Если Ch_i=2, алгоритм P доказывающей стороны генерирует ответ Rsp_i=(r_{1i}, t_{1i}, e_{1i}, c_{1i}).

Ответы Rsp_1, …, Rsp_N, генерируемые в ходе операции #3, посылают алгоритму V проверяющей стороны.

Операция #4:

После приема ответов Rsp_1, …, Rsp_N алгоритм V проверяющей стороны выполняет следующие процессы с (1) по (3) для i=1-N с использованием принятых ответов Rsp_1, …, Rsp_N. Здесь алгоритм V проверяющей стороны выполняет процесс (1) для случая $Ch_i=0$, процесс (2) для случая $Ch_i=1$ и процесс (3) для случая $Ch_i=2$.

Процесс (1): Если $Ch_i=0$, алгоритм V проверяющей стороны выделяет $(r_{0i}, t_{1i}, e_{1i}, c_{0i})$ на основе Rsp_i. После этого, алгоритм V проверяющей стороны вычисляет $c_{1i}=H(r_{0i}-t_{1i}, F(r_{0i})-e_{1i})$. Кроме того, алгоритм V проверяющей стороны вычисляет $c_{2i}=H(t_{1i}, e_{1i})$. Затем алгоритм V проверяющей стороны сохраняет (c_{0i}, c_{1i}, c_{2i}).

Процесс (2): Если $Ch_i=1$, алгоритм V проверяющей стороны выделяет $(r_{1i}, t_{0i}, e_{0i}, c_{2i})$ на основе Rsp_i. После этого, алгоритм V проверяющей стороны вычисляет $c_{0i}=H(r_{1i}, G(t_{0i}, r_{1i})+e_{0i})$. Кроме того, алгоритм V проверяющей стороны вычисляет $c_{1i}=H(t_{0i}, e_{0i})$. Затем алгоритм V проверяющей стороны сохраняет (c_{0i}, c_{1i}, c_{2i}).

Процесс (3): Если $Ch_i=2$, алгоритм V проверяющей стороны выделяет $(r_{1i}, t_{1i}, e_{1i}, c_{1i})$ на основе Rsp_i. После этого, алгоритм V проверяющей стороны вычисляет $H(r_{1i}, y-F(r_{1i})-G(t_{1i}, r_{1i})-e_{1i})$. Кроме того, алгоритм V проверяющей стороны вычисляет $c_{2i}=H(t_{1i}, e_{1i})$. Затем алгоритм V проверяющей стороны сохраняет (c_{0i}, c_{1i}, c_{2i}).

После выполнения описанных выше процессов с (1) по (3) для i=1-N, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $Cmt=H(c_{01}, c_{11}, c_{21}, …, c_{0N}, c_{1N}, C_{2N})$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если процедура верификации закончилась неудачей.

Выше был описан пример структуры параллелизованного эффективного алгоритма, относящегося к 3-проходной схеме.

2-3: Пример структуры алгоритма на основе кубического многочлена от нескольких переменных

Здесь будет рассмотрен способ построения эффективного алгоритма с использованием кубического многочлена f_1 для кольца R, выраженного следующей формулой (12). Многочлен $F=(f_1, …, f_m)$ от нескольких переменных, выраженный с применением пары кубических многочленов f_1 удовлетворяет следующей формуле (13). Здесь, $G_x(x, y)$ представляет линейный член от x. Кроме того, $G_y(x, y)$ представляет линейный член от y. Когда выражены $G_x=(gx_1,…, g_{xm})$ и $G_y=(g_{y1}, …, g_{ym})$, можно вывести параметры g_{x1} и g_{y1} согласно следующим формуле (14) и формуле (15), соответственно. Здесь, поскольку второй член справа в составе g_{x1} также является линейным по одной из переменных x и y, этот второй член справа может содержать g_{y1}.

$$f_l(x_1,\cdots,x_n)=\sum_{i,j,k}a_{lijk}x_ix_jx_k+\sum_{i,j}b_{lij}x_ix_j+\sum_i c_{li}x_i$$

... (12)

$$F(x+y)-F(x)-F(y)=G_x(x,y)+G_y(x,y)$$

... (13)

$$g_{xl}(x_1,\cdots,x_n,y_1,\cdots,y_n)=\sum_{i,j,k}\left(a_{lijk}+a_{likj}+a_{lkji}\right)y_iy_jx_k+\sum_{i,j}\left(b_{lij}+b_{lji}\right)x_iy_j$$

... (14)

$$g_{yl}(x_1,\cdots,x_n,y_1,\cdots,y_n)=\sum_{i,j,k}\left(a_{lijk}+a_{likj}+a_{lkji}\right)x_ix_jy_k$$

... (15)

Как понятно из приведенных выше формулы (14) и формулы (15), многочлены $G_x(x, y)$ и $G_y(x, y)$ становятся аддитивно гомоморфными по x и y. Таким образом, с использованием этого свойства открытый ключ F(s) разделен путем введения новых переменных r_0, r_1, t_0, u_0 и e_0, как в способе построения эффективного алгоритма с использованием квадратичного многочлена f_i.

Поскольку многочлены G_x и G_y являются аддитивно гомоморфными, взаимосвязи между формулами с (16) по (19) установлены с использованием переменных r_0, r_1, t_0, u_0 и e_0. Следующие формулы с (16) по (19) могут быть разделены на первую часть, воспроизводимую с использованием (r_0, t_0, u_0, e_0), вторую часть, воспроизводимую с использованием (r_0, u_1, e_1), третью часть, воспроизводимую с использованием (r_1, t_0, e_0), и четвертую часть, воспроизводимую с использованием (r_1, t_1, u_1, e_1).

Например, "r_0, t_0", включенные в следующую формулу (17), "u_0", включенный в следующую формулу (18), и "$F(r_0)$, $G_y(r_0, u_0)$, e_0", включенные в следующую формулу (19) являются первой частью. Кроме того, "$G_y(r_0, u_1)$, e1", включенные в следующую формулу (19), являются второй частью. Кроме того, "e_0, $G_x(r_0, r_1)$", включенные в следующую формулу (16), являются третьей частью. Кроме того, "e_1, $F(r_1)$, $G_x(t_1, r_1)$", включенные в следующую формулу (16), "t_1", включенный в следующую формулу (17), и "u_1", включенный в следующую формулу (18), составляют четвертую часть.

Другими словами, следующая формула (16) включает третью и четвертую части, следующая формула (17) и следующая формула (18) включают первую и четвертую части и следующая формула (19) включает первую и вторую части. Таким образом, каждая из следующих формул с (16) по (19) включает два вида частей.

Само определение секретного ключа s и взаимосвязь между следующими формулами (16)-(19) обеспечивают, что секретный ключ s невозможно получить, даже если используется какая-либо из групп (r_0, t_0, u_0, e_0), (r_0, u_1, e_1), (r_1, t_0, e_0) и (r_1, t_1, u_1, e_1). Использование этого свойства позволяет, например, построить эффективный алгоритм (далее именуемый - расширенный алгоритм) с использованием кубического многочлена f_1 на кольце R.

$$F(r_0 + r_1) = e_0 + e_1 + F(r_1) + G_x(t_0, r_1) + G_x(t_1, r_1) \quad \cdots (16)$$

$$r_0 = t_0 + t_1 \quad \cdots (17)$$

$$r_1 = u_0 + u_1 \quad \cdots (18)$$

$$F(r_0) + G_y(r_0, u_1) + G_y(r_0, u_0) = e_0 + e_1 \quad \cdots (19)$$

Далее будет описан пример структуры конкретного расширенного алгоритма. К работе расширенного алгоритма относятся два базовых момента, а именно - проверяющей стороне направляют сообщение, выраженное одной из следующих формул (20)-(22), и проверяют одну из указанных частей с первой по четвертую. Однако только такая проверка может и не позволить проверить, что параметр "r_1", входящий в состав третьей части, идентичен параметру "r_1", входящему в состав четвертой части. Аналогично, может оказаться невозможно проверить, что параметр "r_0", входящий в состав первой части, идентичен параметру "r_0", входящему в состав второй части, или что параметры "t_0, e_0", входящие в состав первой части, идентичны параметрам "t_0, e_0", входящим в состав третьей части. Кроме того, может оказаться невозможно проверить, что параметры "u_1, e_1", входящие в состав второй части, идентичны параметрам "u_1, e_1", входящим в состав четвертой части. Соответственно, ниже будет приведен пример структуры, позволяющей осуществить такую проверку.

$$c_0 = H(G_x(t_0, r_1) + e_0)$$

... (20)

$$c_1 = H(t_0, u_0)$$

... (21)

$$c_2 = H(e_1 - G_y(r_0, u_1))$$

... (22)

2-3-1: Базовая структура (Фиг. 7)

Далее, пример конфигурации расширенного алгоритма, относящегося к 3-проходной схеме, будет рассмотрен со ссылками на Фиг. 6. Однако дополнительное описание структуры алгоритма Gen генерации ключей будет опущено.

Операция #1:

Как показано на Фиг. 6, алгоритм P доказывающей стороны случайным образом генерирует векторы r_0, t_0, u_0, являющиеся элементами множества K^n, и вектор e_0, являющийся элементом множества K^m. После этого алгоритм P доказывающей стороны вычисляет $r_1 \leftarrow -s-r_0$. Эти расчеты эквивалентны маскированию секретного ключа s с использованием вектора го. После этого алгоритм P доказывающей стороны вычисляет $t_1 \leftarrow -r_0+t_0$. После этого алгоритм P доказывающей стороны вычисляет $u_1 \leftarrow -r_1+u_0$. После этого алгоритм P доказывающей стороны вычисляет $e_1 \leftarrow -F(r_0)-e_0$.

Операция #1 (продолжение):

После этого алгоритм P доказывающей стороны вычисляет $c_0 \leftarrow -H(r_1, G_x(t_0, r_1)+e_0)$. После этого алгоритм P доказывающей стороны вычисляет $c_1 \leftarrow -H(r_0-t_0, u_0)$. После этого алгоритм P доказывающей стороны вычисляет $c_2 \leftarrow -H(r_0, e_1-G_y(r_0, u_1))$. После этого алгоритм P доказывающей стороны вычисляет $c_3 \leftarrow -H(t_0, e_0)$. После этого алгоритм P доказывающей стороны вычисляет $c_4 \leftarrow -H(u_1, e_1)$. Сообщения ($c_0$, c_1, c_2, c_3, c_4), генерируемые в ходе операции #1 посылают алгоритму V проверяющей стороны.

Операция #2:

После приема сообщений (c_0, c_1, c_2, c_3, c_4) алгоритм V проверяющей стороны выбирает, какую схему использовать, из совокупности четырех схем верификации. Например, алгоритм V проверяющей стороны может выбрать числовую величину из совокупности четырех числовых величин {0, 1, 2, 3}, представляющих схемы верификации, и задает выбранную числовую величину в качестве параметра Запрос Ch. Этот запрос Ch передают алгоритму P доказывающей стороны.

Операция #3:

По получении запроса Ch алгоритм P доказывающей стороны генерирует ответ Rsp

для передачи алгоритму V проверяющей стороны в ответ на принятый запрос Ch. Если Ch=0, алгоритм P доказывающей стороны генерирует ответ $Rsp=(r_0, t_0, u_0, e_0)$. Если Ch=1, алгоритм P доказывающей стороны генерирует ответ $Rsp=(r_0, u_1, e_1)$. Если Ch=2, алгоритм P доказывающей стороны генерирует ответ $Rsp=(r_1, t_0, e_0)$. Если Ch=3, алгоритм P доказывающей стороны генерирует ответ $Rsp=(r_1, t_1, u_1, e_1)$. Ответ Rsp, генерируемый в ходе операции #3, посылают алгоритму V проверяющей стороны.

Операция #4:

После приема ответа Rsp алгоритм V проверяющей стороны выполняет следующую процедуру верификации с использованием принятого ответа Rsp.

Если Ch=0, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_1=H(r_0-t_0, u_0)$. После этого, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_2=H(r_0, F(r_0)+G_y(r_0, u_0)-e_0)$. После этого, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_3=H(t_0, e_0)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Если Ch=1, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_2=H(r_0, e_1-G_y(r_0, u_1))$. После этого, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_4=H(u_1, e_1)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Если Ch=2, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_0=H(r_1, e_0-G_x(t_0, r_1))$. После этого, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_3=H(t_0, e_0)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Если Ch=3, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_0=H(r_1, y-F(r_1)-e_1-G_x(t_1, r_1))$. После этого, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_1=H(t_1, r_1, u_1)$. После этого, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_4=H(u_1, e_1)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Выше был описан пример структуры расширенного алгоритма, относящегося к 3-проходной схеме. Использование кубического многочлена позволяет реализовать более высокую степень защищенности.

3: Структура 5-проходного алгоритма аутентификации с открытым ключом

Далее будут рассмотрены алгоритмы, относящиеся к классу 5-проходных алгоритмов аутентификации с открытым ключом. Отметим, что в последующем описании 5-проходный алгоритм аутентификации с открытым ключом может быть также в некоторых случаях назван «5-проходной схемой» или «5-проходным алгоритмом».

В случае 3-проходной схемы вероятность ложной верификации составляет 2/3 на каждый цикл выполнения протокола взаимодействия. Однако в случае 5-проходной схемы вероятность ложной верификации составляет 1/2+1/q на каждый цикл выполнения протокола взаимодействия. Здесь q обозначает порядок кольца, которое нужно использовать. Соответственно, когда порядок кольца достаточно велик, вероятность ложной верификации на один цикл 5-проходной схемы можно уменьшить, и, таким образом, вероятность ложной верификации может быть значительно уменьшена при выполнении протокола взаимодействия небольшое число раз.

Например, если нужно, чтобы вероятность ложной верификации была не больше $1/2^n$, протокол взаимодействия в 3-проходной схеме нужно выполнить не меньше n/(log3-1)=1.701n раз. С другой стороны, если нужно, чтобы вероятность ложной верификации была не больше $1/2^n$, протокол взаимодействия в 5-проходной схеме нужно выполнить не меньше n/(1-log(1+1/q)) раз. Соответственно, когда q=24, объем информации связи, необходимый для обеспечения одного и того же уровня защищенности, в 5-проходной схеме меньше, чем в 3-проходной схеме.

3-1: Пример конкретной структуры алгоритма (Фиг. 7)

Сначала, пример конкретной структуры алгоритма, относящейся к 5-проходной схеме, будет рассмотрен со ссылками на Фиг. 7. Фиг. 7 представляет пояснительную схему для описания конкретной структуры алгоритма, относящейся к 5-проходной схеме. Здесь будет описан случай использования пары квадратичных многочленов ($f_1(x), \ldots, f_m(x)$), как части открытого ключа pk. Здесь предполагается, что квадратичный многочлен $f_i(x)$ представлен приведенной выше формулой (6). Кроме того, вектор ($x_1, \ldots, x_n$) обозначен x, а пара квадратичных многочленов ($f_1(x), \ldots, f_m(x)$) от нескольких переменных обозначены как многочлены F(x) от нескольких переменных.

Как в эффективных алгоритмах, относящихся к 3-проходной схеме, два вектора, а именно вектор t_0, являющийся элементом множества K^n, и вектор e_0, являющийся элементом множества K^m, используют для представления многочлена $F_1(x)$ от нескольких переменных, служащего для маскирования многочленов $F(x+r_0)$ от нескольких переменных, в виде $F_1(x)=G(x, t_0)+e_0$. При использовании этих формул можно для многочлена $F(x+r_0)$ от нескольких переменных получить соотношение, выраженное следующей формулой (23).

$$\begin{aligned} & Ch_A \cdot F(x+r_0) + F_1(x) \\ & = Ch_A \cdot F(x) + Ch_A \cdot F(r_0) + Ch_A \cdot G(x, r_0) + G(x, t_0) + e_0 \\ & = Ch_A \cdot F(x) + G(x, Ch_A \cdot r_0 + t_0) + Ch_A \cdot F(r_0) + e_0 \end{aligned} \quad \cdots (23)$$

По этой причине, когда $t_1=Ch_A \cdot r_0+t_0$, $e_1=Ch_A \cdot F(r_0)+e_0$, многочлен $F_2(x)=Ch_A \cdot F(x+r_0)+F_1(x)$ от нескольких переменных после маскирования также может быть выражен двумя векторами, а именно вектором t_1, который является элементом множества K^n, и вектором e_1, который является элементом множества K^m. По этой причине, когда задан многочлен $F_1(x)=G(x, t_0)+e_0$, многочлены F_1 и F_2 могут быть выражены с использованием вектора

из множества K^n и вектора из множества K^m, и, следовательно, можно реализовать эффективный алгоритм, которому для связи необходим лишь небольшой объем данных.

Кроме того, из многочлена F_2 (или F_1) совсем нет утечки информации относительно r_0. Например, даже если даны e_1 и t_1 (или e_0 и t_0), информация относительно r_0 остается совсем неизвестной до тех пор, пока неизвестны e_0 и t_0 (или e_1 и t_1). Соответственно, обеспечивается нулевое разглашение. Далее будут описаны алгоритмы, относящиеся к классу 5-проходных алгоритмов и построенные на основе изложенной выше логики. Алгоритм по 5-проходной схеме, который будет описан здесь, построен из алгоритма Gen генерации ключей, алгоритма P доказывающей стороны и алгоритма V проверяющей стороны.

Алгоритм Gen генерации ключей

Алгоритм Gen генерации ключей формирует многочлены $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$ от нескольких переменных, определенные в кольце k и вектор $s=(s_1, \ldots, s_n)$, являющийся элементом множества K^n. Далее алгоритм Gen генерации ключей вычисляет $y=(y1, \ldots, y_m)<-(f_1(s), \ldots, f_m(s))$. Кроме того, алгоритм Gen генерации ключей задает ($f_1 \ldots, f_m$, y) в открытом ключе pk и задает s в качестве секретного ключа. В последующем, вектор $(x_1, \ldots, x_n)$ обозначен x, а пара многочленов $(f_1(x), \ldots, f_m(x))$ от нескольких переменных обозначена как F(x).

Алгоритм P доказывающей стороны, алгоритм V проверяющей стороны

В дальнейшем, процедура, выполняемая алгоритмом P доказывающей стороны, и процедура, выполняемая алгоритмом V проверяющей стороны, в ходе работы по протоколу взаимодействия будут описаны со ссылками на Фиг. 7. Во время исполнения протокола взаимодействия доказывающая сторона совсем не допускает утечки информации о секретном ключе s и сообщает проверяющей стороне, что «она сама знает величину s, удовлетворяющую уравнению y=F(s)». С другой стороны, проверяющая сторона осуществляет проверку, действительно ли доказывающая сторона знает величину s, удовлетворяющую уравнению y=F(s). При этом, предполагается, что открытый ключ pk сделан известным проверяющей стороне. Кроме того, предполагается, что доказывающая сторона сохраняет секретность своего секретного ключа s. В дальнейшем описание будет дано со ссылками на логическую схему, изображенную на Фиг. 7.

Операция #1:

Как показано на Фиг. 7, алгоритм P доказывающей стороны случайным образом генерирует вектор r_0, являющийся элементом множества K^n, вектор t_0, являющийся элементом множества K^n, и вектор e_0, являющийся элементом множества K^m. После этого алгоритм P доказывающей стороны вычисляет $r_1<-s-r_0$. Эти расчеты эквивалентны маскированию секретного ключа s с использованием вектора r_0. После этого алгоритм P доказывающей стороны вычисляет хэш-величину co для векторов r_0, t_0, e_0. Иными словами, алгоритм P доказывающей стороны вычисляет $c_0<-H(r_0, t_0, e_0)$. После этого алгоритм P доказывающей стороны вычисляет $G(t_0, r_1)+e_0$ и хэш-величину c_1 от r_1. Иными словами, алгоритм P доказывающей стороны вычисляет $c_0<-H(r_1, G(t_0, r_1)+e_0)$. Сообщения ($c_0$, c_1), генерируемые в ходе операции #1, посылают алгоритму V проверяющей стороны.

Операция #2:

После приема сообщений (c_0, c_1), алгоритм V проверяющей стороны случайным образом выбирает одно число Ch_A из совокупности q колец K и передает выбранное число Ch_A алгоритму P доказывающей стороны.

Операция #3:

После приема числа Ch_A, алгоритм P доказывающей стороны вычисляет t_1<-$Ch_A \cdot r_0$-t_0. Кроме того, алгоритм P доказывающей стороны вычисляет e_1<-$Ch_A \cdot F(r_0)$-e_0. Алгоритм P доказывающей стороны передает t_1 и e_1 алгоритму V проверяющей стороны.

Операция #4:

После приема t_1 и e_1 алгоритм V проверяющей стороны выбирает, какую схему верификации использовать, из совокупности двух схем верификации. Например, алгоритм V проверяющей стороны может выбрать числовую величину из совокупности двух числовых величин {0, 1}, представляющих схемы верификации, и задает выбранную числовую величину в качестве параметра запроса Ch. Этот запрос Ch_B передают алгоритму P доказывающей стороны.

Операция #5:

По получении запроса Ch_B алгоритм P доказывающей стороны генерирует ответы Rsp для передачи алгоритму V проверяющей стороны в ответ на принятый запрос Ch_B. Если Ch_B=0, алгоритм P доказывающей стороны генерирует ответ Rsp=r_0. Если Ch_B=1, алгоритм P доказывающей стороны генерирует ответ Rsp=r_1. Ответ Rsp, генерируемый в ходе операции #5, посылают алгоритму V проверяющей стороны.

Операция #6:

После приема ответа Rsp алгоритм V проверяющей стороны выполняет следующую процедуру верификации с использованием принятого ответа Rsp.

Если Ch_B=0, алгоритм V проверяющей стороны выполняет r_0<-Rsp. Затем, алгоритм V проверяющей стороны проверяет, выполняется ли равенство c_0=$H(r_0, Ch_A \cdot r_0 - t_1, Ch_A \cdot F(r_0) - e_1)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Если Ch_B=1, алгоритм V проверяющей стороны выполняет r_1<-Rsp. Затем алгоритм V проверяющей стороны проверяет, выполняется ли равенство c_1=$H_1(r_1, Ch_A \cdot (y - F(r_1) - G(t_1, r_1) - e_1))$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Выше был описан пример структуры эффективного алгоритма, относящегося к 5-проходной схеме.

3-2: Пример структуры параллелизованного алгоритма (Фиг. 8)

Далее, способ параллелизации алгоритма, относящегося к 5-проходной схеме, показанной на Фиг. 7, будет описан со ссылками на Фиг. 8. Однако дальнейшее описание структуры алгоритма Gen генерации ключей будет опущено.

Как описано выше, применение рассмотренного выше протокола взаимодействия, относящегося к 5-проходной схеме, делает возможным сохранение вероятности

успешной фальсификации на уровне (1/2+1/q) или меньше. Следовательно, выполнение этого протокола взаимодействия дважды делает возможным сохранение вероятности успешной фальсификации на уровне $(1/2+1/q)^2$ или меньше. Более того, если рассматриваемый протокол взаимодействия выполнить N раз, вероятность успешной фальсификации становится не больше $(1/2+1/q)^N$, а если в качестве N задать достаточно большое число (N=80, например), вероятность успешной фальсификации становится пренебрежимо малой.

В качестве способов многократного выполнения протокола взаимодействия можно представить себе последовательный способ, в соответствии с которым последовательно и многократно производят операции обмена сообщениями, передачи запроса и получения ответа, и параллельный способ, когда большим числом сообщений, запросов и ответов обмениваются в единственном цикле обмена, например. Кроме того, можно также представить себе гибридный способ, сочетающий последовательный способ и параллельный способ. Здесь алгоритмы, выполняющие описанный выше протокол взаимодействия, относящийся к 5-проходной схеме, параллельно (далее называемые параллелизованными алгоритмами) будут теперь описаны.

Операция #1:

Как показано на Фиг. 8, алгоритм P доказывающей стороны сначала выполняет следующие процессы с (1) по (4) для i=1-N.

Процесс (1): Алгоритм P доказывающей стороны генерирует векторы r_{0i}, t_{0i}, являющиеся элементами множества K^n, и вектор e_{0i}, являющийся элементом множества K^m.

Процесс (2): Алгоритм P доказывающей стороны вычисляет r_{1i}<-s-r_{0i}. Эти расчеты эквивалентны маскированию секретного ключа s с использованием вектора r_{0i}.

Процесс (3): Алгоритм P доказывающей стороны вычисляет c_{0i}<-H(r_{0i}, t_{0i}, e_{0i}).

Процесс (4): Алгоритм P доказывающей стороны вычисляет c_{1i}<-H(r_{1i}, G(t_{0i}, r_{1i})+e_{0i}).

После выполнения описанных выше процессов с (1) по (4) для i=1-N, алгоритм P доказывающей стороны вычисляет хэш-величину Cmt<-H(c_{01}, c_{11}, …, c_{0N}, c_{1N}). Хэш-величину Cmt, генерируемую в ходе операции #1, посылают алгоритму V проверяющей стороны.

Операция #2:

После приема хэш-величины Cmt, алгоритм V проверяющей стороны случайным образом выбирает одно число Ch_{Ai} из совокупности q колец K для i=1-N и передает выбранное число Ch_{Ai} (i=1-N) алгоритму P доказывающей стороны.

Операция #3:

После приема числа Ch_{Ai}(i=1-N), алгоритм P доказывающей стороны вычисляет t_{1i}<-$Ch_{Ai}\cdot r_{0i}$-t_{0i} для i=1-N. Кроме того, алгоритм P доказывающей стороны вычисляет e_{1i}<-$Ch_{Ai}\cdot F(r_{0i})$-e_{0i} для i=1-N. Далее, алгоритм P доказывающей стороны вычисляет хэш-величины d<-H(t_{011}, e_{11}, …, t_{1N}, e_{1N}). Затем, алгоритм P доказывающей стороны передает полученную хэш-величину алгоритму V проверяющей величины.

Операция #4:

После приема хэш-величины алгоритм V проверяющей стороны выбирает схему верификации для использования из совокупности двух схем верификации для каждого из i=1-N. Например, алгоритм V проверяющей стороны может, для каждого из i=1-N,

выбрать числовую величину из совокупности двух цифровых величин {0, 1}, представляющих схемы верификации, и задать выбранную числовую величину в качестве параметра запроса Ch_{Bi}. Этот запрос Ch_{Bi}(i=1-N) передают алгоритму P доказывающей стороны.

Операция #5:

По получении запроса Ch_{Bi} (i=1-N) алгоритм P доказывающей стороны генерирует ответ Rsp_i для передачи алгоритму V проверяющей стороны в ответ на принятый запрос Ch_{Bi} для i=1-N. Если Ch_{Bi}=0, алгоритм P доказывающей стороны генерирует ответ $Rsp_i=(r_{0i}, c_{1i})$. Если Ch_{Bi}=1, алгоритм P доказывающей стороны генерирует ответ $Rsp_i=(r_{0i}, t_{0i}, e_{0i}, c_1t_{1i}, e_{1i}, c_{0i})$. Ответ Rsp_i (i=1-N), генерируемый в ходе операции #5, посылают алгоритму V проверяющей стороны.

Операция #6:

После приема ответов Rsp_i (i=1-N) алгоритм V проверяющей стороны выполняет следующие процессы (1) и (2) с использованием принятого ответа Rsp_i (i=1-N).

Процесс (1): Если Ch_{Bi}=0, алгоритм V проверяющей стороны выполняет $(r_{0i}, t_{0i}, e_{0i}, c_{1i})$<-$Rsp_i$. Алгоритм V проверяющей стороны вычисляет $c_{0i}=H(r_{0i}, t_{0i}, e_{0i})$. Далее, алгоритм V проверяющей стороны вычисляет t_{1i}<-$Ch_{Ai}\cdot r_{0i}+t_{0i}$, и e_{1i}<-$Ch_{Ai}\cdot F(r_{0i})-e_{0i}$. Затем алгоритм V проверяющей стороны сохраняет $(c_{0i}, c_{1i}, t_{1i}, e_{1i})$.

Процесс (2): Если Ch_{Bi}=1, алгоритм V проверяющей стороны выполняет $(r_{1i}, t_{1i}, e_{1i}, c_{0i})$<-$Rsp_i$. После этого, алгоритм V проверяющей стороны вычисляет $c_{1i}=H(r_{1i}-Ch_{Ai}\cdot(y-F(r_{1i}))-G(t_{1i}, r_{1i})-e_{1i})$. Затем алгоритм V проверяющей стороны сохраняет $(c_{0i}, c_{1i}, t_{1i}, e_{1i})$.

После вьшолнения процессов с (1) по (2) для i=1-N, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $Cmt=H(c_{0i}, c_{11}, \ldots, c_{0N}, c_{1N})$. После этого, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $d=H(t_{11}, e_{11}, \ldots, t_{1N}, e_{1N})$. Затем, алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Выше был описан пример структуры параллелизованного эффективного алгоритма, относящегося к 5-проходной схеме.

3-3: Пример структуры алгоритма на основе кубического многочлена от нескольких переменных

Здесь будет рассмотрен способ построения эффективного алгоритма с использованием кубического многочлена f_1 для кольца R, как в случае 3-проходной схемы. Когда кубический многочлен f_1 выражен, как в приведенной выше формуле (12), из формулы (14) и формулы (15) можно понять тот факт, что $G_x(x, y)$ и $G_y(x, y)$ становятся линейными по x и y.

Таким образом, используя описанное выше свойство, открытый ключ F(s) разделен на член, кратный Ch_A, путем введения новых переменных r_0, r_1, t_0, u_0 и e_0. Поскольку многочлены G_x и G_y являются линейными по x и y, взаимосвязи между формулами с (24) по (27) установлены с использованием переменных r_0, r_1, t_0, u_0 и e_0. Следующие формулы с (24) по (27) могут быть разбиты на первую часть, зависящую от Ch_A, и вторую часть, не зависящую от Ch_A. Здесь первую часть можно воспроизвести с

использованием (r_1, t_1, u_1, e_1). Вторую часть можно воспроизвести с использованием (r_0, t_1, u_1, e_1).

Например, "e_0, $G_x(t_0, r_1)$", включенные в следующую формулу (32), "t_0", включенный в следующую формулу (33), "u_0" включенный в следующую формулу (26), и "e_0, $G_y(r_0, u_0)$", включенные в следующую формулу (27) являются первыми частями. С другой стороны, "$Ch_{Ai} \cdot F(r_0+r_1)$, e_1, $Ch_A \cdot F(r_1)$, $G_x(t_1, r_1)$", включенные в следующую формулу (24), "$Ch_A \cdot r_0$, t_1", включенные в следующую формулу (25), "$Ch_A \cdot r_1$, u_1", включенные в следующую формулу (34), и "$Ch_A \cdot F(r_0)$, $G_y(r_0, u_1)$, e_1", включенные в следующую формулу (27), являются вторыми частями.

Само определение секретного ключа s и взаимосвязь между следующими формулами (24)-(27) обеспечивают, что секретный ключ s невозможно получить, даже если используется какая-либо из групп (r_1, t_1, u_1, e_1) или (r_0, t_1, u_1, e_1). Использование этого свойства позволяет, например, построить эффективный алгоритм (далее именуемый - расширенный алгоритм) с использованием кубического многочлена f_1 на кольце R.

$$Ch_A \cdot F(r_0 + r_1) = e_0 + e_1 + Ch_A \cdot F(r_1) + G_x(t_0, r_1) + G_x(t_1, r_1) \quad \cdots (24)$$

$$Ch_A \cdot r_0 = t_0 + t_1 \quad \cdots (25)$$

$$Ch_A \cdot r_1 = u_0 + u_1 \quad \cdots (26)$$

$$Ch_A \cdot F(r_0) + G_y(r_0, u_1) + G_y(r_0, u_0) = e_0 + e_1 \quad \cdots (27)$$

Далее будет описан пример структуры конкретного расширенного алгоритма. К работе расширенного алгоритма относятся два базовых момента, а именно - проверяющей стороне направляют сообщение, выраженное одной из следующих формул (28)-(29), и проверяют часть, зависящую от параметра от Ch_A, для величины этого параметра Ch_A, выбранной проверяющей стороной. Здесь, поскольку «векторы r_0 и r_1, используемые в момент генерации сообщения, нельзя в момент верификации заменить другими векторами r_0 и r_1», ниже будет приведен пример структуры, в которую добавлена верификация по r_0 и r_1.

$$c_0 = H(t_0, e_0 - G_y(r_0, u_0)) \quad \cdots (28)$$

$$c_1 = H(u_0, G_x(t_0, r_1) + e_0) \quad \cdots (29)$$

Далее, базовая конфигурация расширенного алгоритма, относящегося к 5-проходной схеме, будет рассмотрена со ссылками на Фиг. 9. Однако дальнейшее описание структуры алгоритма Gen генерации ключей будет опущено.

Операция #1:

Как показано на Фиг. 9, алгоритм P доказывающей стороны случайным образом генерирует векторы r_0, t_0, u_0, являющиеся элементами множества K^n, и вектор e_0, являющийся элементом множества K^m. После этого алгоритм P доказывающей стороны вычисляет $r_1 \leftarrow -s - r_0$. Эти расчеты эквивалентны маскированию секретного ключа s с использованием вектора r_0. После этого алгоритм P доказывающей стороны вычисляет $c_0 \leftarrow H(r_0, t_0, e_0 - G_y(r_0, u_0))$. После этого алгоритм P доказывающей стороны вычисляет $c_1 \leftarrow H(r_1, u_0, G_x(t_0, r_1) + e_0)$. Сообщения (c_0, c_1), генерируемые в ходе операции #1, посылают алгоритму V проверяющей стороны.

Операция #2:

После приема сообщений (c_0, c_1) алгоритм V проверяющей стороны случайным образом выбирает число Ch_A. Это число Ch_A передают алгоритму P доказывающей стороны.

Операция #3:

После приема числа Ch_A, алгоритм P доказывающей стороны вычисляет $t_1 \leftarrow -Ch_A \cdot r_0 - t_0$. После этого, алгоритм P доказывающей стороны вычисляет $u_1 \leftarrow -Ch_A \cdot r_1 - u_0$. После этого, алгоритм P доказывающей стороны вычисляет $e_1 \leftarrow -Ch_A \cdot F(r_0) + Ch_A \cdot G_y(r_0, r1) - e_0$. Затем, (t_1, u_1, e_1), генерируемые в ходе операции #3, посылают алгоритму V проверяющей стороны.

Операция #4:

После приема (t_1, u_1, e_1) алгоритм V проверяющей стороны выбирает, какую схему верификации использовать, из совокупности двух схем верификации. Например, алгоритм V проверяющей стороны может выбрать числовую величину из совокупности двух числовых величин {0, 1}, представляющих схемы верификации, и задает выбранную числовую величину в качестве параметра запроса Ch_B. Этот запрос Ch_B передают алгоритму P доказывающей стороны.

Операция #5:

По получении запроса Ch_B алгоритм P доказывающей стороны генерирует ответ Rsp для передачи алгоритму V проверяющей стороны в ответ на принятый запрос Ch_B. Если $Ch_B=0$, алгоритм P доказывающей стороны генерирует ответ $Rsp=r_0$. Если $Ch_B=1$, алгоритм P доказывающей стороны генерирует ответ $Rsp=r_1$. Ответ Rsp, генерируемый в ходе операции #5, посылают алгоритму V проверяющей стороны.

Операция #6:

После приема ответа Rsp алгоритм V проверяющей стороны выполняет следующую процедуру верификации с использованием принятого ответа Rsp.

Если $Ch_B=0$, алгоритм V проверяющей стороны проверяет, выполняется ли равенство

$c_0=H(r_0, Ch_A \cdot r_0-t_1, Ch_A \cdot F(r_0)+G_y(r_0, u_1)-e_1)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Если $Ch_B=1$, алгоритм V проверяющей стороны проверяет, выполняется ли равенство $c_1=H(r_1, Ch_A \cdot r_1-u_1, Ch_A \cdot (y-F(r_1))-G_x(t_1, r_1)-e_1)$. Алгоритм V проверяющей стороны передает на выход величину 1, чтобы указать на успех аутентификации, если все эти процедуры верификации завершились успешно, и передает на выход величину 0, чтобы указать на неудачу аутентификации, если какая-либо процедура верификации закончилась неудачей.

Выше был описан пример структуры расширенного алгоритма, относящегося к 5-проходной схеме. Кроме того, использование кубического многочлена позволяет реализовать более высокую степень защищенности.

4: Модификация алгоритма цифровой подписи

Далее, будет рассмотрен способ модификации изложенного выше алгоритма аутентификации с открытым ключом для преобразования его в алгоритм цифровой подписи.

Когда доказывающая сторона в модели алгоритма аутентификации с открытым ключом совпадает с подписывающей стороной в случае алгоритма цифровой подписи, аппроксимация модели алгоритма цифровой подписи может быть легко понята, если только доказывающая сторона может обращаться к проверяющей стороне. На основе этого подхода будет рассмотрен способ модификации изложенного выше алгоритма аутентификации с открытым ключом для преобразования его в алгоритм цифровой подписи.

4-1: Модификация 3-проходного алгоритма аутентификации с открытым ключом и преобразование его в алгоритм цифровой подписи (Фиг. 10)

Сначала будет рассмотрен способ модификации 3-проходного алгоритма аутентификации с открытым ключом для преобразования его в алгоритм цифровой подписи.

На Фиг. 10 показан эффективный алгоритм (например, см. Фиг. 5), относящийся к 3-проходной схеме с интерактивностью три раза и четырьмя операциями, т.е. с операции #1 по операцию #4.

Операция #1 содержит процесс (1) генерации $a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i}, c_{2i})$ и процесс (2) вычисления $Cmt<-H(c_{0i}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$. Параметр Cmt, генерируемые алгоритмом P доказывающей стороны в ходе операции #1, посылают алгоритму V проверяющей стороны.

Операция #2 содержит процедуру выбора параметра $Ch_1, \ldots, Ch_N$. Параметр $Ch_1, \ldots, Ch_N$, выбранный в ходе операции #2 посредством алгоритма V проверяющей стороны, посылают алгоритму P доказывающей стороны.

Операция #3 содержит процедуру генерации ответов $Rsp_1, \ldots, Rsp_N$ с использованием $Ch_1, \ldots, Ch_N$ и $a_1 \ldots, a_N$. Этот процесс выражен в виде $Rsp_i<-Select (Ch_i, a_i)$. Ответы $Rsp_1, \ldots, Rsp_N$, генерируемые алгоритмом P доказывающей стороны в ходе операции #3, посылают алгоритму V проверяющей стороны.

Операция #4 содержит процесс (1) воспроизведения $c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$ с использованием $Ch_1, \ldots, Ch_N$ и $Rsp_1, \ldots, Rsp_N$ и процесс (2) верификации $Cmt=H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$ с использованием воспроизведенных $c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$

.

Алгоритм аутентификации с открытым ключом, представленный в виде операций с операции #1 по операцию #4 модифицирован для преобразования в алгоритм Sig генерации подписи и алгоритм Ver верификации подписи, изображенные на Фиг. 10.

(Алгоритм Sig генерации подписи)

Сначала будет описана алгоритма Sig генерации подписи. Алгоритм Sig генерации подписи содержит процессы с (1) по (5),

Процесс (1): Алгоритм Sig генерации подписи генерирует $a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i}, c_{2i})$.

Процесс (2): Алгоритм Sig генерации подписи вычисляет $Cmt<-H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$.

Процесс (3): Алгоритм Sig генерации подписи вычисляет $(Ch_1, \ldots, Ch_N)<-H(M, Cmt)$. Здесь M обозначает документ, к которому прикрепляют подпись.

Процесс (4): Алгоритм Sig генерации подписи вычисляет $Rsp_i<-Select\ (Ch_i, a_i)$.

Процесс (5): Алгоритм Sig генерации подписи задает $(Cmt, Rsp_1, \ldots, Rsp_N)$ в качестве подписи.

Алгоритм Ver верификации подписи

Далее будет описана структура алгоритма Ver верификации подписи. Алгоритм Ver верификации подписи содержит следующие процессы с (1) по (3),

Процесс (1): Алгоритм Ver верификации подписи вычисляет $(Ch_1, \ldots, ch_N)<-H(M, Cmt)$.

Процесс (2): Алгоритм Ver верификации подписи генерирует $c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$ с использованием $Ch_1, \ldots, Ch_N$ и $Rsp_1, \ldots, Rsp_N$.

Процесс (3): Алгоритм Ver верификации подписи проверяет $Cmt=H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$ с использованием воспроизведенных $c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$.

Как описано выше, если совместить доказывающую сторону в модели алгоритма аутентификации с открытым ключом с подписывающей стороной в случае алгоритма цифровой подписи, алгоритм аутентификации с открытым ключом может быть модифицирован и преобразован в алгоритм цифровой подписи.

4-2: Модификация 5-проходного алгоритма аутентификации с открытым ключом и преобразование его в алгоритм цифровой подписи (Фиг. 11)

Далее будет рассмотрен способ модификации 5-проходного алгоритма аутентификации с открытым ключом для преобразования его в алгоритм цифровой подписи.

На Фиг. 11 показан эффективный алгоритм (например, см. Фиг. 8), относящийся к 5-проходной схеме с интерактивностью пять раз и шестью операциями, т.е. с операции #1 по операцию #6.

Операция #1 содержит процесс (1) генерации $a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i})$ для i=1-N и процесс (2) вычисления $Cmt<-H(c_{01}, c_{11}, \ldots, c_{0N}, c_{1N})$. Параметр Cmt, генерируемый алгоритмом P доказывающей стороны в ходе операции #1, посылают алгоритму V проверяющей стороны.

Операция #2 содержит процедуру выбора параметра $Ch_{A1}, \ldots, Ch_{AN}$. Параметр $Ch_{A1}, \ldots, Ch_{AN}$, выбранный в ходе операции #2 посредством алгоритма V проверяющей стороны, посылают алгоритму P доказывающей стороны.

Операция #3 содержит процедуру генерации $b_i=(t_{1i}, e_{1i})$ и процедуру генерации $d=H(t_{11}, e_{11}, \ldots, t_{1N}, e_{1N})$ для i=1-N. Здесь параметры d, генерируемые алгоритмом P доказывающей стороны в ходе операции #3, посылают алгоритму V проверяющей стороны.

Операция #4 содержит процедуру выбора параметра $Ch_{B1}, \ldots, Ch_{BN}$. Параметры $Ch_{B1}, \ldots, Ch_{BN}$, выбранные в ходе операции #4 посредством алгоритма V проверяющей стороны, посылают алгоритму P доказывающей стороны.

Операция #5 содержит процедуру генерации ответов $Rsp_1, \ldots, Rsp_N$ с использованием $Ch_{B1}, \ldots, Ch_{BN}, a_1, \ldots, a_N, b_1, \ldots, b_N$. Этот процесс выражен в виде Rsp_i<-Select (Ch_{Bi}, a_i, b_i). Ответы $Rsp_1, \ldots, Rsp_N$, генерируемые алгоритмом P доказывающей стороны в ходе операции #5, посылают алгоритму V проверяющей стороны.

Операция #6 содержит процесс воспроизведения $c_{01}, c_{11}, \ldots, c_{0N}, c_{1N}, t_{11}, e_{11}, \ldots, t_{1N}, e_{1N}$ с использованием $Ch_{A1}, \ldots, Ch_{AN}, Ch_{B1}, \ldots, Ch_{BN}, Rsp_1, \ldots, Rsp_N$, процесс верификации $Cmt=H(c_{01}, c_{11}, \ldots, c_{0N}, c_{1N})$ с использованием воспроизведенных $c_{01}, c_{11}, \ldots, c_{0N}, c_{1N}$ и процесс верификации $d=H(t_{11}, e_{11}, \ldots, t_{1N}, e_{1N})$.

Алгоритм аутентификации с открытым ключом, представленный в виде операций с операции #1 по операцию #6 модифицирован для преобразования в алгоритм Sig генерации подписи и алгоритм Ver верификации подписи, изображенные на Фиг. 11.

(Алгоритм Sig генерации подписи)

Сначала будет описана структура алгоритма Sig генерации подписи. Алгоритм Sig генерации подписи содержит процессы с (1) по (7).

Процесс (1): Алгоритм Sig генерации подписи генерирует $a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i})$.

Процесс (2): Алгоритм Sig генерации подписи вычисляет Cmt<-$H(c_{01}, c_{11}, \ldots, c_{0N}, c_{1N})$.

Процесс (3): Алгоритм Sig генерации подписи вычисляет $(Ch_{A1}, \ldots, Ch_{AN})$<-H(M, Cmt). Здесь M обозначает документ, к которому прикрепляют подпись.

Процесс (4): Алгоритм Sig генерации подписи осуществляет генерацию $b_i=(t_{1i}, e_{1i})$ для i=1-N. Далее, алгоритм Sig генерации подписи вычисляет $d=H(t_{11}, e_{11}, \ldots, t_{1N}, e_{1N})$.

Процесс (5): Алгоритм Sig генерации подписи вычисляет $(Ch_{B1}, \ldots, Ch_{BN})$<-H(M, Cmt, $Ch_{A1}, \ldots, Ch_{AN}$, d). Кроме того можно осуществить модификацию для преобразования в $(Ch_{B1}, \ldots, Ch_{BN})$<-$H(Ch_{A1}, \ldots, Ch_{AN}, d)$.

Процесс (6): Алгоритм Sig генерации подписи вычисляет Rsp_i<-Select (Ch_{Bi}, a_i, b_i).

Процесс (7): Алгоритм Sig генерации подписи задает $(Cmt, d, Rsp_1, \ldots, Rsp_N)$ в качестве цифровой подписи.

Алгоритм Ver верификации подписи

Далее будет описана структура алгоритма Ver верификации подписи. Алгоритм Ver верификации подписи содержит следующие процессы с (1) по (4).

Процесс (1): Алгоритм Ver верификации подписи вычисляет $(Ch_{A1}, \ldots, Ch_{AN})$=H(M, Cmt).

Процесс (2): Алгоритм Ver верификации подписи вычисляет $(Ch_{B1}, \ldots, Ch_{BN})$=H(M, Cmt, $Ch_{A1}, \ldots, Ch_{AN}$, d). Когда осуществляется модификация для преобразования в $(Ch_{B1}, \ldots, Ch_{BN})$=$H(Ch_{A1}, \ldots, Ch_{AN}, d)$ в ходе выполнения процесса (5), осуществляемого

алгоритмом Ver верификации подписи, этот алгоритм Ver верификации подписи вычисляет $(Ch_{B1}, \ldots, Ch_{BN})=H(Ch_{A1}, \ldots, Ch_{AN}, d)$.

Процесс (3): Алгоритм Ver верификации подписи генерирует $t_{11}, e_{11}, \ldots, t_{1N}, e_{1N}, c_{01}, c_{11}, \ldots, c_{0N}, c_{1N}$ с использованием $Ch_{A1}, \ldots, Ch_{AN}, Ch_{B1}, \ldots, Ch_{BN}, Rsp_1, \ldots, Rsp_N$.

Процесс (4): Алгоритм Ver верификации подписи проверяет $Cmt=H(c_{01}, c_{11}, \ldots, c_{0N}, c_{1N})$ с использованием воспроизведенных $c_{01}, c_{11}, \ldots, c_{0N}, c_{1N}$, и $d=H(t_{11}, e_{11}, \ldots, t_{1N}, e_{1N})$.

Как описано выше, если совместить доказывающую сторону в модели алгоритма аутентификации с открытым ключом с подписывающей стороной в случае алгоритма цифровой подписи, алгоритм аутентификации с открытым ключом может быть модифицирован и преобразован в алгоритм цифровой подписи.

5: Способ уменьшения объема памяти, необходимого для верификации подписи

В частности, согласно описанному выше алгоритму цифровой подписи процедура верификации подписи выполнялась после того, как алгоритм Ver верификации подписи получит все цифровые подписи. Однако в случае описанного выше алгоритма цифровой подписи объем данных цифровой подписи относительно велик. По этой причине при выполнении аутентификации с использованием такого устройства, как радио идентификационное устройство (Radio Frequency IDentification (RFID)), имеющее память небольшого объема, необходимо всегда обращать внимание на свободный объем памяти, коэффициент использования памяти в процессе аутентификации или другой подобный показатель. Кроме того, при использовании устройства, обладающего недостаточно большой памятью, предполагается, что в некоторых случаях аутентификация не производится. Соответственно, авторы настоящего изобретения разработали способ уменьшения объема памяти, необходимого для верификации подписи.

5-1: Структура хэш-функции (Фиг. 12)

Сначала авторы настоящего изобретения сосредоточились на структуре хэш-функции. Во многих случаях, хэш-функция имеет структуру, группирующую входные данные в блоки и обрабатывающую эти данные в единицах блоков. Например, в случае SHA-1, хэш-функция имеет структуру, показанную на Фиг. 12. Хэш-функция, показанная на Фиг. 12, генерирует хэш-величину путем группирования заполненных входных данных M в Z блоков $m_1, \ldots, m_Z$ и работает с блоками m_j с применением заданной функции CF вместе с начальной величиной IV или промежуточной величиной CV_j, последовательно увеличивая индекс j. Таким образом, когда получена промежуточная величина CV_j, ранее использованные блоки становятся ненужными. Соответственно, на основе этих характеристик была разработана структура (далее именуемая способ уменьшения требований к памяти) для уменьшения объема памяти, требуемого для реализации алгоритма. Далее будет рассмотрен способ применения этой структуры к описанному выше алгоритму цифровой подписи.

5-2: Пример применения 3-проходного алгоритма цифровой подписи (Фиг. 14)

Сначала, будет описан способ применения указанного выше подхода для уменьшения необходимого объема памяти к алгоритму цифровой подписи по 3-проходной схеме, изображенному на Фиг. 10.

Нормальный способ установки: Фиг. 13

Обычно, как показано на Фиг. 13, алгоритм Ver верификации подписи, относящийся к описанному выше алгоритму цифровой подписи, принимает $(Cmt, Rsp_1, \ldots, Rsp_N)$, входящие в состав цифровой подписи, за один раз (S101). После этого, алгоритм Ver

верификации подписи выполняет (Ch_1, …, Ch_N)<-H(M, cmt) (S102). После этого, алгоритм Ver верификации подписи выполняет (c_{01}, c_{11}, c_{21}, …, c_{0N}, c_{1N}, c_{2N})<- Reproduce (Ch_1, …, Ch_N; Rsp_1, …, Rsp_N) (S103). После этого, алгоритм Ver верификации подписи проверяет Cmt=H(c_{01}, c_{11}, c_{21}, …, c_{0N}, c_{1N}, c_{2N}) (S104) и завершает последовательность операций, относящихся к верификации подписи.

Способ уменьшения объема памяти: Фиг. 14

В случае нормального способа установки, когда цифровые подписи принимают за один раз, как на этапе S101, изображенном на Фиг. 13, необходимо использовать память для сохранения (Rsp_1, …, Rsp_N) до завершения процедуры этапа S103. Однако, как понятно из структуры алгоритма, показанной на Фиг. 5, для воспроизведения (c_{0i}, c_{1i}, c_{2i}), осуществляемого на этапе S103, не используется никакая информация, кроме (Ch_i; Rsp_i). Кроме того, если рассматривать структуру хэш-функции, показанную на Фиг. 12, понятно, что при вычислении хэш-функции на этапе S104 осуществляют группирование и обработку в единицах блоков. Соответственно, структура, относящаяся к верификации подписи оказывается усовершенствована и преобразована в структуру, показанную на Фиг. 14.

В случае структуры, показанной на Фиг. 14, алгоритм Ver верификации подписи сначала принимает только параметр Cmt, входящий в состав цифровой подписи (S111). После этого, алгоритм Ver верификации подписи выполняет (Ch_1, …, Ch_N)<-H(M, cmt) (S112). После этого, алгоритм Ver верификации подписи последовательно выполняет процедуры этапов с S113 по S115, увеличивая i от i=1 до N.

На этапе S113 алгоритм Ver верификации подписи принимает Rsp_i (S113). После этого, алгоритм Ver верификации подписи выполняет (c_{0i}, c_{1i}, c_{2i})<- Reproduce (Ch_i; Rsp_i) с использованием принятого Rsp_i (S114). После выполнения процедуры этапа S114 параметры Ch_i и Rsp_i становятся ненужными. Соответственно, алгоритм Ver верификации подписи стирает эти параметры Ch_i и Rsp_i из памяти после выполнения процедуры этапа S114.

После этого, алгоритм Ver верификации подписи выполняет tmp_i<-H_i (tmp_{i-1}; c_{0i}, c_{1i}, c_{2i}) (этап S115). Кроме того, хэш-функдия H_i представляет собой функцию, передающую на выход промежуточную величину, генерируемую при вычислении до c_{0i}, c_{1i}, c_{2i} посредством хэш-функции H. На практике, поскольку размер входных данных функции Hj отличается в зависимости от выбранной функции, выполняют при необходимости подходящую коррекцию длины входных данных, такую как добавление битов. При использовании функции Hi, хэш-функция H выражена в виде алгоритма, содержащего процессы с (1) по (3), которые будут описаны ниже. Тогда величина tmp_N является конечной выходной величиной (хэш-величиной) хэш-функции H. На практике, на конечной стадии выполняют дополнительную операцию заполнения в соответствии со спецификацией хэш-функции.

Процесс (1): tmp_0<- нулевая строка символов

Процесс (2):

для i=1-N

tmp_i<-H_i (tmp_{i-1}; c_{0i}, c_{1i}, c_{2i})

end for

Процесс (3): передача tmp_N на выход

После выполнения процедур этапов с S113 по S115 для i=1-N, алгоритм V проверяющей стороны проверяет (S116), выполняется ли равенство Cmt=tmp_N и завершает последовательность процедур, относящихся к верификации плоскости. Как описано выше, алгоритм Ver верификации подписи стирает информацию, которая становится ненужной во время многократного выполнения процедур этапов с S113 по S115. из памяти. По этой причине объем памяти, необходимый для верификации подписи, оказывается уменьшен до минимально возможного уровня. Вследствие этого, описанная выше процедура верификации подписи может быть выполнена даже в устройстве, имеющем память лишь небольшого объема.

5-3: Пример применения 5-проходного алгоритма цифровой подписи (Фиг. 16)

Далее, будет описан способ применения указанного выше подхода для уменьшения необходимого объема памяти к алгоритму цифровой подписи по 5-проходной схеме, изображенному на Фиг. 11.

Нормальный способ установки: Фиг. 15

Обычно, как показано на Фиг. 15, алгоритм Ver верификации подписи, относящийся к описанному выше алгоритму цифровой подписи, принимает (Cmt, d, Rsp_1, …, Rsp_N), входящие в состав цифровой подписи, за один раз (S121). После этого, алгоритм Ver верификации подписи выполняет (Ch_{A1}, …, Ch_{AN})<- H(M, cmt) (S122). После этого, алгоритм Ver верификации подписи выполняет (Ch_{B1}, …, Ch_{BN})<- H(M, Cmt, Ch_{A1}, …, Ch_{AN}, d) (S123). После этого, алгоритм Ver верификации подписи выполняет (c_{01}, c_{11}, …, c_{0N}, c_{1N}, d_{11}, e_{11}, …, d_{1N}, e_{1N})<- Reproduce (Ch_{A1}, …, Ch_{AN}, Ch_{B1}, …, Ch_{BN}; Rsp_1, …, Rsp_N) (S124). После этого, алгоритм Ver верификации подписи проверяет Cmt=H(c_{01}, c_{11}, …, c_{0N}, c_{1N}) и d=H(d_{11}, e_{11}, …, d_{1N}, e_{1N}) (S125) и завершает последовательность операций, относящихся к верификации подписи.

Способ уменьшения объема памяти: Фиг. 16

Когда цифровые подписи принимают за один раз, как на этапе S121, изображенном на Фиг. 15, необходимо использовать память для сохранения (Rsp_1, …, Rsp_N) до завершения процедуры этапа S124. Однако, как понятно из структуры алгоритма, показанной на Фиг. 8, для воспроизведения (c_{0i}, c_{1i}, d_{1i}, e_{1i}), осуществляемого на этапе S124, не используется никакая информация, кроме (Ch_{Ai}, Ch_{Bi}; Rsp_i). Кроме того, если рассматривать структуру хэш-функции, показанную на Фиг. 12, понятно, что при вычислении хэш-функции на этапе S125 осуществляют группирование и обработку в единицах блоков. Соответственно, структура, относящаяся к верификации подписи. оказывается усовершенствована и преобразована в структуру, показанную на Фиг. 16.

В случае структуры, показанной на Фиг. 16, алгоритм Ver верификации подписи сначала принимает только параметр Cmt, входящий в состав цифровой подписи (S131). После этого, алгоритм Ver верификации подписи выполняет (Ch_{A1}, …, Ch_{AN})<- H(M, cmt) (S132).

После этого, алгоритм Ver верификации подписи принимает параметр d (S133). После этого, алгоритм Ver верификации подписи выполняет (Ch_{B1}, …, Ch_{BN})<- H(M, Cmt, Ch_{A1}, …, Ch_{AN}, d) с использованием принятого параметра d (S134). После выполнения процедуры этапа S134 параметр d становится ненужным. Соответственно, алгоритм Ver верификации подписи стирает параметр d из памяти после выполнения процедуры этапа S134. После этого, алгоритм Ver верификации подписи последовательно выполняет процедуры этапов с S135 по S137, увеличивая i от i=1 до N.

На этапе S135 алгоритм Ver верификации подписи принимает Rsp_i (S135). После этого, алгоритм Ver верификации подписи выполняет (c_{0i}, c_{1i}, t_{1i}, e_{1i})<- Reproduce (Ch_{Ai}, Ch_{Bi}; Rsp_i) с использованием принятого Rsp_i (S136). После выполнения процедуры этапа S133 параметры Ch_{Ai}, Ch_{Bi}, и Rsp_i становятся ненужными. Соответственно, алгоритм Ver верификации подписи стирает эти параметры Ch_{Ai}, Ch_{Bi} и Rspi из памяти после выполнения процедуры этапа S136.

После этого, алгоритм Ver верификации подписи выполняет tmp_i<- H_i (tmp_{i-1}; c_{0i}, c_{1i}) и tmp_i'<- Hi (tmp_{i-1}'; t_{1i}, e_{1i}) (этап S137). После выполнения процедур этапов с S135 по S137 для i=1-N, алгоритм V проверяющей стороны проверяет (S138), выполняются ли равенства $Cmt=tmp_N$ и $d=tmp_N'$ и завершает последовательность процедур, относящихся к верификации плоскости. Как описано выше, алгоритм Ver верификации подписи стирает информацию, которая становится ненужной во время многократного выполнения процедур этапов с S135 по S137, из памяти. По этой причине объем памяти, необходимый для верификации подписи, оказывается уменьшен до минимально возможного уровня. Вследствие этого, описанная выше процедура верификации подписи может быть выполнена даже в устройстве, имеющем память лишь небольшого объема.

Выше были описаны способы уменьшения объема памяти, необходимого для верификации подписи.

6: Способ выделения последовательности троичных случайных чисел из последовательности двоичных случайных чисел

В частности, возможна ситуация, когда алгоритм аутентификации с открытым ключом по 3-проходной схеме генерирует N или более троичных случайных чисел с равномерным распределением. Однако высококачественный генератор случайных чисел, способный генерировать троичные случайные числа с равномерным распределением, не являются обычными. По этой причине необходимо рассмотреть способ генерации троичных случайных чисел с равномерным распределением с использованием высококачественного генератора случайных чисел, который генерирует двоичные случайные числа с равномерным распределением. Соответственно, авторы настоящего изобретения разработали способы эффективной генерации троичных случайных чисел с равномерным распределением на основе двоичных случайных чисел с равномерным распределением. Далее, эти способы будут описаны подробно. В последующем описании предполагается, что одно число, представленное по основанию 1 (где 1 равно 2 или 3), будет считаться 1 символом.

6-1: Способ #1 выделения (2-битовое группирование) (Фиг. 17)

Сначала, способ (далее именуемый способ #1 выделения) группирования двоичных чисел из M бит по два бита каждое и выделения троичных чисел будет рассмотрен со ссылками на Фиг. 17. Как показано Фиг. 17, когда строку случайных чисел в двоичном представлении группируют по два бита каждое, можно получить M/2 2-битовых двоичных чисел. Например, если "00" соответствует троичному числу "0", "01" соответствует троичному числу "1" и "10" соответствует троичному числу "2", из случайного числа, представленного в двоичном виде в единицах по 2 бит, можно получить строку троичных случайных чисел. Однако 2-битовая величина "11" исключена. Иными словами, способ #1 выделения представляет способ выделения 3^1 элементов, выраженных 1 троичным символом, и 2^2 элементов, выраженных 2 двоичными символами. Таким образом, вероятность P_1 того, что N или более троичных чисел выделить невозможно представлена следующей формулой (30).

Математическое выражение 16

$$P_1 = \sum_{M/2-N<i\le M/2} {}_{M/2}C_i (1/4)^i (3/4)^{M/2-i}$$

... (30)

6-2: Способ #2 выделения (нет группирования) (Фиг. 18)

Затем, способ (далее именуемый способ #2 выделения) выделения случайных чисел из L троичных символов с использованием случайных чисел из M двоичных символов без группирования будет рассмотрен со ссылками на Фиг. 18. Здесь, L максимальное целое число, удовлетворяющее соотношению $3^L \le 2^M$. Имеется 2^M чисел, которые могут быть выражены посредством M двоичных символов. С другой стороны, имеется только 3^L чисел, которые могут быть выражены посредством L троичных символов. По этой причине, из 2^M чисел, которые могут быть выражены посредством M двоичных символов, 2^M-3^L чисел не используются в качестве случайных чисел в троичном представлении. Таким образом, вероятность P_2 того, что N или более троичных чисел выделить невозможно, представлена следующей формулой (31).

$$P_2 = 1 - 3^L / 2^M$$

... (31)

6-3: Способ #3 выделения (k-битовое группирование) (Фиг. 19) Описанный выше способ #1 выделения представляет собой способ группирования строки случайных чисел в двоичном представлении для получения совокупности групп минимального размера. С другой стороны, описанный выше способ #2 выделения представляет собой способ группирования строки случайных чисел в двоичном представлении для получения совокупности групп максимального размера (поскольку рассматривается M-битовое группирование). Как понятно из приведенных выше формул (30) и (31), вероятность того, что N или более троичных чисел будет невозможно выделить, различна в зависимости от длины группирования. Кроме того, когда строку случайных чисел из M двоичных символов группируют в единичные блоки по k бит, как показано на Фиг. 19, вероятность P_3 того, что N или более троичных чисел выделить невозможно, представлена следующей формулой (32).

$$P_3 = \sum_{M/k-N/L<i\le M/k} {}_{M/k}C_i (1-3^L/2^k)^i (3^L/2^k)^{M/k-i}$$

... (32)

Таким образом, вероятность P_3 того, что N или более троичных чисел выделить невозможно, можно минимизировать, и можно более эффективно выделить строку случайных чисел в троичном представлении. Например, если M=512 и N=140, вероятность P_3 оказывается минимальной при k=8.

6-3-1: Базовая структура (Фиг. 20)

Здесь порядок процедуры выделения строки случайных чисел, содержащей L троичных символов, из строки случайных чисел, содержащей M двоичных символов,

будет рассмотрен со ссылками на Фиг. 20. Как показано на Фиг. 20, сначала генерируют строку случайных чисел из M двоичных символов (S201). После этого, строку случайных чисел из M двоичных символов группируют в единичные блоки по k бит (S202). После этого, выделяют битовую строку, удовлетворяющую условию $X_{2k} \leq 3^L$, из битовых строк X_{2k}, сгруппированных в единичные блоки по k бит (S203). После этого, выделенную битовую строку передают на выход в троичном представлении (S204) и выполнение последовательности процедур завершается.

6-3-2: Дополнительный способ выделения (Фиг. 21)

Вычисляя длину k группирования, при которой вероятность P_3, выраженная приведенной выше формулой (15), является минимальной, и, выполняя алгоритм, показанный на Фиг. 20, можно эффективно выделить строку случайных чисел в троичном представлении из строки случайных чисел в двоичном представлении. Однако авторы настоящего изобретения разработали способ выделения строки случайных чисел в троичном представлении более эффективно, сосредоточившись на том факте, что битовая строка, удовлетворяющая соотношению $X_{2k} > 3^L$, не используется на этапе S204, показанном на Фиг. 20. Этот способ будет описан ниже со ссылками на Фиг. 21.

Этот способ представляет собой способ выделения строки символов в троичном представлении с использованием битовых строк, не выделенных на этапе S204, показанном на Фиг. 20. Как показано на Фиг. 21, сначала выделяют множество битовых строк, не выделенных на этапе S204, показанном на Фиг. 20, и удовлетворяющих соотношению $X;k>3^{1'}$ (например, когда множество битовых строк выражено как $y_1 y_2 \ldots y_N$, и каждая индивидуальная битовая строка y_i удовлетворяет соотношению $3^L \leq y_i < 2^k$) (S211). После этого, вычитают 3^L из каждой выделенной битовой строки y; и вычисляют множество новых битовых строк (например, когда множество новых битовых строк выражено как $z_1 z_2 \ldots z_{N'}$, и каждая индивидуальная битовая строка $z_i = y_i - 3^L$ удовлетворяет соотношению $0 \leq z_i < 2^k - 3^L$) (S212).

После этого, выделяют битовую строку X, удовлетворяющую условию $X < 3^L$, из множества новых битовых строк (S213). Здесь, L′ максимальное целое число, удовлетворяющее соотношению $3^{L'} \leq 2^k - 3^L$. После этого, битовую строку, выделенную на этапе S213, передают на выход в троичном представлении (S214) и выполнение последовательности процедур завершается. Применяя этот алгоритм, можно выделить L′ новых троичных чисел с вероятностью $3^{L'}/(2^k - 3^L)$. Кроме того, рекурсивно используя этот способ, можно выделить больше троичных чисел. Иными словами, на этапе S213 можно аналогичным образом выделить троичные числа из битовых строк, удовлетворяющих соотношению $X \geq 3^{L'}$.

Выше был описан способ эффективной генерации троичных случайных чисел с равномерным распределением на основе двоичных случайных чисел с равномерным распределением.

7: Способ эффективной подстановки коэффициентов в многочлены от нескольких переменных

Однако выше не был специально описан способ совместного использования многочленов от нескольких переменных между доказывающей стороной (или подписывающей стороной) и проверяющей стороной. Можно представить себе способ

совместного использования многочленов, в состав которого входит способ совместного использования начального числа, применяемого в момент генерации коэффициентов (случайных чисел) для многочленов от нескольких переменных, между доказывающей стороной и проверяющей стороной. Однако многочлены от нескольких переменных не могут быть использованы совместно, если последовательность, посредством которой генерируют случайные числа с применением совместно используемого начального числа, не используется совместно доказывающей стороной и проверяющей стороной.

7-1: Базовое определение

Соответственно, выполняют базовое определение последовательности, посредством которой строку случайных чисел, генерируемую с применением начального числа, совместно используемого доказывающей стороной (или подписывающей стороной) и проверяющей стороной, применяют к многочленам от нескольких переменных. Тогда, если используются многочлены от нескольких переменных, строку случайных чисел применяют к таким многочленам от нескольких переменных в соответствии с этим базовым определением. При применении этого способа многочлены от нескольких переменных совместно используются доказывающей стороной (или подписывающей стороной) и проверяющей стороной.

7-2: Структурирование данных

Однако число коэффициентов в многочленах от нескольких переменных является значительным. Когда один коэффициент выражен в единицах по 1 бит, данные в объеме по меньшей мере несколько десятков тысяч бит необходимы для представления многочлена от нескольких переменных. По этой причине нагрузка при подстановке чисел для коэффициентов многочлена от нескольких переменных очень велика. Соответственно, авторы настоящего изобретения разработали способы (способы #1 и #2 структурирования) для реализации эффективности процесса подстановки чисел для коэффициентов путем структурирования коэффициентов многочлена от нескольких переменных в виде совокупности заданных единиц (единичных блоков). Кроме того, авторы настоящего изобретения разработали способ (способ #3 структурирования) для повышения эффективности обработки данных, когда процедура подстановки осуществляется несколько раз применительно к коэффициентам одного и того же многочлена от нескольких переменных. Ниже эти способы будут описаны подробно.

7-2-1: Способ #1 структурирования (Фиг. 22-26)

Сначала будет описан способ #1 структурирования. Как показано на Фиг. 22 и 23, способ #1 структурирования представляет собой способ сбора коэффициентов членов одного рода, входящих в состав многочлена от нескольких переменных, в качестве одной структуры данных. В-примере, показанном на Фиг. 22, коэффициенты с a_{1IJ} по a_{MIJ} многочлена F от нескольких переменных собирают в качестве структуры A данных, а коэффициенты с b_{1I} по b_{MI} собирают в качестве структуры B данных. Кроме того, как показано на Фиг. 23, этот же способ применим к многочлену G от нескольких переменных. В этом случае, коэффициенты с $(a_{1IJ}+a_{1JI})$ по $(a_{MIJ}+a_{MJI})$ собирают в качестве структур данных.

Когда способ #1 структурирования не применяется, вычисление многочлена F от нескольких переменных, содержащего M многочленов от N переменных каждый осуществляется посредством алгоритма, показанного в (примере 1). В случае (примера 1), необходимо выполнять 1-битовую операцию "И" (AND) (&) $2\times N\times(N+1)\times M/2$ раз. Кроме того, необходимо выполнить 1-битовую операцию "исключающее-ИЛИ" (XOR) (^) $1\times N\times(N+1)\times M/2$ раз. Здесь предполагается, что входные данные x содержат N бит.

(пример 1)

```
input x;
for L = 1 to M
    for I = 1 to N
        for J = I to N
            [L^th bit of y] ^ = [a_LIJ] & [I^th bit of x] & [J^th bit of x];
        end for
    end for
end for
output y;
```

С другой стороны, как показано на Фиг. 22, когда коэффициенты структурированы, а генерируемые случайные числа применяют последовательно по несколько за раз в качестве коэффициентов многочлена F от нескольких переменных, алгоритм подстановки коэффициентов выражен как в (примере 2). В случае (примера 2), выполняют M-битовую операцию "И" (AND) (&) только 2×N×(N+1)/2 раз и выполняют M-битовую операцию "исключающее ИЛИ" (XOR) (^) только N×(N+1)/2 раз. Кроме того, в каждом цикле генерируют $a_{(1-M)IJ}$. Коэффициенты могут быть использованы в обратном порядке. Например, когда цикл выполняется N(N-1)/2 раз, можно не генерировать коэффициенты [$a_{(1-M)IJ}$] каждый раз, а формировать их только однажды на каждые M раз. Кроме того, в цикле из M раз можно использовать коэффициенты [$a_{(1-M)IJ}$], поворачивая их бит за битом.

(пример 2)

```
input x;
for I = 1 to N
    for J = I to N
        [1st to M^th bits of y] ^ = [a_(1 to M)IJ] & [I^th bit of x] & [J^th bit
of x];
        end for
    end for
output y;
```

Как показано Фиг. 22, когда коэффициенты структурированы, промежуточные результаты, получаемые в результате применения этих коэффициентов к многочлену F от нескольких переменных, могут быть сохранены в таблице. В этом случае применяется следующий алгоритм, представленный как в (примере 3). Кроме того, $a_{IJ}[x_1, \ldots, x_k]\,[z_1, \ldots, z_k]=(a_{(k(I-1)+1)(k(J-1)+1)}\&x_1\&z_1) \,\hat{}\ldots\hat{}\, (a_{(k(I-1)+1)(k(J-1)+k)}\&x_1\&z_k) \,\hat{}\ldots\hat{}\, (a_{(k(I-1)+k)(k(J-1)+1)}\&x_k\&z_1) \,\hat{}\ldots\hat{}\, (a_{(k(I-1)+k)(k(J-1)+k)}\&x_k\&z_k)$ сохраняют в массивах с a_{IJ} [0] [0] по a_{IJ} [2^k-1], соответственно. В случае (примера 3), выполняют M-битовую операцию "исключающее ИЛИ" (XOR) (^) только (N/k)(N/k+1)/2 раз. Однако необходимый объем памяти равен произведению $2^{2k}/k^2$ на объем памяти, требуемый для алгоритма из (примера 2).

Например, когда k=1, M-битовую операцию "исключающее-ИЛИ" (XOR) вьшолняют 120*119/2=7140 раз, необходимый объем памяти в 2^2=4 раз больше объема памяти для (примера 2), а число циклов не изменяется. Кроме того, когда k=2, M-битовую операцию

"исключающее-ИЛИ" (XOR) вьшолняют 60*59/2=1770 раз, необходимый объем памяти в 2^4=4 раз больше, а число циклов составляет 1/4. Когда k=4, M-битовую операцию "исключающее-ИЛИ" (XOR) вьшолняют 30*29/2=435 раз, необходимый объем памяти в $2^8/4^2$=16 раз больше, а число циклов составляет 1/16,4. Когда k=6, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 20*19/2=190 раз, необходимый объем памяти в $2^{12}/6^2$=114 раз больше, а число циклов составляет 1/37,6. Когда k=8, M-битовую операцию "исключающее-ИЛИ" (XOR) вьшолняют 15*14/2=135 раз, необходимый объем памяти в $2^{16}/8^2$=1024 раз больше, а число циклов составляет 1/52,9. (пример 3)

```
input x;
for I = 1 to N/k
    for J = I to N/k
        [1st to Mth bit of y] ^ = [a(1 to M)IJ [k(I − 1) + 1th to kth bits of x] [k(J − 1) + 1th to kth bits of x];
        end for
    end for
output y;
```

Способ, описанный в (примере 3) может содержать вычисление заранее величин F_{IJ} (…) согласно следующей формуле (33) и сохранение результатов в виде массива.

$$F(x_1,\cdots,x_n)=\sum_{I,J}F_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},x_{k(J-1)+1},\cdots,x_{k(J-1)+k})=y \quad \cdots \quad (33)$$

Здесь, $F_{IJ}(x_{k(I-1)+1}, \ldots, x_{k(I-1)+k}, x_{k(J-1)+1}, \ldots, x_{k(J-1)+k})$ обозначает часть, определяемую посредством $[x_{k(I-1)+1}, \ldots, x_{k(I-1)+k}]$ и $[x_{k(J-1)+1}, \ldots, x_{k(J-1)+k}]$ в $F(x_1, \ldots, x_n)$.

Приведенные здесь конкретные алгоритмы представляют собой примеры случаев применения способа #1 структурирования к многочлену F от нескольких переменных. В такой структуре можно ожидать, что процесс выполнения этого алгоритма будет проходить с высокой скоростью.

(Применение к G)

Выше алгоритм для вычисления многочлена F от нескольких переменных с применением способа #1 структурирования был описан со ссылками на Фиг. 22. С другой стороны, каждый элемент многочлена G от нескольких переменных выражен в квадратичной форме. Поэтому, как показано на Фиг. 23, описанный выше способ #1 структурирования аналогичным образом применим к вычислению многочлена G от нескольких переменных без изменений.

Например, когда описанный выше способ #1 структурирования не применяется, алгоритм для вычисления многочлена G выражен в следующем (примере 1′). Здесь предполагается, что входные данные x и y содержат каждые по N бит.

(пример 1′)

```
input x, y;
for L = 1 to M
    for I = 1 to N
        for J = I to N
            [L^th bit of z] ^ = [a_LIJ + a_LJI] & [I^th bit of x] & [J^th bit of y];
        end for
    end for
end for
output z;
```

Когда описанный выше способ #1 структурирования применяется в описанном выше (примере 1′), алгоритм для вычисления многочлена G выражен в следующем (примере 2′).

(пример 2′)

```
input x, y;
for I = 1 to N
    for J = 1 to N
        [1^st to M^th bits of z] ^ = [a_(1 to M)IJ + a_(1 to M)JI] & [I^th bit of x] & [J^th
bit of y];
    end for
end for
output z;
```

В случае способа сохранения промежуточного результата, полученного путем применения коэффициентов многочлена G от нескольких переменных, в таблице, алгоритм для вычисления многочлена G от нескольких переменных выражен в следующем (пример 3′) в соответствии с указанным выше (пример 3). Кроме того, $a_{IJ}[x_1, \ldots, x_k]\,[y_1, \ldots, y_k]=(a_{(k(I-1)+1)(k(J-1)+1)}\&x_1\&y_1) \wedge \ldots \wedge (a_{(k(I-1)+1)(k(J-1)+k)}\&x_1\&y_k) \wedge \ldots \wedge (a_{(k(I-1)+k)(k(J-1)+1)}\&x_k\&y_1) \wedge \ldots \wedge (a_{(k(I-1)+k)(k(J-1)+k)}\&x_k\&y_k)$ сохраняют в массивах с $a_{IJ}\,[0]\,[0]$ по $a_{IJ}\,[2^k-1]\,[2^k-1]$, соответственно.

В случае (примера 3′), выполняют M-битовую операцию "исключающее ИЛИ" (XOR) ($\wedge$) только $(N/k)^2$ раз. Однако необходимый объем памяти равен произведению $2^{2k}/k^2$ на объем памяти, требуемый для алгоритма из (примера 2′).

Например, когда k=1, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют $120^2=14400$ раз, необходимый объем памяти в $2^2=4$ раз больше объема памяти для (примерп 2′), а число циклов не изменяется. Кроме того, когда k=2, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют $60^2=3600$ раз, необходимый объем памяти в $2^4=4$ раз больше, а число циклов составляет 1/4. Когда k=4, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют $30^2=900$ раз, необходимый объем памяти в $2^8/4^2=16$ раз больше, а число циклов составляет 1/16. Когда k=6, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют $20^2=400$ раз, необходимый объем памяти в $2^{12}/6^2=114$ раз больше, а число циклов составляет 1/36. Когда k=8, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют $15^2=225$ раз, необходимый объем памяти в

$2^{16}/8^2$=1024 раз больше, а число циклов составляет 1/64.

(пример 3′)

```
input x, y;
for I = 1 to N/k
    for J = I to N/k
        [1st to Mth bits of z] ^ = a(1 to M)IJ [k(I − 1) + 1th to kth bits of x] [k(J −
1) + 1th to kth bits of y];
    end for
end for
output z;
```

Способ, описанный в (примере 3′) может, в частности, представлять собой способ вычисления заранее величин $G_{IJ}(\ldots)$ согласно следующей формуле (34) и сохранение результатов в виде массива.

$$G(x_1,\cdots,x_n)=\sum_{I,J}G_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},y_{k(J-1)+1},\cdots,y_{k(J-1)+k})=z \quad \cdots (34)$$

Здесь, $G_{IJ}(x_{k(I-1)+1}, \ldots, x_{k(I-1)+k}, x_{k(J-1)+1}, \ldots, x_{k(J-1)+k})$ обозначает часть, определяемую посредством $[x_{k(I-1)+1}, \ldots, x_{k(I-1)+k}]$ и $[x_{k(J-1)}+1, \ldots, x_{k(J-1)+k}]$ в $G(x_1, \ldots, x_n, y_1, \ldots, y_N)$.

Конкретные структуры алгоритма, относящегося к способу #1 структурирования, были описаны на примере случая вычисления многочлена G от нескольких переменных. В такой структуре можно ожидать, что процесс выполнения этого алгоритма будет проходить с высокой скоростью.

Применение к кубическому многочлену F от нескольких переменных Способы структурирования коэффициентов квадратичного многочлена от нескольких переменных были рассмотрены выше в качестве конкретных примеров способа #1 структурирования. Аналогично можно структурировать также коэффициенты кубического многочлена от нескольких переменных. В примере, показанном на Фиг. 24, коэффициенты с a_{1IJL} по a_{MIJL} собирают в качестве структуры A данных, а коэффициенты с b_{1IJ} по b_{MIJ} собирают в качестве структуры B данных и коэффициенты с c_{1I} по c_{MI} собирают в качестве структуры C данных. Благодаря структурированию коэффициентов таким способом можно эффективно вычислять многочлен от нескольких переменных.

Например, будут конкретно рассмотрены вычисления, относящиеся к структуре A данных. Когда коэффициенты не структурированы, алгоритм для вычислений, относящихся к структуре A данных, выражен в следующем (примере 4).

(пример 4)

```
input x;
for Q = I to M
    for I = 1 to N
        for J = I to N
            for L = J to N
                [Q^th bit of y] ^ = [a_QIJL] & [I^th bit of x] & [J^th bit of x]
& [L^th bit of x]
            end for
        end for
    end for
end for
output y;
```

С другой стороны, когда коэффициенты структурированы, алгоритм выражен в следующем (примере 5). В случае (примера 5), M-битовую операцию "И" (AND) (&) выполняют только $2\times N\times(N+1)\times(N+2)/6$ раз и M-битовую операцию "исключающее ИЛИ" (XOR) (^) выполняют только $1\times N\times(N+1)\times(N+2)/6$ раз. Кроме того, в каждом цикле генерируют $a_{(1\text{-}M)IJL}$. Коэффициенты могут быть использованы в обратном порядке. Например, когда цикл выполняется, можно не генерировать коэффициенты $[a_{(1\text{-}M)IJL}]$ каждый раз, а формировать их только однажды на каждые M раз. Кроме того, в цикле из M раз можно использовать коэффициенты $[a_{(1\text{-}M)IJ}]$, поворачивая их бит за битом.

(пример 5)

```
input x;
for I = 1 to N
    for J = I to N
        for L = J to N
            [1^st to M^th bits of y] ^ = [a_(1 to M)IJL] & [I^th bit of x] & [J^th bit
of x] & [L^th bit of x]
        end for
    end for
end for
output y;
```

Когда коэффициенты структурированы, промежуточные результаты, получаемые в результате применения этих коэффициентов к многочлену F от нескольких переменных, могут быть сохранены в таблице. В этом случае применяется следующий алгоритм, представленный как в (примере 6). Кроме того, $a_{IJL}[x_1, \ldots, x_k][z_1, \ldots, z_k][w_1, \ldots, w_k]=\sum_{1\le\alpha,\beta,\gamma\le k}(a_{(k(I-1)+\alpha)(k(J-1)+\beta)(k(L-1)+\gamma)}\ \&x_\alpha\ \&z_\beta\ \&w_\gamma)$ сохраняют в массивах с a_{IJL} [0] [0] [0] по a_{IJL} [2^k-1] [2^k-1] [2^k-1], соответственно. В случае (примера 6), выполняют M-битовую операцию "исключающее ИЛИ" (XOR) (^) только $(N/k)(N/k+1)(N/k+2)/6$ раз. Однако необходимый объем памяти равен произведению $2^{2k}/k^3$ на объем памяти, требуемый для алгоритма из (примера 4).

Например, когда k=1, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 120*121*122/6=295240 раз, необходимый объем памяти в 2^3=8 раз больше объема памяти для (примера 4), а число циклов не изменяется. Кроме того, когда k=2, M-битовую

операцию "исключающее-ИЛИ" (XOR) выполняют 60*61*62/6=37820 раз, необходимый объем памяти в $2^6/8=8$ раз больше, а число циклов составляет 1/7.8. Когда k=4, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 30*31*32/6=4960 раз, необходимый объем памяти в $2^{12}/4^3=64$ раз больше, а число циклов составляет 1/59,5. Когда k=6, M-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 20*21*22/6=1540 раз, необходимый объем памяти в $2^{18}/6^3=1214$ раз больше, а число циклов составляет 1/192.

(пример 6)

```
input x;
for I = 1 to N/k
        for J = I to N/k
                for L = J to N/k
                        [1st to Mth bits of y] ^ = a(1 to M)IJL[k(I − 1) +1th to kth bits of
x] [k(J − 1) +1th to kth bits of x] [k(L − 1) +1th to kth bits of x]
                end for
        end for
end for
output y;
```

Способ, описанный в (примере 6) может, в частности, представлять собой способ вычисления заранее величин $F_{IJ}(\ldots)$ согласно следующей формуле (35) и сохранения величин в виде массива.

$$F(x_1,\cdots,x_n)=y$$
$$=\sum_{I,J}F_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},x_{k(J-1)+1},\cdots,x_{k(J-1)+k},x_{k(L-1)+1},\cdots,x_{k(L-1)+k}) \quad \cdots (35)$$

Здесь, $F_{IJ}(x_{k(I-1)+1}, \ldots, x_{k(I-1)+k}, x_{k(J-1)+1}, \ldots, x_{k(J-1)+k}, x_{k(L-1)+1}, \ldots, x_{k(L-1)+k})$ обозначает часть, определяемую посредством $[x_{k(I-1)+1}, \ldots, x_{k(I-1)+k}]$, $[x_{k(J-1)+1}, \ldots, x_{k(J-1)+k}]$, и $[x_{k(L-1)+1}, \ldots, x_{k(L-1)+k}]$ в $F(x_1, \ldots, x_n)$.

Выше были описаны случаи применения способа #1 структурирования к кубическому многочлену F от нескольких переменных. В такой структуре можно ожидать, что процесс выполнения этого алгоритма будет проходить с высокой скоростью.

(Применение к кубическим многочленам G_x и G_y от нескольких переменных) Как и в предыдущих (примере 5) и (примере 6) способ #1 структурирования также применим к многочленам G_x и G_y, как показано на Фиг. 25 и 26. Когда способ #1 структурирования не применяется, вычисления, относящиеся к структуре A данных кубического многочлена G_x от нескольких переменных, осуществляются, например, посредством следующего алгоритма (пример 4′X).

(пример 4′X)

```
input x, y;
for Q = 1 to M
    for I = 1 to N
        for J = 1 to N
            for L = 1 to N
                [Q^th bit of z] ^ = [a_QIJL + a_QILJ + a_QLJI] & [I^th bit of y]
& [J^th bit of y] & [L^th bit of x]
            end for

        end for
    end for
end for
output z;
```

Когда описанный выше способ #1 структурирования применяется в описанном выше (примере 4′X), алгоритм для вычисления многочлена G_x от нескольких переменных выражен в следующем (примере 5′X). Кроме того, в описанном выше (примере 6) промежуточные результаты, получаемые в результате применения этих коэффициентов к многочлену G_x от нескольких переменных, могут быть сохранены в таблице.

(пример 5′X)

```
input x, y;
for I = 1 to N
    for J = I to N
        for L = 1 to N
            [1st to Mth bits of z] ^ = [a_(1 to M)IJL + a_(1 to M)ILJ + a_(1 to M)LJI] &
[I^th bit of y] & [J^th bit of y] & [L^th bit of x]
        end for
    end for
end for
output z;
```

Когда способ #1 структурирования не применяется, вычисления кубического многочлена G_y от нескольких переменных, осуществляются, например, посредством следующего алгоритма из (примера 4′Y).

(пример 4′Y)

```
input x, y;
for Q = 1 to M
    for I = 1 to N
        for J = 1 to N
            for L = 1 to N
                [Q^th bit of z] ^ = [a_QIJL + a_QILJ + a_QLJI] & [I^th bit of x]
& [J^th bit of x] & [L^th bit of y]
            end for
        end for
    end for
end for

output z;
```

Когда описанный выше способ #1 структурирования применяется в описанном выше (примере 4′Y), алгоритм для вычисления многочлена G_y от нескольких переменных выражен в следующем (примере 5′Y). Кроме того, в описанном выше (примере 6) промежуточные результаты, получаемые в результате применения этих коэффициентов к многочлену G_y от нескольких переменных, могут быть сохранены в таблице.

(пример 5′Y)

```
input x, y;
for I = 1 to N
    for J = 1 to N
        for L = 1 to N
            [1^st to M^th bits of z] ^ = [a_(1 to M)IJL + a_(1 to M)ILJ + a_(1 to M)LJI] &
[I^th bit of x] & [J^th bit of x] & [L^th bit of y]
        end for
    end for
end for
output z;
```

Выше были описаны случаи применения способа #1 структурирования к кубическим многочленам G_x и G_y от нескольких переменных. В такой структуре можно ожидать, что процесс выполнения этого алгоритма будет проходить с высокой скоростью. 7-2-2: Способ #2 структурирования (Фиг. 27)

Далее будет описан способ #2 структурирования. Как показано на Фиг. 27, способ #2 структурирования представляет собой способ представления многочлена от нескольких переменных в квадратичной форме и сбора строк и столбцов этой квадратичной формы в качестве одной структуры данных. В примере, показанном на Фиг. 27, структура данных собрана в направлении строк.

Как показано на Фиг. 27, когда коэффициенты структурированы, а генерируемые случайные числа применяют последовательно по несколько за раз в качестве коэффициентов многочлена от нескольких переменных, алгоритм подстановки коэффициентов выражен как в (примере 4). В случае примера 4, выполняют N-битовую операцию "И" (AND) (&) только (N+1)×M раз, выполняют N-битовую операцию "исключающее ИЛИ" (XOR) (^) только N×M раз и выполняют операцию функции Q только M раз.

(пример 4)

```
for I = 1 to N

        T ^ = A_I & [I^th bit of x]
end for
T & = x;
output Q(T);

Q(z) {
        z = z ^ (z >> 1);
        z = z ^ (z >> 2);
        z = z ^ (z >> 4);
        z = z ^ (z >> 8);
        …
        Z = z ^ (z >> 2^Log(N));
        return z & 1;
}
```

Как показано на Фиг. 27, когда коэффициенты структурированы, промежуточные результаты, получаемые в результате применения этих коэффициентов к многочлену от нескольких переменных, могут быть сохранены в таблице. В этом случае применяется алгоритм подстановки коэффициентов, представленный как в (примере 5). Кроме того, $A_I [x_1, \ldots, x_k]=(A_{(k(I-1)+1)}\&x_1) \wedge \ldots \wedge (A_{(k(I-1)+k)}\&x_k)$ сохраняют в массивах с $A_I [0]$ по $A_I [2^k-1]$, соответственно. В случае примера 5, выполняют N-битовую операцию "исключающее ИЛИ" (XOR) ($\wedge$) только $(N/k)\times M$ раз и выполняют N-битовую операцию функции Q только M раз. Однако необходимый объем памяти равен произведению $2^k/k$ на объем памяти, требуемый для алгоритма из (примера 4).

Например, когда k=1, N-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 120 раз, необходимый объем памяти в два раза больше объема памяти для (примера 4), а число циклов не изменяется. Кроме того, когда k=4, N-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 30 раз, необходимый объем памяти в $2^4/4=4$ раз больше, а число циклов составляет 1/4. Когда k=8, N-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 15 раз, необходимый объем памяти в $2^8/8=32$ раз больше, а число циклов составляет 1/8. Когда k=16, N-битовую операцию "исключающее-ИЛИ" (XOR) выполняют 8 раз, необходимый объем памяти в $2^{16}/16=4096$ раз больше, а число циклов составляет 1/15.

(пример 5)

```
for I = 1 to N/k
        T ^ = A_I [k(I − 1) + 1^th to k(I − 1) + k^th bits of x]
```

end for

T & = x;

output Q(T);

Выше были описаны конкретные алгоритмы подстановки коэффициентов, относящиеся к способу #2 структурирования. В такой структуре можно ожидать, что процесс выполнения этого алгоритма будет проходить с высокой скоростью.

7-2-3: Способ #3 структурирования

Далее будет описан способ #3 структурирования. Способ #3 структурирования представляет собой способ последовательного выполнения процедуры N раз (где $N \geq 2$) параллельно путем выполнения сегмента «для генерации нескольких коэффициентов и выполнения процедуры подстановки нескольких коэффициентов N раз» в единичных блоках вместо того, чтобы генерировать многочлен из случайных чисел N раз и выполнять процедуру подстановки в одном и том же многочлене от нескольких переменных N раз. При использовании этого способа повышается производительность всех процессов, выполняемых N раз, в случае, когда стоимость генерации случайных чисел не является пренебрежимо малой.

Например, в алгоритме, представленном на Фиг. 5, вычисление многочленов F и G от нескольких переменных осуществляется многократно N раз, и при этом происходит обновление факторов операции #1. Соответственно, в таком вычислительном сегменте процедура вычислений конфигурирована для многократного выполнения с использованием одних и тех же коэффициентов. Когда многочлен $F(r_{0i})$ (где i=1-N) от нескольких переменных вычисляют с использованием описанного выше алгоритма из (примера 2), все из N векторов r_{0i} конфигурированы для применения к однажды сформированному массиву $[a_{IJL}]$, а затем выполняется процедура относительно следующего массива $[a_{IJL}]$. В такой конфигурации один и тот же массив $[a_{IJL}]$ коэффициентов, в конечном итоге, не генерируют N раз.

Выше был описан конкретный алгоритм подстановки коэффициентов, относящийся к способу #3 структурирования. В такой конфигурации производительность повышается для всех процессов, выполняемых N раз, в целом.

8: Пример конфигурации аппаратуры (Фиг. 28)

Каждый алгоритм, описанный выше, может быть выполнен с использованием, например, конфигурации аппаратуры устройства для обработки информации, представленного на Фиг. 28. Иными словами, выполнение каждого алгоритма может быть реализовано посредством управления аппаратурой, показанной на Фиг. 28 с использованием компьютерной программы. Кроме того, характер аппаратуры может быть произвольным - это может быть персональный компьютер, мобильный информационный терминал, такой как мобильный телефон, PHS или PDA, игровой автомат, контактный или бесконтактный кристалл интегральной схемы, контактная или бесконтактная электронная карточка с встроенной интегральной схемой или разнообразная информационная аппаратура. Здесь PHS представляет собой сокращение от Personal Handy-phone System, т.е. персональный миниатюрный радиотелефон. Кроме того, PDA представляет собой сокращение от Personal Digital Assistant, т.е. персональный цифровой помощник.

Как показано на Фиг. 28, эта аппаратура содержит главным образом процессор CPU 902, ПЗУ ROM 904, ЗУПВ RAM 906, главную шину 908 и мост 910. Кроме того, эта аппаратура содержит внешнюю шину 912, интерфейс 914, модуль 916 ввода, модуль

918 вывода, модуль 920 памяти, привод 922 носителя записи, соединительный порт 924 и модуль 926 связи. Здесь CPU представляет собой сокращение от Central Processing Unit, т.е. центральный процессор. Кроме того, ROM представляет собой сокращение от Read Only Memory, т.е. постоянное запоминающее устройство (ПЗУ). Далее, RAM представляет собой сокращение от Random Access Memory, т.е. запоминающее устройство с произвольной выборкой (ЗУПВ).

Центральный процессор CPU 902 служит арифметическим модулем или модулем управления, например, и управляет всей работой или частью работы каждого из структурных элементов на основе различных программ, записанных в ПЗУ ROM 904, в ЗУПВ RAM 906, в модуле 920 памяти или на сменном носителе 928 записи. ПЗУ ROM 904 служит для сохранения программы, подлежащей загрузке в процессор CPU 902, или данных или другой подобной информации, используемой при выполнении арифметических операций. ЗУПВ RAM 906 служит для временного или постоянного хранения программы, подлежащей загрузке в процессор CPU 902, или различных параметров или другой подобной информации, произвольно изменяющейся в ходе выполнения программы.

Эти структурные элементы соединены один с другим посредством, например, главной шины 908, способной передавать данные с высокой скоростью. Со своей стороны главная шина 908 соединена через мост 910 с внешней шиной 912, имеющей относительно низкую скорость передачи данных, например. Модуль 916 ввода представляет собой, например, мышь, клавиатуру, сенсорную панель, кнопки, переключатель или джойстик. Кроме того, модуль 916 ввода может представлять собой пульт дистанционного управления, который может передавать сигнал управления с использования инфракрасного излучения или радиоволн.

Модуль 918 вывода представляет собой, например, дисплей, такой как CRT, LCD, PDP или ELD, выходное аудио устройство, такое как громкоговоритель или наушники, принтер, мобильный телефон или факсимильный аппарат, который может визуально или акустически сообщать пользователю о полученной информации. Здесь CRT представляет собой сокращение от Cathode Ray Tube, т.е. электронно-лучевая трубка. Кроме того, LCD представляет собой сокращение от Liquid Crystal Display, т.е. жидкокристаллический дисплей. Кроме того, PDP представляет собой сокращение от Plasma Display Panel, т.е. панель плазменного дисплея. Кроме того, ELD представляет собой сокращение от Electro-Luminescence Display, т.е. электролюминесцентный дисплей.

Модуль 920 памяти представляет собой устройство для хранения разнообразных данных. Указанный модуль 920 памяти представляет собой, например, магнитное запоминающее устройство, такое как накопитель HDD, полупроводниковое запоминающее устройство, оптическое запоминающее устройство или магнитооптическое запоминающее устройство. Здесь, HDD представляет собой сокращение от Hard Disk Drive, т.е. накопитель на жестком диске.

Привод 922 носителя записи представляет собой устройство, считывающее информацию, записанную на сменном носителе 928 записи, таком как магнитный диск, оптический диск, магнитооптический диск или полупроводниковое запоминающее устройство, или записывает информацию на сменном носителе 928 записи. Этот сменный носитель 928 записи может представлять собой, например, диск DVD, диск Блю-рей, диск HD-DVD, разнообразные полупроводниковые запоминающие устройства или аналогичные компоненты. Безусловно, сменный носитель 928 записи может представлять собой, например, электронное устройство или электронную карточку, на которой смонтирован бесконтактный кристалл интегральной схемы. Здесь 1C представляет

собой сокращение от Integrated Circuit, т.е. интегральная схема.

Соединительный порт 924 представляет собой порт, такой как USB-порт, порт IEEE 1394, порт SCSI, порт RS-232C или порт для присоединения внешнего устройства 930, такого как оптический аудио терминал. Указанное внешнее устройство 930 представляет собой принтер, мобильный музыкальный плеер, цифровой фотоаппарат, цифровая видеокамера или записывающее устройство на основе интегральной схемы. Более того, USB представляет собой сокращение от Universal Serial Bus, т.е. универсальная последовательная шина. Кроме того, SCSI представляет собой сокращение от Small Computer System Interface, т.е. интерфейс малых вычислительных систем.

Модуль 926 связи представляет собой устройство связи для соединения с сетью 932 связи и может быть, например, картой связи для взаимодействия с кабельной ли радио сетью LAN, карту стандарта Bluetooth (зарегистрированная торговая марка) или WUSB, оптическим маршрутизатором связи, маршрутизатором ADSL или устройством для контактной или бесконтактной связи. Сеть 932 связи, соединяемая с модулем 926 связи, конфигурирована по принципу кабельной или беспроводной ста связи и может быть, например, сетью Интернет, домашней сетью LAN, сетью инфракрасной связи, сетью оптической связи в видимом диапазоне, сетью вещания или сетью спутниковой связи. Здесь, LAN представляет собой сокращение от Local Area Network, т.е. локальная сеть связи. Кроме того, WUSB представляет собой сокращение от Wireless USB, т.е. радио USB. Далее, ADSL представляет собой сокращение от Asymmetric Digital Subscriber Line, т.е. асимметричная цифровая абонентская линия.

9: Резюме

Наконец, далее будет вкратце изложено техническое содержание рассмотренного варианта настоящего изобретения. Изложенное здесь техническое содержание может быть применено в различных устройствах для обработки информации, таких как персональный компьютер, мобильный телефон, игровой терминал, информационный терминал, информационная аппаратура, автомобильный навигатор или другое подобное устройство. Кроме того, функции устройства для обработки информации, описанные ниже, могут быть реализованы с использованием одного устройства для обработки информации или нескольких устройств для обработки информации. Далее, устройства для хранения данных и арифметические процессоры, используемые для выполнения обработки данных в устройстве для обработки информации, описанном ниже, могут быть смонтированы в самом устройстве для обработки информации или могут быть смонтированы в каком-либо устройстве, соединенном с рассматриваемым устройством для обработки информации через сеть связи.

Функциональная конфигурация устройства для обработки информации, описанного выше, реализована следующим образом. Например, устройство для обработки информации, описанное в следующем п. (1), имеет функцию выполнения алгоритма, относящегося к эффективным алгоритмам аутентификации с открытым ключом или к эффективным алгоритмам цифровой подписи, защищенность которых базируется на трудности решения систем уравнений множества порядков с несколькими переменными.

(1) Устройство обработки информации, содержащее:

модуль генерации чисел, выполненный с возможностью генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с

несколькими переменными; и

модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом модуль назначения выполнен с возможностью группирования коэффициентов членов, имеющих идентичное сочетание переменных, из совокупности коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, и выполнения процедуры назначения в единицах групп.

(2) Устройство обработки информации, содержащее:

модуль генерирования чисел, выполненный с возможностью генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом модуль назначения выполнен с возможностью группирования матрицы коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, причем модуль назначения выполнен с возможностью выполнения процедуры назначения в единицах групп.

(3) Устройство обработки информации согласно (1) или (2), в котором модуль генерации чисел выполнен с возможностью генерации чисел по количеству коэффициентов, принадлежащих одной группе, при выполнении модулем назначения процедуры назначения в указанной одной группе.

(4) Устройство обработки информации согласно (3), в котором модуль генерации чисел выполнен с возможностью генерации одного числа при генерации чисел по количеству коэффициентов, принадлежащих одной группе, и генерации необходимого количества чисел путем поворота генерируемых чисел на каждый бит.

(5) Устройство обработки информации согласно (1) или (2), дополнительно содержащее:

модуль хранения таблицы, выполненный с возможностью хранения в виде таблицы значений, получаемых путем назначения коэффициентов члену какого-либо вида, соответствующего каждой группе, и подстановки данного числа для переменных соответствующих членов.

(6) Устройство обработки информации, содержащее:

модуль генерации чисел, выполненный с возможностью генерации чисел в количестве, идентичном количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной

функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, согласованной заранее между доказывающей стороной и проверяющей стороной.

(7) Устройство обработки информации согласно любому из (1)-(7),

в котором указанная информация является начальным числом для генерации случайного числа, а

указанная заданная функция является генератором случайных чисел, выполненным с возможностью генерации случайного числа с использованием указанного начального числа.

(8) Устройство обработки информации согласно любому из (1)-(7), содержащее:

модуль генерации сообщений, выполненный с возможностью генерации сообщения на основе указанной пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и вектора s, являющегося элементом множества K^n;

модуль передачи сообщений, выполненный с возможностью передачи указанного сообщения проверяющей стороне, хранящей указанную пару многочленов F множества порядков от нескольких переменных и векторы $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$; и

модуль передачи ответа, выполненный с возможностью передачи проверяющей стороне ответной информации, соответствующей схеме верификации, выбранной проверяющей стороной из совокупности k схем верификации, где $k\geq 3$,

при этом вектор s представляет собой секретный ключ,

указанная пара многочленов F множества порядков от нескольких переменных и вектор y являются открытыми ключами,

ответная информация представляет собой информацию, выбранную из пары случайных чисел, и указанное сообщение согласно схеме верификации, и

причем указанное сообщение представляет собой информацию, полученную путем вычислений, выполненных заранее для схемы верификации, соответствующей ответной информации, на основе открытых ключей и ответной информации.

(9) Устройство обработки информации согласно любому из (1)-(7), содержащее:

модуль хранения информации, выполненный с возможностью хранения указанной пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и векторов $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$;

модуль получения сообщений, выполненный с возможностью получения сообщения, генерируемого на основе указанной пары многочленов F множества порядков от нескольких переменных и вектора s, являющегося элементом множества K^n;

модуль передачи информации о схеме, выполненный с возможностью передачи проверяющей стороне, передавшей сообщение, информации о схеме верификации, выбранной случайным образом из совокупности k схем верификации, где $k\geq 3$;

модуль получения ответа, выполненный с возможностью получения от доказывающей стороны ответной информации, соответствующей выбранной схеме верификации; и

модуль верификации, выполненный с возможностью проверки, хранит ли доказывающая сторона вектор s, основанный на сообщении, указанной паре многочленов F множества порядков с несколькими переменными, векторов y и ответной информации,

при этом вектор s представляет собой секретный ключ,

пара многочленов F множества порядков от нескольких переменных и векторы y являются открытыми ключами, а

указанное сообщение представляет собой информацию, получаемую путем вычислений, выполненных заранее для схемы верификации, соответствующей информации ответа, на основе открытых ключей и информации ответа.

(10) Устройство обработки информации согласно любому из (1)-(7), содержащее:

модуль генерации сообщений, выполненный с возможностью генерации сообщения на основе указанной пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и вектора s, являющегося элементом множества K^n;

модуль передачи сообщений, выполненный с возможностью передачи указанного сообщения проверяющей стороне, хранящей указанную пару многочленов F множества порядков от нескольких переменных и векторы $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$;

модуль генерации промежуточной информации, выполненный с возможностью генерации третьей информации с использованием первой информации, выбранной случайным образом проверяющей стороной, и второй информации, полученной в момент генерации сообщения;

модуль передачи промежуточной информации, выполненный с возможностью передачи третьей информации проверяющей стороне; и

модуль передачи ответа, выполненный с возможностью передачи проверяющей стороне ответной информации, соответствующей схеме верификации, выбранной проверяющей стороной из совокупности k схем верификации, где $k \geq 2$,

при этом вектор s представляет собой секретный ключ,

пара многочленов F множества порядков от нескольких переменных и векторы y являются открытыми ключами,

ответная информация представляет собой информацию, выбранную из сообщений согласно указанной схеме верификации, а

указанное сообщение представляет собой информацию, полученную путем вычислений, выполненных заранее для схемы верификации, соответствующей ответной информации, на основе открытых ключей, первой информации, третьей информации и ответной информации.

(11) Устройство обработки информации согласно любому из (1)-(7), содержащее:

модуль хранения информации, выполненный с возможностью хранения пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и векторов $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$;

модуль получения сообщений, выполненный с возможностью получения сообщения, генерируемого на основе указанной пары многочленов F множества порядков от нескольких переменных и вектора s, являющегося элементом множества K^n;

модуль передачи информации, выполненный с возможностью передачи доказывающей стороне, передавшей указанное сообщение, выбранной случайным образом первой информации;

модуль получения промежуточной информации, выполненный с возможностью

получения третьей информации, генерируемой доказывающей стороной на основе указанной первой информации, и второй информации, получаемой в момент генерации сообщения;

модуль передачи информации о схеме, выполненный с возможностью передачи доказывающей стороне информации о схеме верификации, выбранной случайным образом из совокупности k схем верификации, где $k \geq 3$;

модуль получения ответа, выполненный с возможностью получения от доказывающей стороны ответной информации, соответствующей выбранной схеме верификации; и

модуль верификации, выполненный с возможностью проверки, сохраняет ли доказывающая сторона вектор s, на основе сообщения, первой информации, третьей информации, пары многочленов F множества порядков с несколькими переменными и ответной информации,

при этом вектор s представляет собой секретный ключ,

пара многочленов F множества порядков от нескольких переменных и векторы y являются открытыми ключами,

причем указанное сообщение представляет собой информацию, полученную путем вычислений, выполненных заранее для схемы верификации, соответствующей ответной информации, на основе открытых ключей, первой информации, третьей информации и ответной информации.

(12) Устройство обработки информации согласно (1) или (2), в котором при повторном выполнении алгоритма множество раз модуль генерации чисел выполнен с возможностью генерации чисел только один раз, а модуль назначения выполнен с возможностью выполнения процедуры назначения только один раз,

причем алгоритм многократно использует коэффициенты, назначаемые посредством модуля назначения.

(13) Устройство обработки информации согласно любому из (1)-(7), содержащее:

модуль генерации подписи, выполненный с возможностью генерации цифровой подписи для документа M с использованием указанной пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и ключа подписи, являющегося элементом множества K^n; и

модуль передачи подписи, выполненный с возможностью передачи цифровой подписи проверяющей стороне, хранящей пару многочленов F множества порядков от нескольких переменных и векторы y=(f1(s), …, fm(s));

(14) Способ обработки информации, содержащий:

этап генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

этап назначения сгенерированных чисел коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом на этапе назначения чисел группируют коэффициенты членов, имеющих идентичный тип сочетания переменных, из совокупности коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими

переменными, и выполняют процедуру назначения в блоках групп.

(15) Способ обработки информации, содержащий:

этап генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

этап назначения генерируемых чисел коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом на этапе назначения чисел группируют матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, и выполняют процедуру назначения в блоках групп.

(16) Способ обработки информации, содержащий:

этап генерации чисел в количестве, идентичном количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

этап назначения чисел, сгенерированных посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, определенной заранее между доказывающей стороной и проверяющей стороной.

(17) Программа, вызывающая выполнение компьютером:

функции генерации чисел для генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

функции назначения для назначения чисел, генерируемых посредством функции генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом функция назначения группирует коэффициенты членов, имеющих идентичный тип сочетания переменных, из совокупности коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, и выполняет процедуру назначения в блоках групп.

(18) Программа, вызывающая выполнение компьютером:

функции генерации чисел для генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими

переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

функции назначения для назначения чисел, генерируемых посредством функции генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом функция назначения группирует матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, при этом функция назначения выполняет процедуру назначения в блоках групп.

(19) Программа, вызывающая выполнение компьютером:

функции генератора чисел, осуществляющую генерацию чисел в количестве, идентичном количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий пару многочленов F множества порядков от нескольких переменных; и

функцию назначения для назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков от нескольких переменных, составляющими элементами которых является указанная пара многочленов F множества порядков от нескольких переменных, в последовательности, определенной заранее между доказывающей стороной и проверяющей стороной.

(20) Компьютерный носитель записи, на котором записана программа согласно (19).

Замечание

Описанные выше алгоритм P доказывающей стороны, алгоритм V проверяющей стороны, алгоритм Sig генерации подписи и алгоритм Ver верификации подписи являются примерами модуля генератора чисел, модуля назначения и модуля хранения таблиц. Описанный выше алгоритм P доказывающей стороны представляет собой пример модуля генератора сообщений, модуля передачи сообщений, модуля передачи ответа, модуля генератора промежуточной информации и модуля передачи промежуточной информации. Кроме того, описанный выше алгоритм V проверяющей стороны представляет собой пример модуля хранения информации, модуля получения сообщений, модуля передачи информации схемы, модуля получения ответа и модуля получения промежуточной информации.

Предпочтительные варианты настоящего изобретения были описаны выше со ссылками на прилагаемые чертежи, тогда как это изобретение, безусловно, приведенными выше примерами не ограничивается. Специалист в рассматриваемой области может найти разнообразные изменения или модификации в пределах объема прилагаемой Формулы изобретения, и должно быть понятно, что такие изменения и модификации естественным образом попадают в пределы технического объема настоящего изобретения.

В приведенном выше описании были предложены алгоритмы, использующие хэш-функцию H, однако здесь может быть также применена функция фиксации (commitment function COM) вместо хэш-функции H. Функция фиксации COM представляет собой

функцию, входными данными которой являются строка S символов и случайное число ρ. Одним из примеров функции фиксации является алгоритм, предложенный Шаи Галеви и Сильвио Микали (Shai Halevi and Silvio Micali) на международной конференции CRYPTO 1996.

Список позиционных обозначений

Gen алгоритм генерации ключей

P алгоритм доказывающей стороны

V алгоритм проверяющей стороны

Sig алгоритм генерации подписи

Ver алгоритм верификации подписи

Формула изобретения

1. Устройство обработки информации, содержащее:

модуль генерации чисел, выполненный с возможностью генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом модуль назначения выполнен с возможностью группировать коэффициенты членов, имеющих идентичный тип сочетания переменных, из множества коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, и выполнять процедуру назначения в блоках групп.

2. Устройство обработки информации, содержащее:

модуль генерации чисел, выполненный с возможностью генерирования чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом модуль назначения выполнен с возможностью группировать матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, причем модуль назначения выполнен с возможностью выполнять процедуру назначения в блоках групп.

3. Устройство обработки информации по п. 1, в котором модуль генерации чисел

выполнен с возможностью генерировать числа по количеству коэффициентов, принадлежащих одной группе, при выполнении модулем назначения процедуры назначения в указанной одной группе.

4. Устройство обработки информации по п. 3, в котором модуль генерации чисел выполнен с возможностью генерировать одно число при генерировании чисел по количеству коэффициентов, принадлежащих одной группе, и генерировать необходимое количество чисел путем поворота генерируемых чисел на каждый бит.

5. Устройство обработки информации по п. 1, дополнительно содержащее:

модуль хранения таблиц, выполненный с возможностью хранения в виде таблицы значений, получаемых путем назначения коэффициентов члену какого-либо типа, соответствующего каждой группе, и подстановки данного числа для переменных соответствующих членов.

6. Устройство обработки информации, содержащее:

модуль генерации чисел, выполненный с возможностью генерации чисел в количестве, идентичном количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий пару многочленов F множества порядков с несколькими переменными; и

модуль назначения, выполненный с возможностью назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, согласованной заранее между доказывающей стороной и проверяющей стороной.

7. Устройство обработки информации по п. 1, в котором указанная информация является начальным числом для генерации случайного числа, а указанная заданная функция является генератором случайных чисел, выполненным с возможностью генерации случайного числа с использованием указанного начального числа.

8. Устройство обработки информации по п. 1, содержащее:

модуль генерации сообщений, выполненный с возможностью генерации сообщения на основе указанной пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и вектора s, являющегося элементом множества K^n;

модуль передачи сообщений, выполненный с возможностью передачи указанного сообщения проверяющей стороне, хранящей указанную пару многочленов F множества порядков от нескольких переменных и векторы $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$; и

модуль передачи ответа, выполненный с возможностью передачи проверяющей стороне ответной информации, соответствующей схеме верификации, выбранной проверяющей стороной из k схем верификации, где $k \geq 3$,

при этом s представляет собой секретный ключ,

указанная пара многочленов F множества порядков от нескольких переменных и векторы y являются открытыми ключами,

ответная информация представляет собой информацию, выбранную из пары случайных чисел и указанного сообщения согласно схеме верификации,

причем указанное сообщение представляет собой информацию, полученную путем

вычислений, выполненных заранее для схемы верификации, соответствующей ответной информации, на основе указанных открытых ключей и ответной информации.

9. Устройство обработки информации по п. 1, содержащее:

модуль для хранения информации, выполненный с возможностью хранения пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и векторов $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$;

модуль получения сообщений, выполненный с возможностью получения сообщения, генерируемого на основе указанной пары многочленов F множества порядков от нескольких переменных и вектора s, являющегося элементом множества K^n;

модуль передачи информации о схеме, выполненный с возможностью передачи проверяющей стороне, передавшей сообщение, информации о схеме верификации, выбранной случайным образом из k схем верификации, где $k\geq 3$

модуль получения ответа, выполненный с возможностью приема ответной информации, соответствующей выбранной схеме верификации, от доказывающей стороны; и

модуль верификации, выполненный с возможностью проверки, хранит ли доказывающая сторона вектор s на основе указанного сообщения, указанной пары многочленов F множества порядков с несколькими переменными, векторов y и ответной информации,

при этом вектор s представляет собой секретный ключ,

указанная пара многочленов F множества порядков от нескольких переменных и векторы y являются открытыми ключами, а

указанное сообщение представляет собой информацию, полученную путем вычислений, выполненных заранее для схемы верификации, соответствующей информации ответа, на основе открытых ключей и ответной информации.

10. Устройство обработки информации по п. 1, содержащее:

модуль генерации сообщений, выполненный с возможностью генерации сообщения на основе указанной пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и вектора s, являющегося элементом множества K^n;

модуль передачи сообщений, выполненный с возможностью передачи указанного сообщения проверяющей стороне, хранящей указанную пару многочленов F множества порядков от нескольких переменных и векторы $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$;

модуль генерации промежуточной информации, выполненный с возможностью генерации третьей информации с использованием первой информации, выбранной случайным образом проверяющей стороной, и второй информации, полученной во время генерации сообщения;

модуль передачи промежуточной информации, выполненный с возможностью передачи третьей информации проверяющей стороне; и

модуль передачи ответа, выполненный с возможностью передачи проверяющей стороне ответной информации, соответствующей схеме верификации, выбранной проверяющей стороной из k схем верификации, где $k\geq 2$

при этом вектор s представляет собой секретный ключ,

указанная пара многочленов F множества порядков от нескольких переменных и векторы y являются открытыми ключами,

ответная информация представляет собой информацию, выбранную из совокупности сообщений согласно схеме верификации, а

указанное сообщение представляет собой информацию, полученную путем вычислений, выполненных заранее для схемы верификации, соответствующей ответной информации, на основе открытых ключей, первой информации, третьей информации и ответной информации.

11. Устройство обработки информации по п. 1, содержащее:

модуль хранения информации, выполненный с возможностью хранения указанной пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и векторов $y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$;

модуль получения сообщений, выполненный с возможностью получения сообщения, генерируемого на основе указанной пары многочленов F множества порядков от нескольких переменных и вектора s, являющегося элементом множества K^n;

модуль передачи информации, выполненный с возможностью передачи выбранной случайным образом первой информации доказывающей стороне, передающей сообщение;

модуль получения промежуточной информации, выполненный с возможностью получения третьей информации, генерируемой доказывающей стороной на основе указанной первой информации и второй информации, полученной во время генерации сообщения;

модуль передачи информации о схеме, выполненный с возможностью передачи доказывающей стороне информации о схеме верификации, выбранной случайным образом из k схем верификации, где $k \geq 2$,

модуль получения ответа, выполненный с возможностью получения от доказывающей стороны информации, соответствующей выбранной схеме верификации; и

модуль верификации, выполненный с возможностью проверки, хранит ли доказывающая сторона вектор s, на основе указанного сообщения, первой информации, третьей информации, указанной пары многочленов F множества порядков с несколькими переменными и ответной информации,

при этом вектор s представляет собой секретный ключ,

указанная пара многочленов F множества порядков от нескольких переменных и векторы y являются открытыми ключами,

причем указанное сообщение представляет собой информацию, получаемую путем вычислений, выполненных заранее для схемы верификации, соответствующей ответной информации, на основе открытых ключей, первой информации, третьей информации и ответной информации.

12. Устройство обработки информации по п. 1, в котором

при многократном исполнении алгоритма модуль генерации чисел выполнен с возможностью генерировать числа только один раз, а модуль назначения выполнен с возможностью выполнения процедуры назначения только один раз,

при этом алгоритм многократно использует коэффициенты, назначаемые посредством модуля назначения.

13. Устройство обработки информации по п. 1, содержащее:

модуль генерации подписи, выполненный с возможностью генерации цифровой подписи для документа M с использованием пары многочленов $F=(f_1, \ldots, f_m)$ множества порядков от нескольких переменных, определенных в кольце K, и ключа подписи, являющегося элементом множества K^n; и

модуль передачи подписи, выполненный с возможностью передачи цифровой

подписи проверяющей стороне, хранящей указанную пару многочленов F множества порядков от нескольких переменных и векторы $y=(f_1(s), \ldots, f_m(s))$.

14. Способ обработки информации, содержащий:

этап генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

этап назначения сгенерированных чисел коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом на этапе назначения чисел группируют коэффициенты членов, имеющих идентичный тип сочетания переменных, из множества коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, и выполняют процедуру назначения в блоках групп.

15. Способ обработки информации, содержащий:

этап генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

этап назначения сгенерированных чисел коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом на этапе назначения чисел группируют матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, и выполняют процедуру назначения в блоках групп.

16. Способ обработки информации, содержащий:

этап генерации чисел в количестве, идентичном количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

этап назначения чисел, генерируемых посредством модуля генератора чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, определенной заранее между доказывающей стороной и проверяющей стороной.

17. Считываемый компьютером носитель записи, содержащий записанную на нем программу, вызывающую выполнение компьютером:

функции генерации чисел для генерации чисел, используемых в коэффициентах

членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

функции назначения для назначения чисел, генерируемых посредством функции генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

при этом функция назначения группирует коэффициенты членов, имеющих идентичный тип сочетания переменных, из множества коэффициентов многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, и выполняет процедуру назначения в блоках групп.

18. Считываемый компьютером носитель записи, содержащий записанную на нем программу, вызывающую выполнение компьютером:

функции генерации чисел для генерации чисел, используемых в коэффициентах членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

функции назначения для назначения чисел, генерируемых посредством функции генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными,

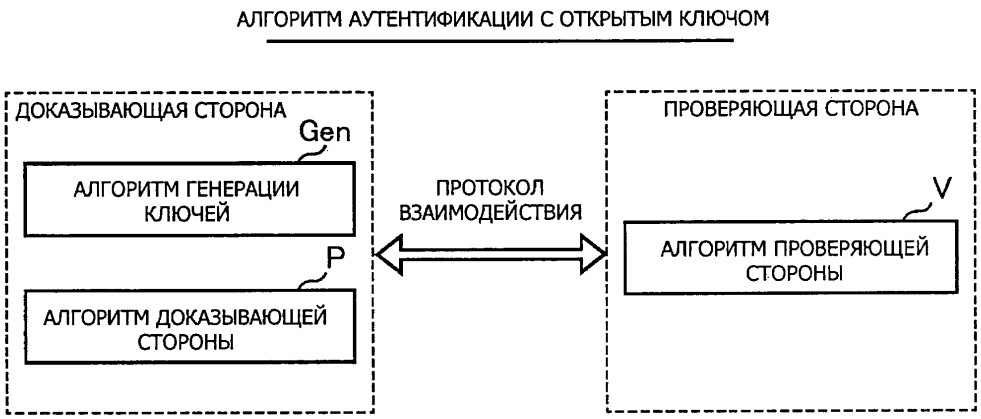
при этом функция назначения группирует матрицу коэффициентов в каждой строке или столбце, когда многочлены множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, выражены в квадратичной форме, при этом функция назначения выполняет процедуру назначения в блоках групп.

19. Считываемый компьютером носитель записи, содержащий записанную на нем программу, вызывающую выполнение компьютером:

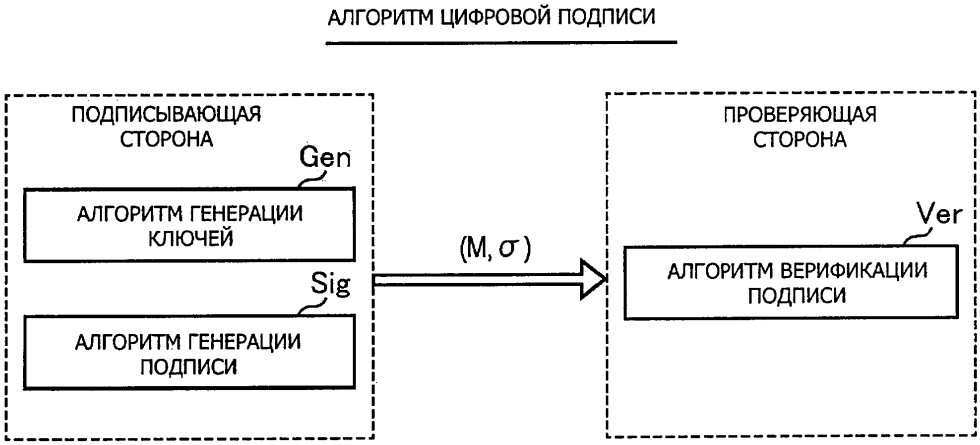
функции генерации чисел для генерации чисел в количестве, идентичном количеству коэффициентов членов, входящих в пару многочленов $F=(f_1, \ldots, f_m)$ множества порядков с несколькими переменными, с использованием заданной функции на основе информации, совместно используемой объектами, исполняющими алгоритм аутентификации с открытым ключом или алгоритм цифровой подписи, применяющий открытый ключ, содержащий указанную пару многочленов F множества порядков с несколькими переменными; и

функции назначения чисел для назначения чисел, генерируемых посредством модуля генерации чисел, коэффициентам многочленов множества порядков с несколькими переменными, составляющими элементами которых является указанная пара многочленов F множества порядков с несколькими переменными, в последовательности, согласованной заранее между доказывающей стороной и проверяющей стороной.

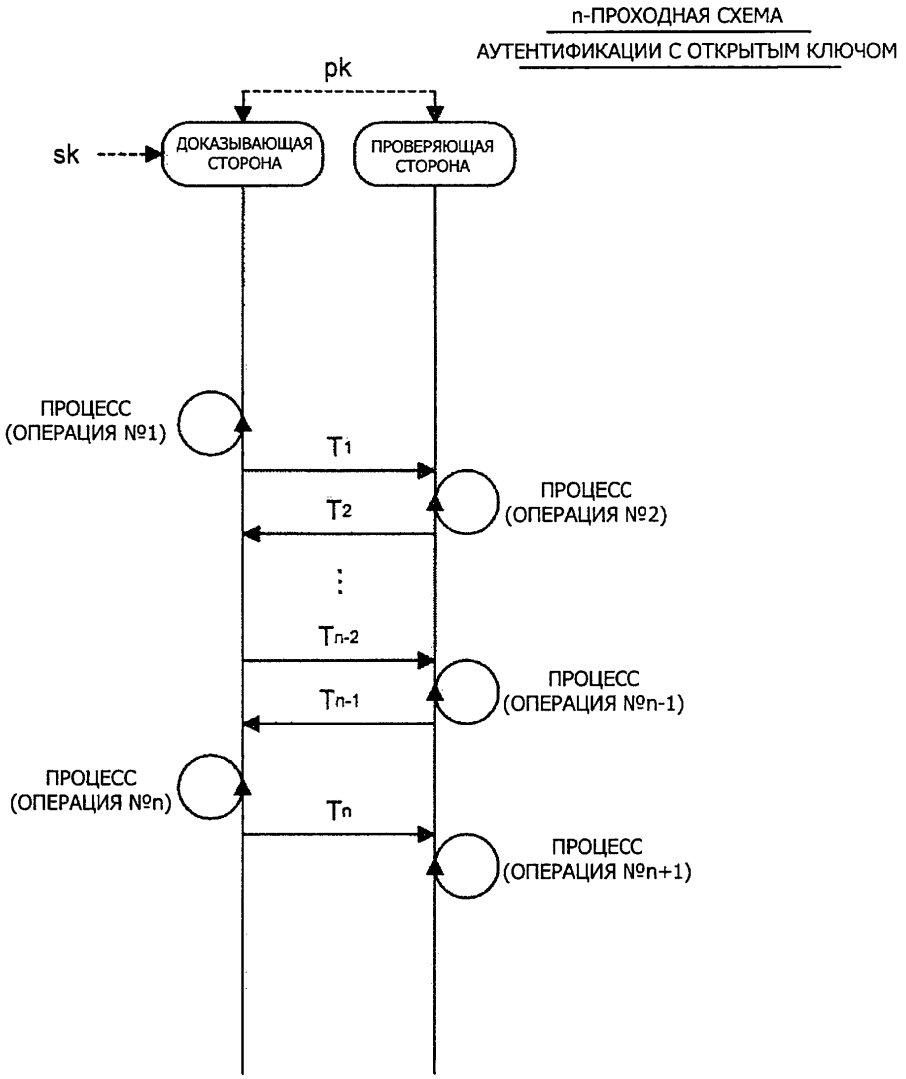
1410027



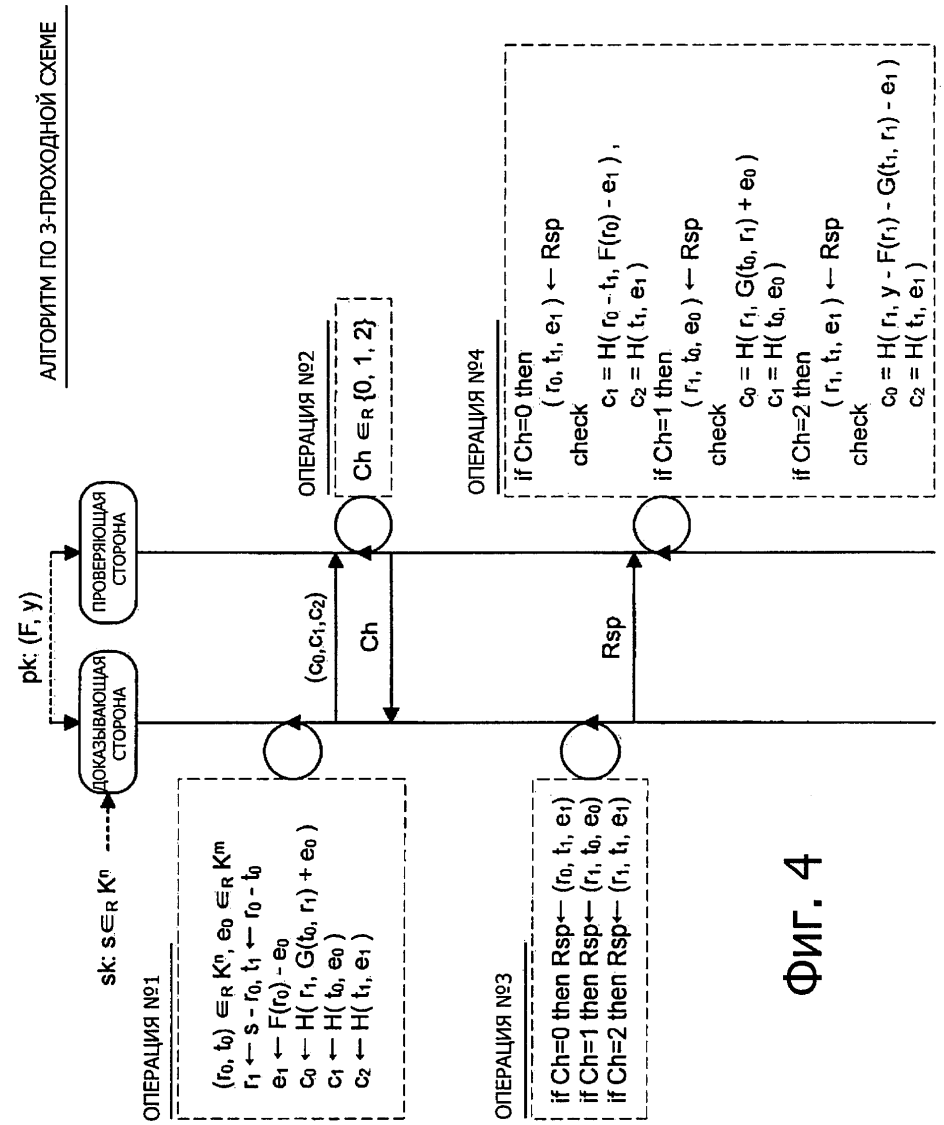
Фиг. 1



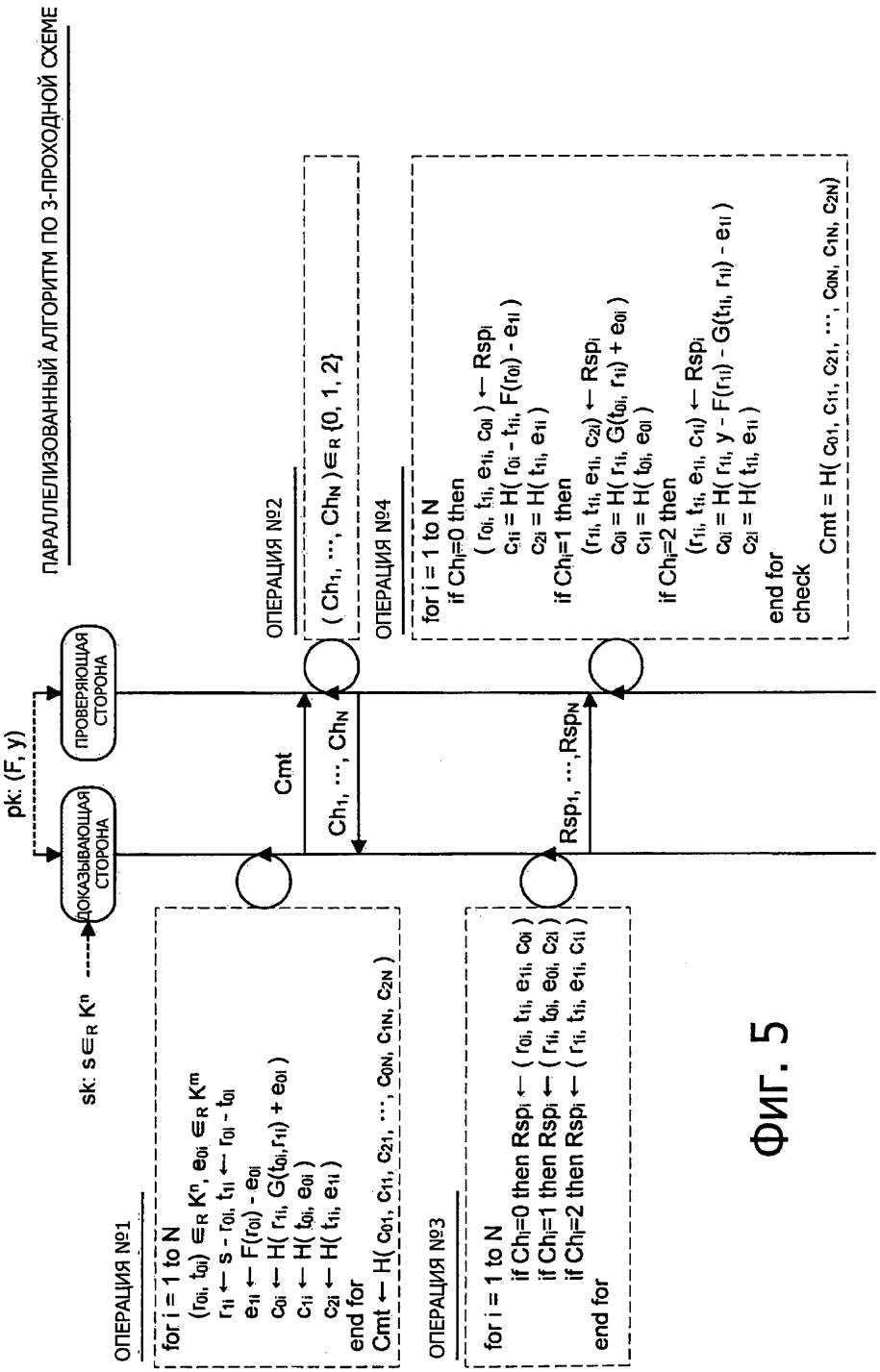
Фиг. 2


n-ПРОХОДНАЯ СХЕМА
АУТЕНТИФИКАЦИИ С ОТКРЫТЫМ КЛЮЧОМ
pk
sk
ДОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
ПРОЦЕСС (ОПЕРАЦИЯ №1)
T1
T2
ПРОЦЕСС (ОПЕРАЦИЯ №2)
Tn-2
Tn-1
ПРОЦЕСС (ОПЕРАЦИЯ №n-1)
ПРОЦЕСС (ОПЕРАЦИЯ №n)
Tn
ПРОЦЕСС (ОПЕРАЦИЯ №n+1)

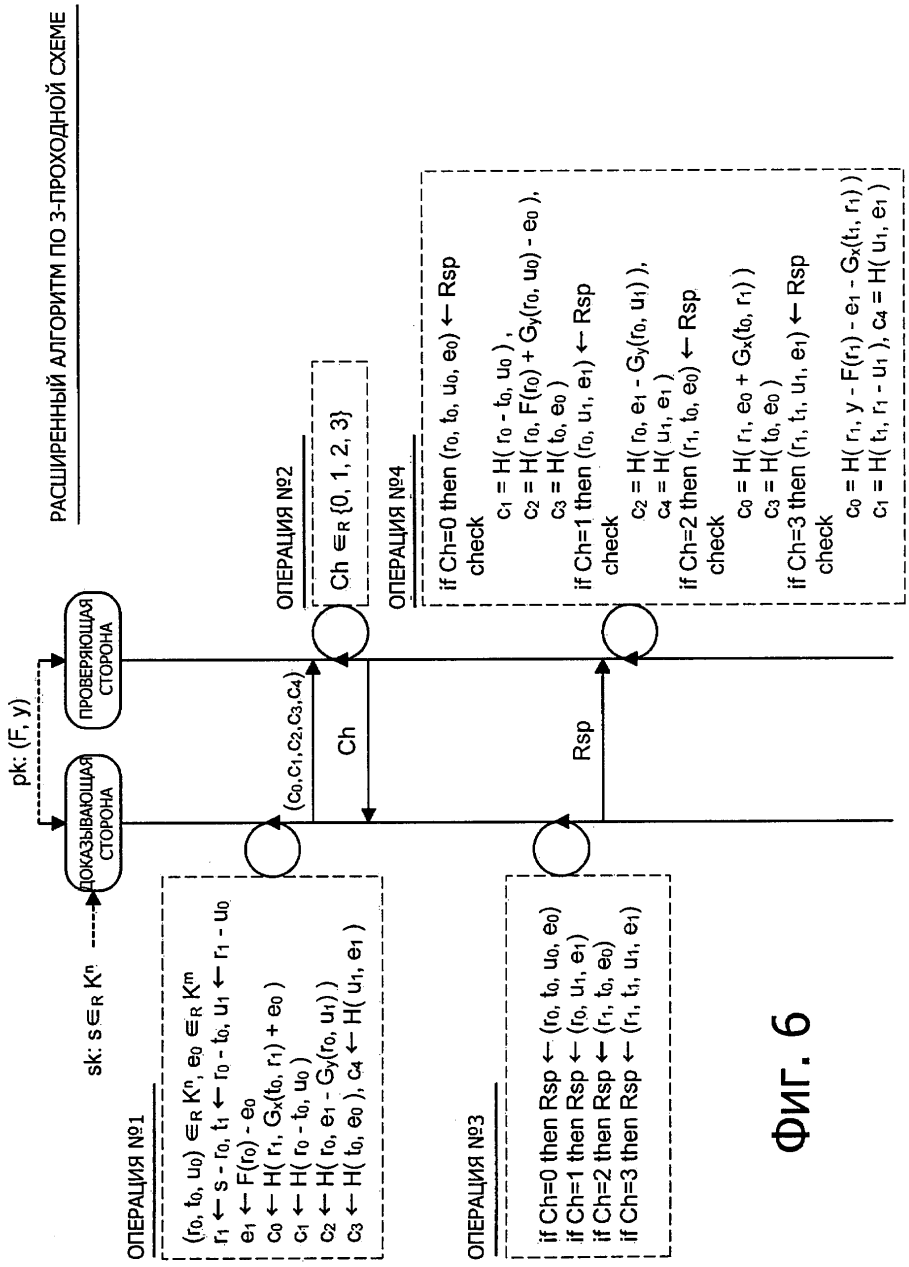
Фиг. 3


АЛГОРИТМ ПО 3-ПРОХОДНОЙ СХЕМЕ
ДОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
pk: (F, y)
sk: s∈R K^n
ОПЕРАЦИЯ №1
(r0, t0) ∈R K^n, e0 ∈R K^m
r1 ← s - r0, t1 ← r0 - t0
e1 ← F(r0) - e0
c0 ← H(r1, G(t0, r1) + e0)
c1 ← H(t0, e0)
c2 ← H(t1, e1)
(c0,c1,c2)
Ch
ОПЕРАЦИЯ №2
Ch ∈R {0, 1, 2}
Rsp
ОПЕРАЦИЯ №3
if Ch=0 then Rsp← (r0, t1, e1)
if Ch=1 then Rsp← (r1, t0, e0)
if Ch=2 then Rsp← (r1, t1, e1)
ОПЕРАЦИЯ №4
if Ch=0 then
(r0, t1, e1) ← Rsp
check
c1 = H(r0 - t1, F(r0) - e1) ,
c2 = H(t1, e1)
if Ch=1 then
(r1, t0, e0) ← Rsp
check
c0 = H(r1, G(t0, r1) + e0)
c1 = H(t0, e0)
if Ch=2 then
(r1, t1, e1) ← Rsp
check
c0 = H(r1, y - F(r1) - G(t1, r1) - e1)
c2 = H(t1, e1)

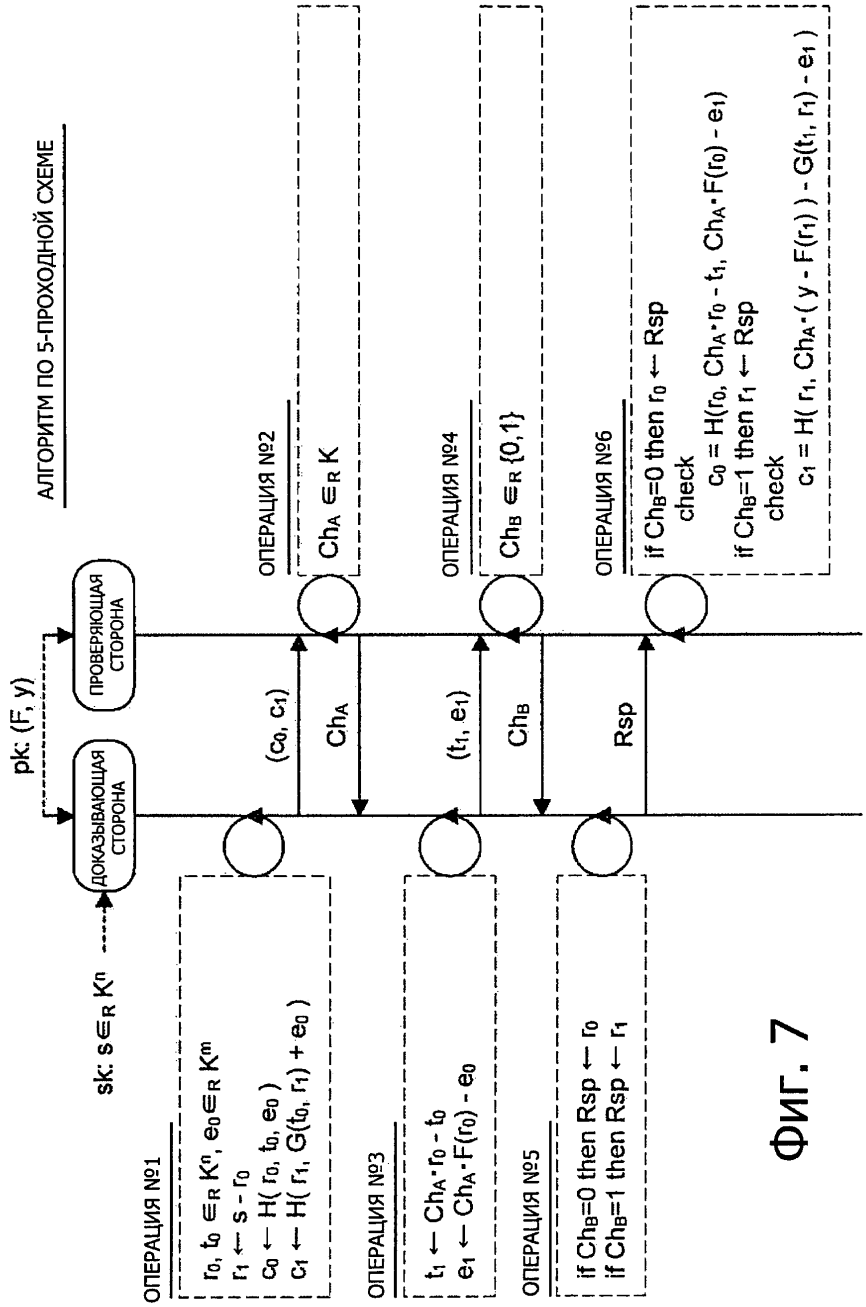
Фиг. 4


ПАРАЛЛЕЛИЗОВАННЫЙ АЛГОРИТМ ПО 3-ПРОХОДНОЙ СХЕМЕ
ПОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
pk: (F, y)
sk: s ∈R Kn
Cmt
Ch1, ···, ChN
Rsp1, ···, RspN
ОПЕРАЦИЯ №1
for i = 1 to N
(r0i, t0i) ∈R Kn, e0i ∈R Km
r1i ← s - r0i, t1i ← r0i - t0i
e1i ← F(r0i) - e0i
c0i ← H(r1i, G(t0i,r1i) + e0i)
c1i ← H(t0i, e0i)
c2i ← H(t1i, e1i)
end for
Cmt ← H(c01, c11, c21, ···, c0N, c1N, c2N)
ОПЕРАЦИЯ №2
(Ch1, ···, ChN)∈R {0, 1, 2}
ОПЕРАЦИЯ №3
for i = 1 to N
if Chi=0 then Rspi ← (r0i, t1i, e1i, c0i)
if Chi=1 then Rspi ← (r1i, t0i, e0i, c2i)
if Chi=2 then Rspi ← (r1i, t1i, e1i, c1i)
end for
ОПЕРАЦИЯ №4
for i = 1 to N
if Chi=0 then
(r0i, t1i, e1i, c0i) ← Rspi
c1i = H(r0i - t1i, F(r0i) - e1i)
c2i = H(t1i, e1i)
if Chi=1 then
(r1i, t1i, e1i, c2i) ← Rspi
c0i = H(r1i, G(t0i, r1i) + e0i)
c1i = H(t0i, e0i)
if Chi=2 then
(r1i, t1i, e1i, c1i) ← Rspi
c0i = H(r1i, y - F(r1i) - G(t1i, r1i) - e1i)
c2i = H(t1i, e1i)
end for
check
Cmt = H(c01, c11, c21, ···, c0N, c1N, c2N)

Фиг. 5

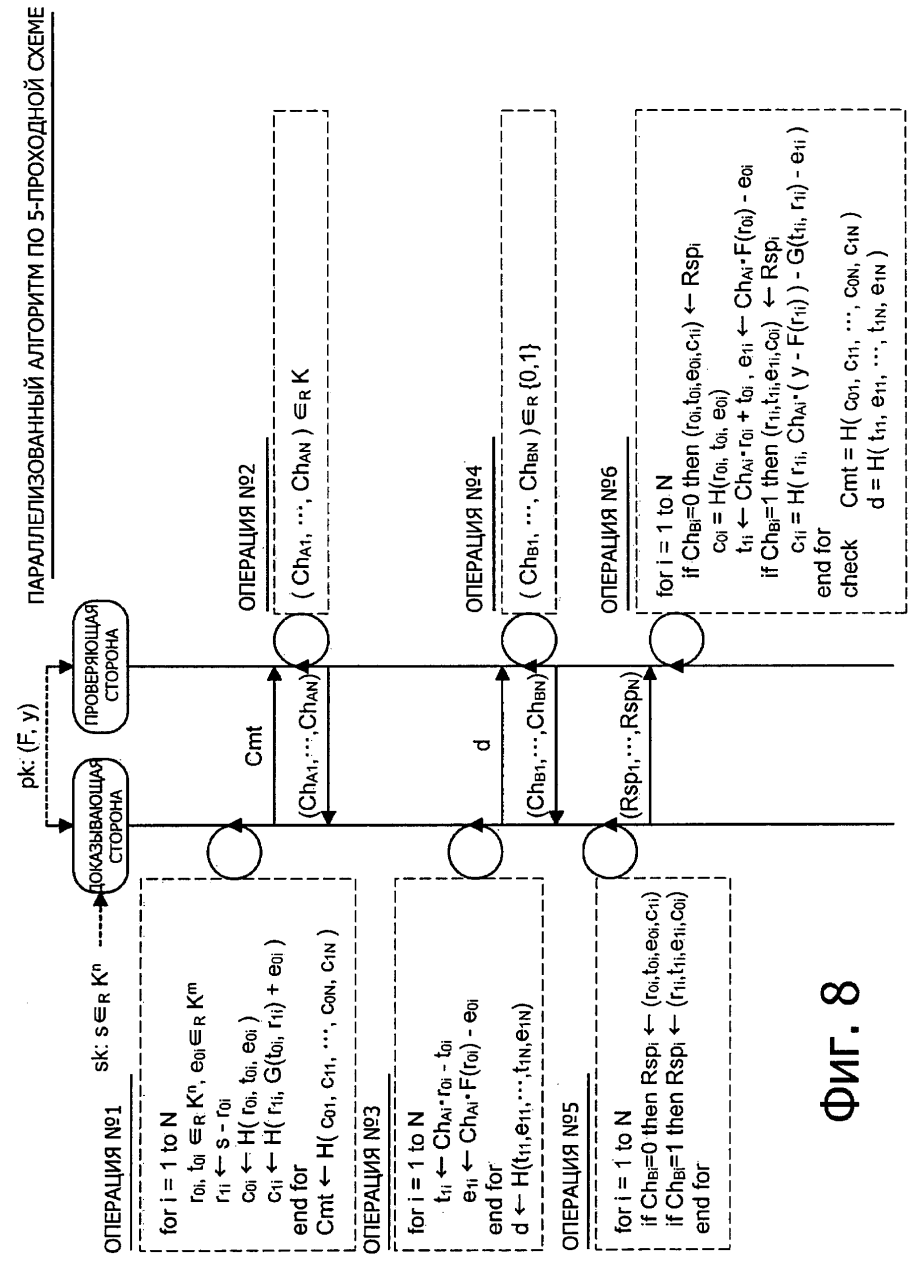

РАСШИРЕННЫЙ АЛГОРИТМ ПО 3-ПРОХОДНОЙ СХЕМЕ
pk: (F, y)
sk: s∈R Kn
ДОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
ОПЕРАЦИЯ №1
(r0, t0, u0) ∈R Kn, e0 ∈R Km
r1 ← s - r0, t1 ← r0 - t0, u1 ← r1 - u0
e1 ← F(r0) - e0
c0 ← H(r1, Gx(t0, r1) + e0)
c1 ← H(r0 - t0, u0)
c2 ← H(r0, e1 - Gy(r0, u1))
c3 ← H(t0, e0), c4 ← H(u1, e1)
(c0,c1,c2,c3,c4)
Ch
Rsp
ОПЕРАЦИЯ №2
Ch ∈R {0, 1, 2, 3}
ОПЕРАЦИЯ №3
if Ch=0 then Rsp ← (r0, t0, u0, e0)
if Ch=1 then Rsp ← (r0, u1, e1)
if Ch=2 then Rsp ← (r1, t0, e0)
if Ch=3 then Rsp ← (r1, t1, u1, e1)
ОПЕРАЦИЯ №4
if Ch=0 then (r0, t0, u0, e0) ← Rsp
check
c1 = H(r0 - t0, u0) ,
c2 = H(r0, F(r0) + Gy(r0, u0) - e0),
c3 = H(t0, e0)
if Ch=1 then (r0, u1, e1) ← Rsp
check
c2 = H(r0, e1 - Gy(r0, u1)),
c4 = H(u1, e1)
if Ch=2 then (r1, t0, e0) ← Rsp
check
c0 = H(r1, e0 + Gx(t0, r1))
c3 = H(t0, e0)
if Ch=3 then (r1, t1, u1, e1) ← Rsp
check
c0 = H(r1, y - F(r1) - e1 - Gx(t1, r1))
c1 = H(t1, r1 - u1), c4 = H(u1, e1)

Фиг. 6

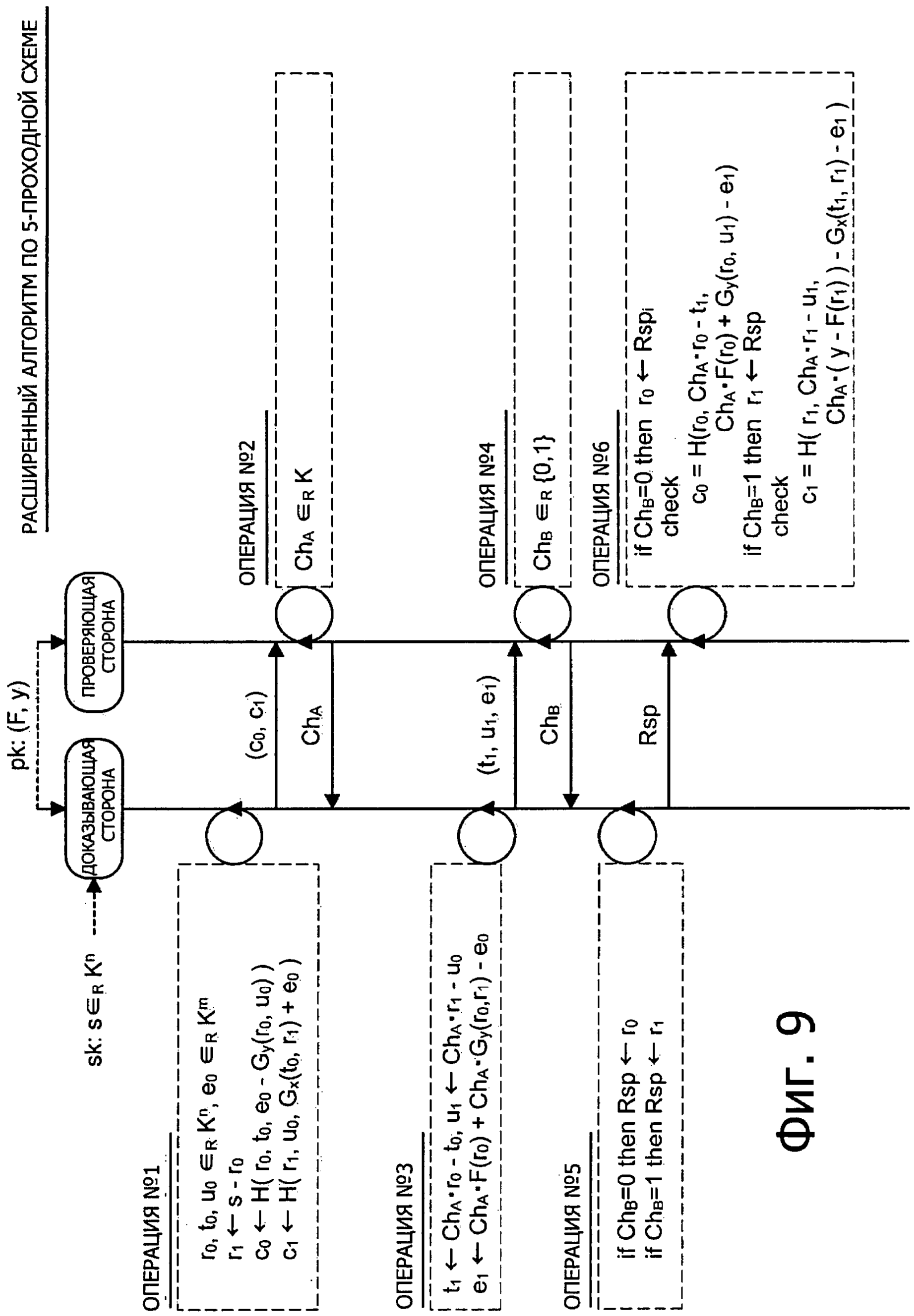

АЛГОРИТМ ПО 5-ПРОХОДНОЙ СХЕМЕ
ДОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
pk: (F, y)
sk: s ∈R Kn
ОПЕРАЦИЯ №1
r0, t0 ∈R Kn, e0 ∈R Km
r1 ← s - r0
c0 ← H(r0, t0, e0)
c1 ← H(r1, G(t0, r1) + e0)
(c0, c1)
ChA
ОПЕРАЦИЯ №2
ChA ∈R K
ОПЕРАЦИЯ №3
t1 ← ChA·r0 - t0
e1 ← ChA·F(r0) - e0
(t1, e1)
ChB
ОПЕРАЦИЯ №4
ChB ∈R {0,1}
ОПЕРАЦИЯ №5
if ChB=0 then Rsp ← r0
if ChB=1 then Rsp ← r1
Rsp
ОПЕРАЦИЯ №6
if ChB=0 then r0 ← Rsp
check
c0 = H(r0, ChA·r0 - t1, ChA·F(r0) - e1)
if ChB=1 then r1 ← Rsp
check
c1 = H(r1, ChA·(y - F(r1)) - G(t1, r1) - e1)

Фиг. 7

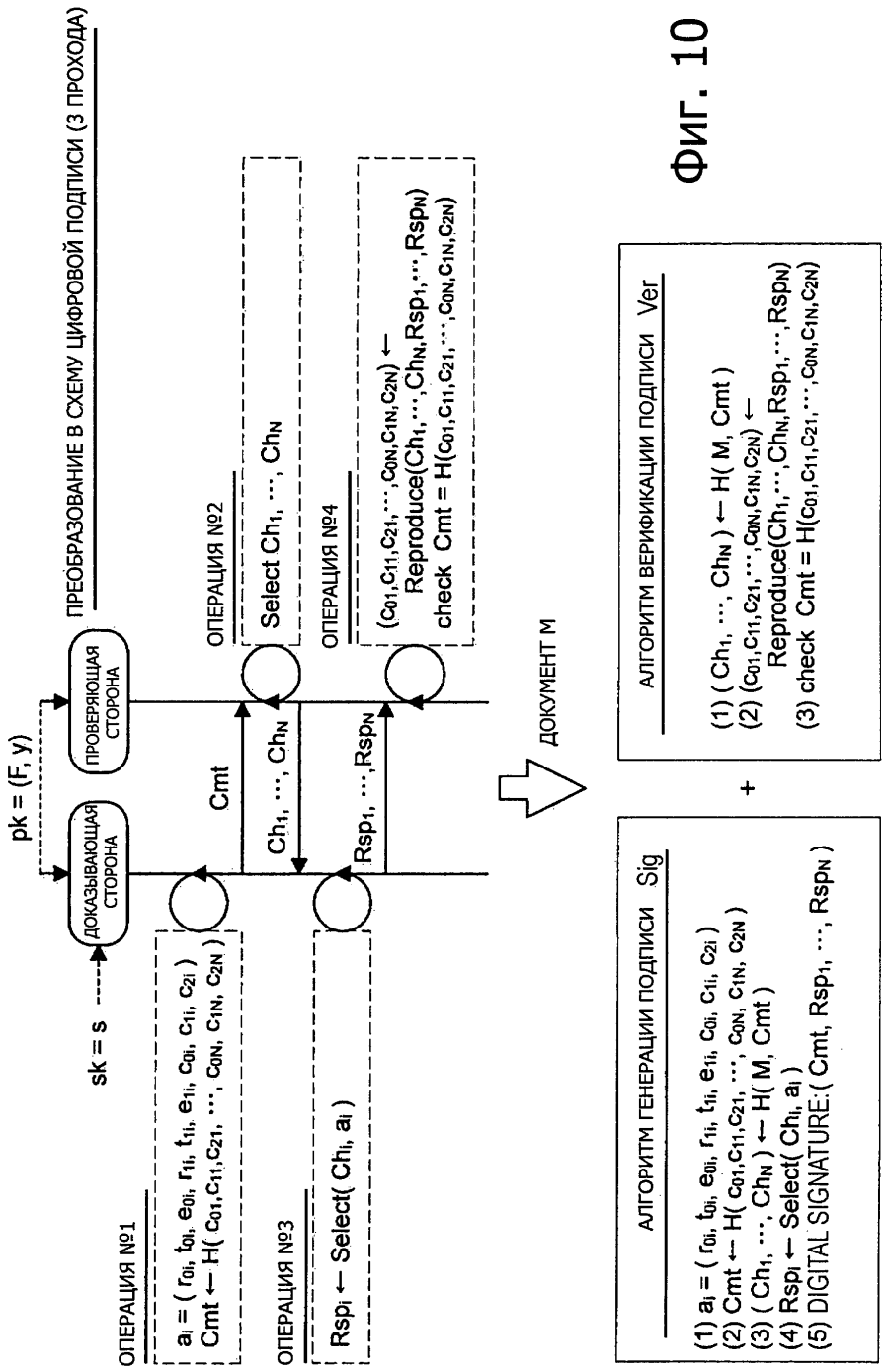
7/25


ПАРАЛЛЕЛИЗОВАННЫЙ АЛГОРИТМ ПО 5-ПРОХОДНОЙ СХЕМЕ
pk: (F, y)
sk: s ∈R K^n
ПОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
Cmt
(Ch_A1, ···, Ch_AN)
d
(Ch_B1, ···, Ch_BN)
(Rsp_1, ···, Rsp_N)
ОПЕРАЦИЯ №1
for i = 1 to N
r_0i, t_0i ∈R K^n, e_0i ∈R K^m
r_1i ← s - r_0i
c_0i ← H(r_0i, t_0i, e_0i)
c_1i ← H(r_1i, G(t_0i, r_1i) + e_0i)
end for
Cmt ← H(c_01, c_11, ···, c_0N, c_1N)
ОПЕРАЦИЯ №2
(Ch_A1, ···, Ch_AN) ∈R K
ОПЕРАЦИЯ №3
for i = 1 to N
t_1i ← Ch_Ai·r_0i - t_0i
e_1i ← Ch_Ai·F(r_0i) - e_0i
end for
d ← H(t_11, e_11, ···, t_1N, e_1N)
ОПЕРАЦИЯ №4
(Ch_B1, ···, Ch_BN)∈R {0,1}
ОПЕРАЦИЯ №5
for i = 1 to N
if Ch_Bi=0 then Rsp_i ← (r_0i, t_0i, e_0i, c_1i)
if Ch_Bi=1 then Rsp_i ← (r_1i, t_1i, e_1i, c_0i)
end for
ОПЕРАЦИЯ №6
for i = 1 to N
if Ch_Bi=0 then (r_0i, t_0i, e_0i, c_1i) ← Rsp_i
c_0i = H(r_0i, t_0i, e_0i)
t_1i ← Ch_Ai·r_0i + t_0i , e_1i ← Ch_Ai·F(r_0i) - e_0i
if Ch_Bi=1 then (r_1i, t_1i, e_1i, c_0i) ← Rsp_i
c_1i = H(r_1i, Ch_Ai·(y - F(r_1i)) - G(t_1i, r_1i) - e_1i)
end for
check Cmt = H(c_01, c_11, ···, c_0N, c_1N)
d = H(t_11, e_11, ···, t_1N, e_1N)

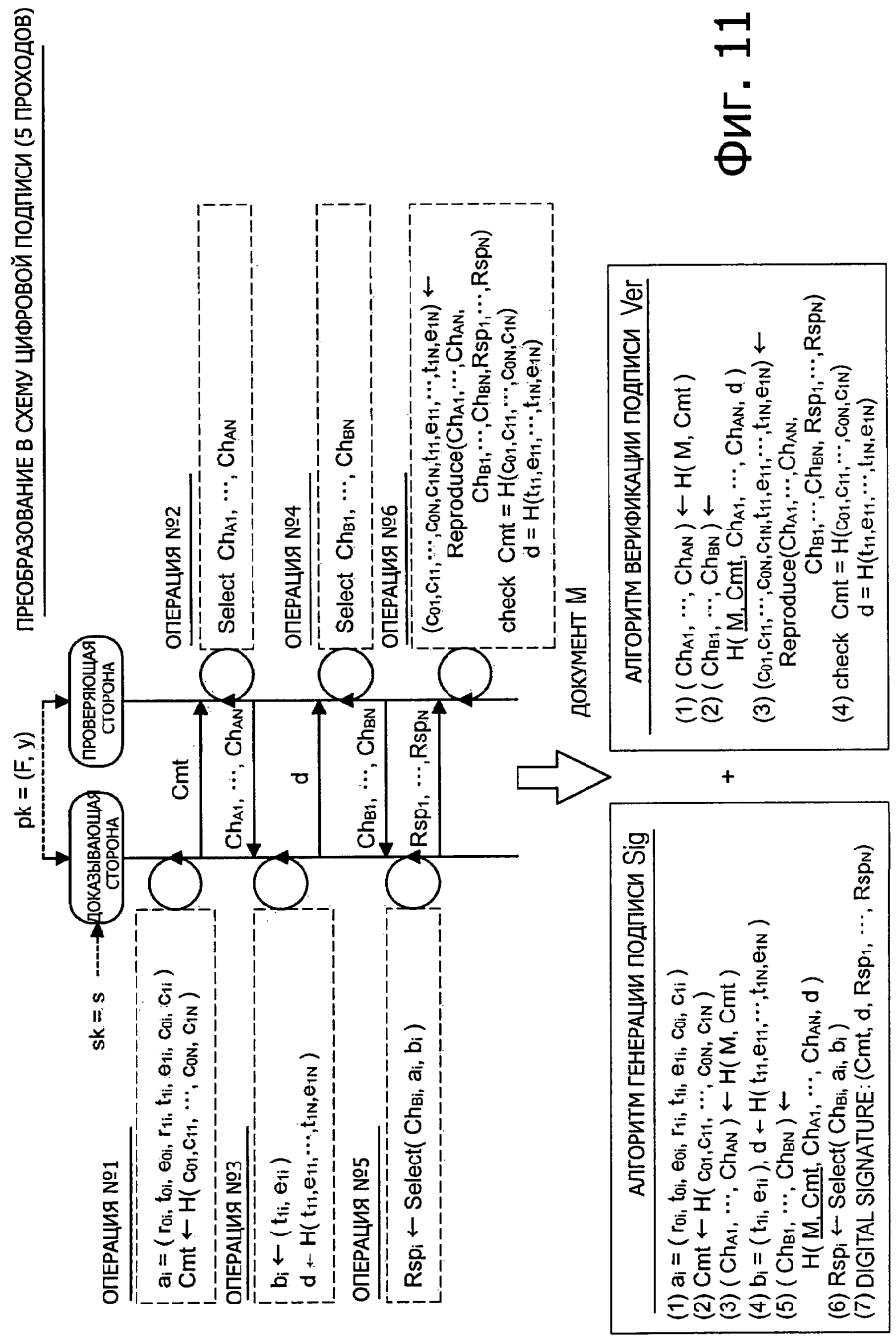
Фиг. 8


РАСШИРЕННЫЙ АЛГОРИТМ ПО 5-ПРОХОДНОЙ СХЕМЕ
ДОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
pk: (F, y)
sk: s ∈R Kn
ОПЕРАЦИЯ №1
r0, t0, u0 ∈R Kn, e0 ∈R Km
r1 ← s - r0
c0 ← H(r0, t0, e0 - Gy(r0, u0))
c1 ← H(r1, u0, Gx(t0, r1) + e0)
(c0, c1)
ОПЕРАЦИЯ №2
ChA ∈R K
ChA
ОПЕРАЦИЯ №3
t1 ← ChA·r0 - t0, u1 ← ChA·r1 - u0
e1 ← ChA·F(r0) + ChA·Gy(r0,r1) - e0
(t1, u1, e1)
ОПЕРАЦИЯ №4
ChB ∈R {0,1}
ChB
ОПЕРАЦИЯ №5
if ChB=0 then Rsp ← r0
if ChB=1 then Rsp ← r1
Rsp
ОПЕРАЦИЯ №6
if ChB=0 then r0 ← Rspi
check
c0 = H(r0, ChA·r0 - t1,
ChA·F(r0) + Gy(r0, u1) - e1)
if ChB=1 then r1 ← Rsp
check
c1 = H(r1, ChA·r1 - u1,
ChA·(y - F(r1)) - Gx(t1, r1) - e1)

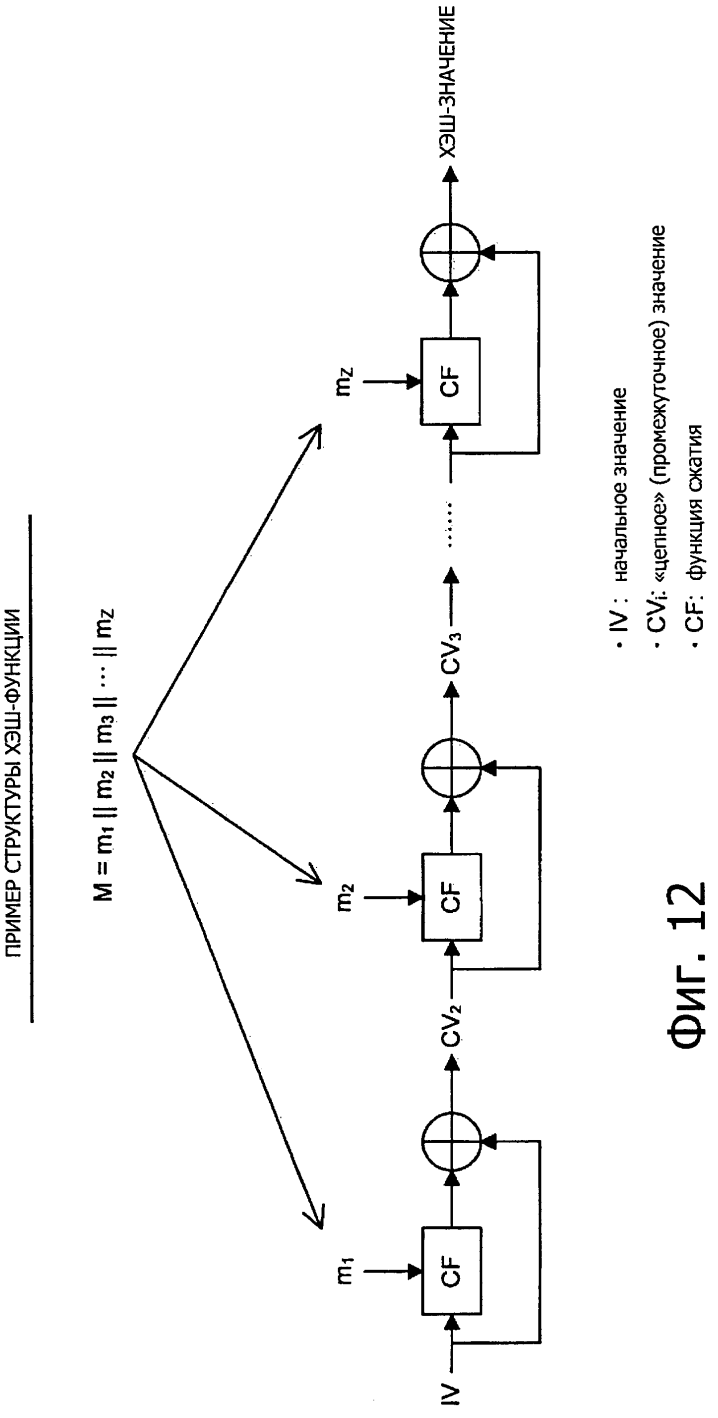
Фиг. 9


ПРЕОБРАЗОВАНИЕ В СХЕМУ ЦИФРОВОЙ ПОДПИСИ (3 ПРОХОДА)
pk = (F, y)
sk = s
ДОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
Cmt
Ch1, ···, ChN
Rsp1, ···, RspN
ОПЕРАЦИЯ №1
ai = (r0i, t0i, e0i, r1i, t1i, e1i, c0i, c1i, c2i)
Cmt ← H(c01,c11,c21, ···, c0N, c1N, c2N)
ОПЕРАЦИЯ №2
Select Ch1, ···, ChN
ОПЕРАЦИЯ №3
Rspi ← Select(Chi, ai)
ОПЕРАЦИЯ №4
(c01,c11,c21, ···, c0N,c1N,c2N) ←
Reproduce(Ch1, ···, ChN,Rsp1, ···, RspN)
check Cmt = H(c01,c11,c21, ···, c0N,c1N,c2N)
ДОКУМЕНТ M
АЛГОРИТМ ГЕНЕРАЦИИ ПОДПИСИ Sig
(1) ai = (r0i, t0i, e0i, r1i, t1i, e1i, c0i, c1i, c2i)
(2) Cmt ← H(c01,c11,c21, ···, c0N, c1N, c2N)
(3) (Ch1, ···, ChN) ← H(M, Cmt)
(4) Rspi ← Select(Chi, ai)
(5) DIGITAL SIGNATURE: (Cmt, Rsp1, ···, RspN)
+
АЛГОРИТМ ВЕРИФИКАЦИИ ПОДПИСИ Ver
(1) (Ch1, ···, ChN) ← H(M, Cmt)
(2) (c01,c11,c21, ···, c0N,c1N,c2N) ←
Reproduce(Ch1, ···, ChN,Rsp1, ···, RspN)
(3) check Cmt = H(c01,c11,c21, ···, c0N,c1N,c2N)

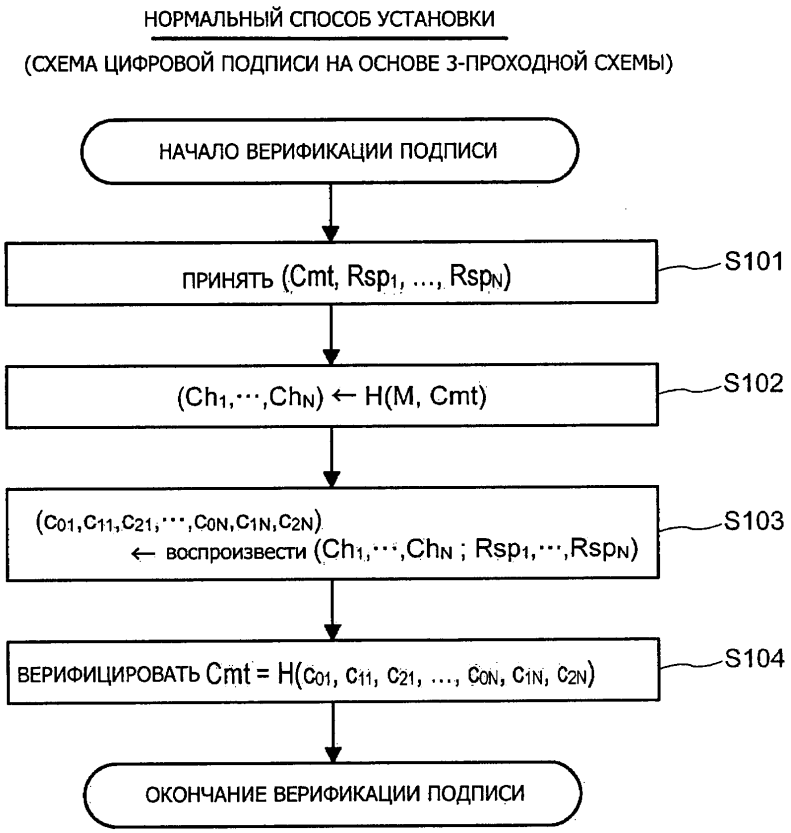
Фиг. 10


ПРЕОБРАЗОВАНИЕ В СХЕМУ ЦИФРОВОЙ ПОДПИСИ (5 ПРОХОДОВ)
pk = (F, y)
sk = s
ПОКАЗЫВАЮЩАЯ СТОРОНА
ПРОВЕРЯЮЩАЯ СТОРОНА
Cmt
Ch_A1, ···, Ch_AN
d
Ch_B1, ···, Ch_BN
Rsp_1, ···, Rsp_N
ОПЕРАЦИЯ №1
a_i = (r_0i, t_0i, e_0i, r_1i, t_1i, e_1i, c_0i, c_1i)
Cmt ← H(c_01,c_11, ···, c_0N, c_1N)
ОПЕРАЦИЯ №2
Select Ch_A1, ···, Ch_AN
ОПЕРАЦИЯ №3
b_i ← (t_1i, e_1i)
d ← H(t_11,e_11, ···,t_1N,e_1N)
ОПЕРАЦИЯ №4
Select Ch_B1, ···, Ch_BN
ОПЕРАЦИЯ №5
Rsp_i ← Select(Ch_Bi, a_i, b_i)
ОПЕРАЦИЯ №6
(c_01,c_11, ···, c_0N,c_1N,t_11,e_11, ···,t_1N,e_1N) ←
Reproduce(Ch_A1, ···,Ch_AN,
Ch_B1, ···,Ch_BN,Rsp_1, ···,Rsp_N)
check Cmt = H(c_01,c_11, ···, c_0N,c_1N)
d = H(t_11,e_11, ···, t_1N,e_1N)
ДОКУМЕНТ M
АЛГОРИТМ ГЕНЕРАЦИИ ПОДПИСИ Sig
(1) a_i = (r_0i, t_0i, e_0i, r_1i, t_1i, e_1i, c_0i, c_1i)
(2) Cmt ← H(c_01,c_11, ···, c_0N, c_1N)
(3) (Ch_A1, ···, Ch_AN) ← H(M, Cmt)
(4) b_i = (t_1i, e_1i), d ← H(t_11,e_11, ···,t_1N,e_1N)
(5) (Ch_B1, ···, Ch_BN) ←
H(M, Cmt, Ch_A1, ···, Ch_AN, d)
(6) Rsp_i ← Select(Ch_Bi, a_i, b_i)
(7) DIGITAL SIGNATURE: (Cmt, d, Rsp_1, ···, Rsp_N)
+
АЛГОРИТМ ВЕРИФИКАЦИИ ПОДПИСИ Ver
(1) (Ch_A1, ···, Ch_AN) ← H(M, Cmt)
(2) (Ch_B1, ···, Ch_BN) ←
H(M, Cmt, Ch_A1, ···, Ch_AN, d)
(3) (c_01,c_11, ···, c_0N,c_1N,t_11,e_11, ···,t_1N,e_1N) ←
Reproduce(Ch_A1, ···,Ch_AN,
Ch_B1, ···,Ch_BN, Rsp_1, ···,Rsp_N)
(4) check Cmt = H(c_01,c_11, ···, c_0N,c_1N)
d = H(t_11,e_11, ···,t_1N,e_1N)

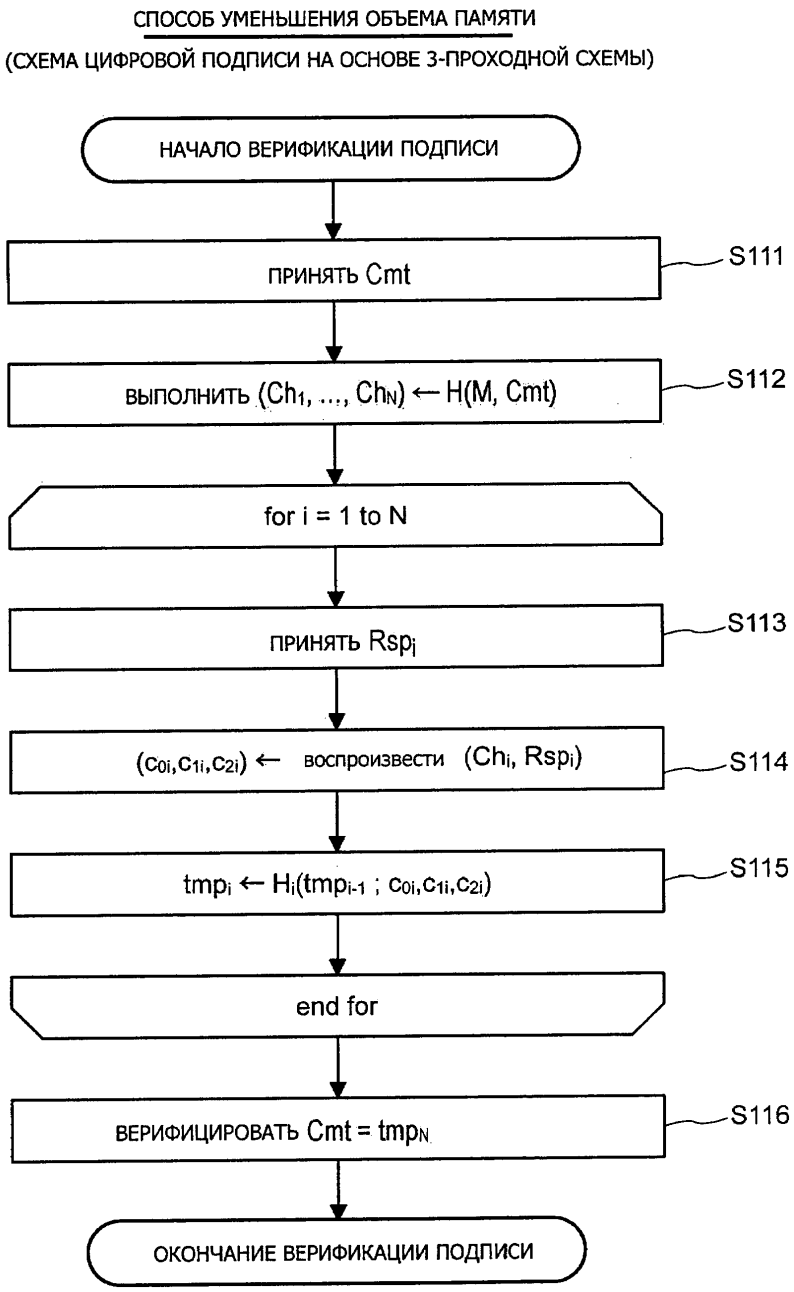
Фиг. 11


ПРИМЕР СТРУКТУРЫ ХЭШ-ФУНКЦИИ
M = m1 || m2 || m3 || ··· || mZ
IV
m1
CF
CV2
m2
CF
CV3
mZ
CF
ХЭШ-ЗНАЧЕНИЕ
• IV: начальное значение
• CVi: «цепное» (промежуточное) значение
• CF: функция сжатия

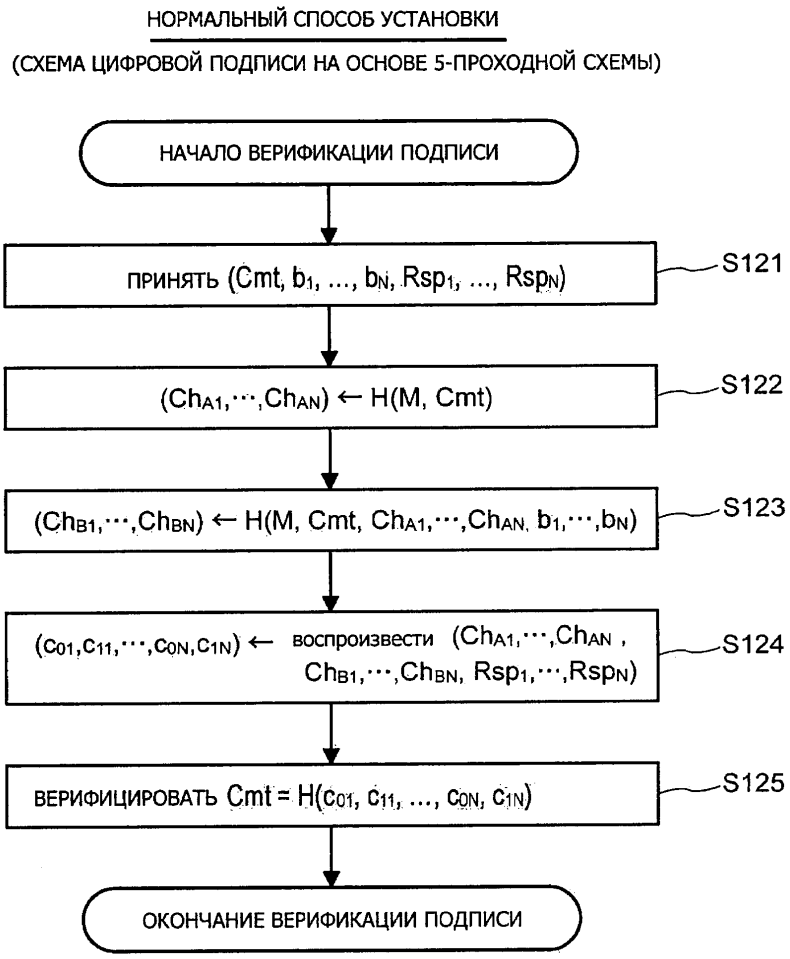
Фиг. 12


НОРМАЛЬНЫЙ СПОСОБ УСТАНОВКИ
(СХЕМА ЦИФРОВОЙ ПОДПИСИ НА ОСНОВЕ 3-ПРОХОДНОЙ СХЕМЫ)
НАЧАЛО ВЕРИФИКАЦИИ ПОДПИСИ
ПРИНЯТЬ (Cmt, Rsp1, ..., RspN)
S101
(Ch1,⋯,ChN) ← H(M, Cmt)
S102
(c01,c11,c21,⋯,c0N,c1N,c2N)
← воспроизвести (Ch1,⋯,ChN ; Rsp1,⋯,RspN)
S103
ВЕРИФИЦИРОВАТЬ Cmt = H(c01, c11, c21, ..., c0N, c1N, c2N)
S104
ОКОНЧАНИЕ ВЕРИФИКАЦИИ ПОДПИСИ

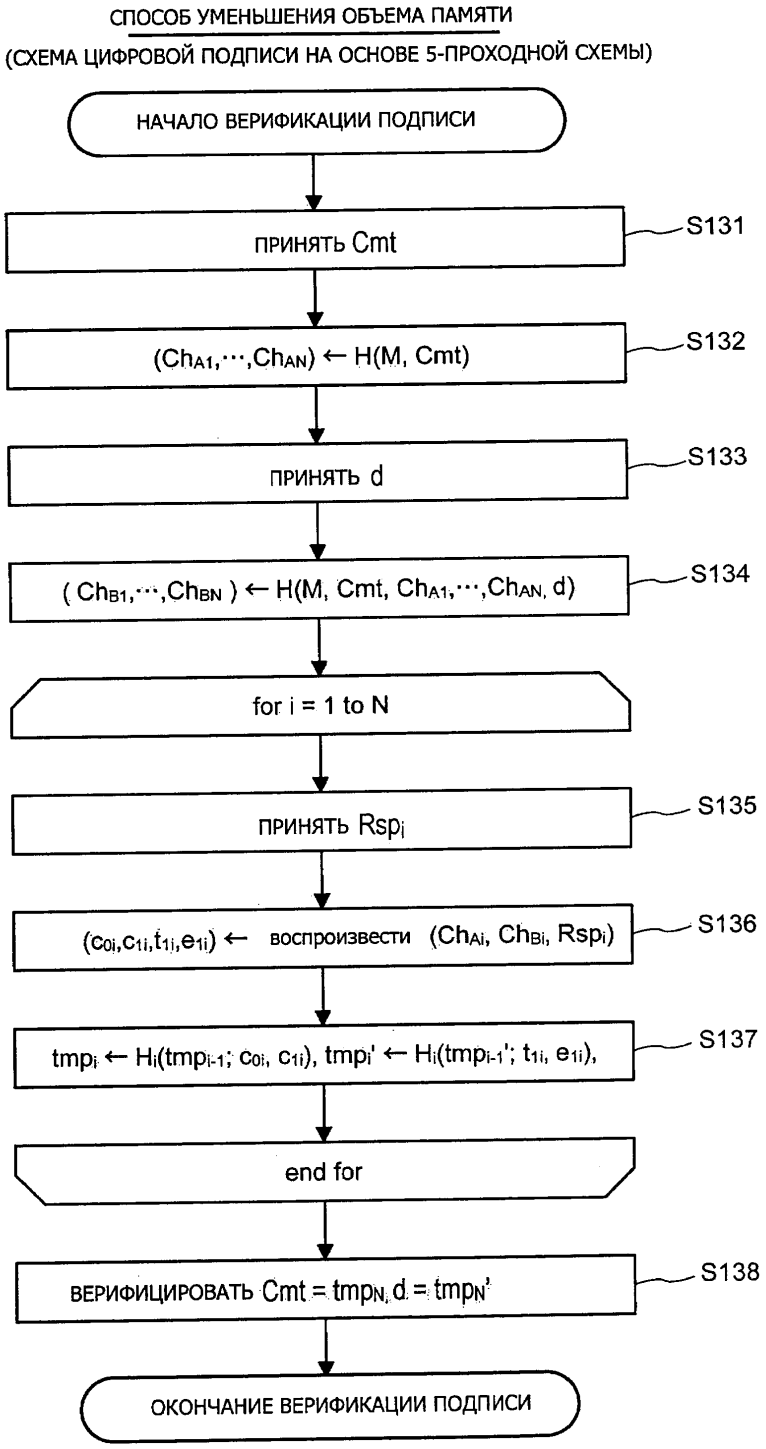
Фиг. 13


СПОСОБ УМЕНЬШЕНИЯ ОБЪЕМА ПАМЯТИ
(СХЕМА ЦИФРОВОЙ ПОДПИСИ НА ОСНОВЕ 3-ПРОХОДНОЙ СХЕМЫ)
НАЧАЛО ВЕРИФИКАЦИИ ПОДПИСИ
ПРИНЯТЬ Cmt
S111
ВЫПОЛНИТЬ (Ch1, ..., ChN) ← H(M, Cmt)
S112
for i = 1 to N
ПРИНЯТЬ Rspi
S113
(c0i,c1i,c2i) ← воспроизвести (Chi, Rspi)
S114
tmpi ← Hi(tmpi-1 ; c0i,c1i,c2i)
S115
end for
ВЕРИФИЦИРОВАТЬ Cmt = tmpN
S116
ОКОНЧАНИЕ ВЕРИФИКАЦИИ ПОДПИСИ

Фиг. 14


НОРМАЛЬНЫЙ СПОСОБ УСТАНОВКИ
(СХЕМА ЦИФРОВОЙ ПОДПИСИ НА ОСНОВЕ 5-ПРОХОДНОЙ СХЕМЫ)
НАЧАЛО ВЕРИФИКАЦИИ ПОДПИСИ
принять (Cmt, b1, ..., bN, Rsp1, ..., RspN)
S121
(ChA1,⋯,ChAN) ← H(M, Cmt)
S122
(ChB1,⋯,ChBN) ← H(M, Cmt, ChA1,⋯,ChAN, b1,⋯,bN)
S123
(c01,c11,⋯,c0N,c1N) ← воспроизвести (ChA1,⋯,ChAN , ChB1,⋯,ChBN, Rsp1,⋯,RspN)
S124
ВЕРИФИЦИРОВАТЬ Cmt = H(c01, c11, ..., c0N, c1N)
S125
ОКОНЧАНИЕ ВЕРИФИКАЦИИ ПОДПИСИ

Фиг. 15

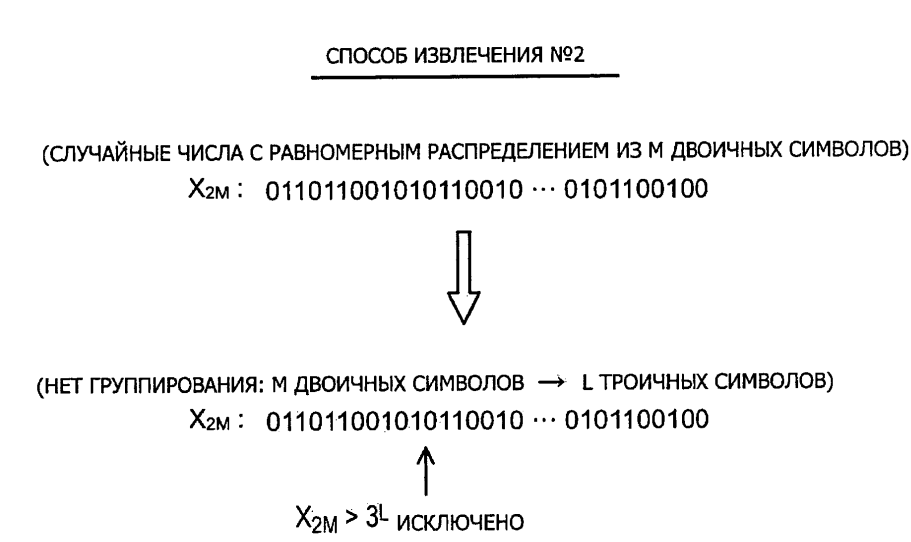

СПОСОБ УМЕНЬШЕНИЯ ОБЪЕМА ПАМЯТИ
(СХЕМА ЦИФРОВОЙ ПОДПИСИ НА ОСНОВЕ 5-ПРОХОДНОЙ СХЕМЫ)
НАЧАЛО ВЕРИФИКАЦИИ ПОДПИСИ
ПРИНЯТЬ Cmt
S131
(ChA1,⋯,ChAN) ← H(M, Cmt)
S132
ПРИНЯТЬ d
S133
(ChB1,⋯,ChBN) ← H(M, Cmt, ChA1,⋯,ChAN, d)
S134
for i = 1 to N
ПРИНЯТЬ Rspi
S135
(c0i,c1i,t1i,e1i) ← воспроизвести (ChAi, ChBi, Rspi)
S136
tmpi ← Hi(tmpi-1; c0i, c1i), tmpi' ← Hi(tmpi-1'; t1i, e1i),
S137
end for
ВЕРИФИЦИРОВАТЬ Cmt = tmpN, d = tmpN'
S138
ОКОНЧАНИЕ ВЕРИФИКАЦИИ ПОДПИСИ

Фиг. 16

16/25

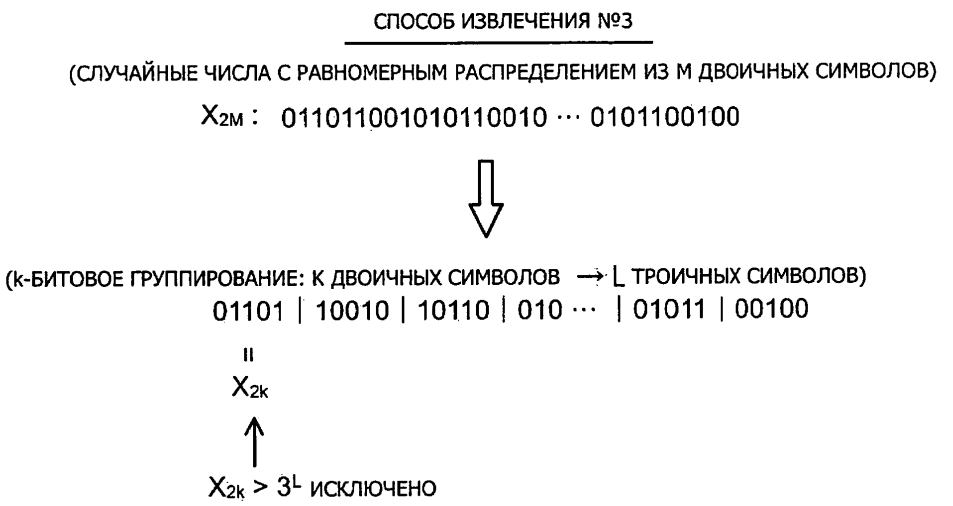


Фиг. 17

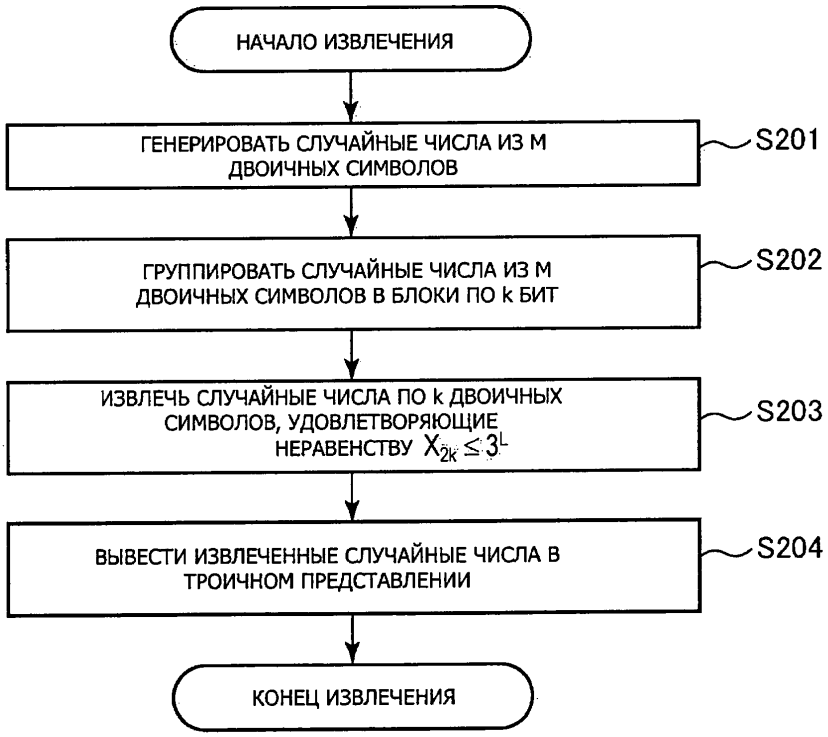


Фиг. 18

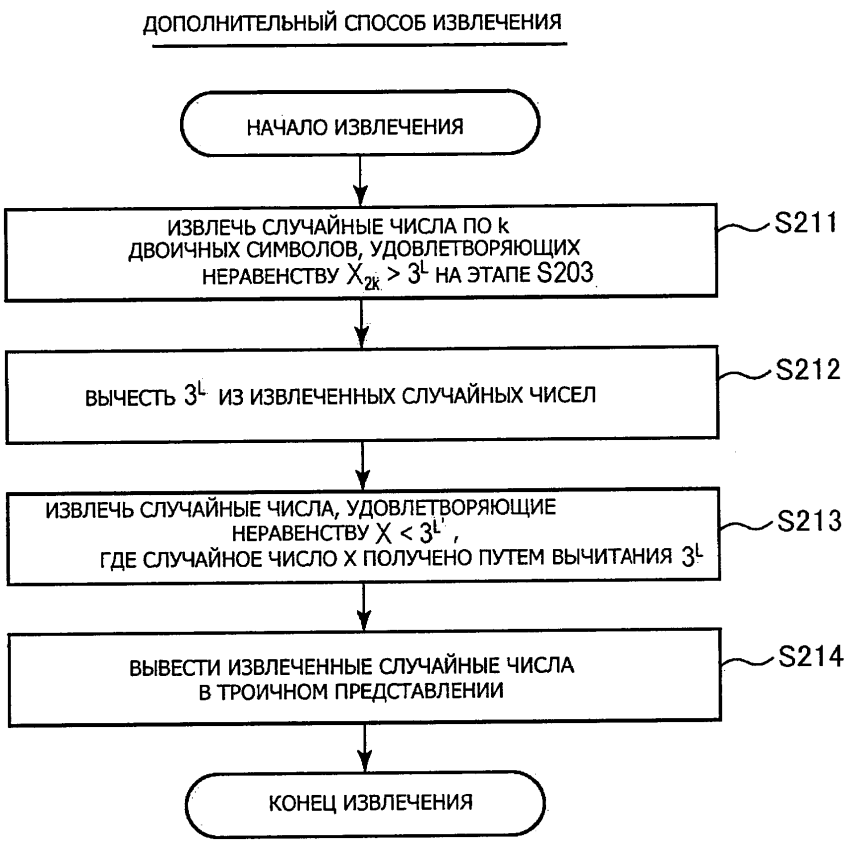
17/25



Фиг. 19



Фиг. 20


ДОПОЛНИТЕЛЬНЫЙ СПОСОБ ИЗВЛЕЧЕНИЯ
НАЧАЛО ИЗВЛЕЧЕНИЯ
ИЗВЛЕЧЬ СЛУЧАЙНЫЕ ЧИСЛА ПО k ДВОИЧНЫХ СИМВОЛОВ, УДОВЛЕТВОРЯЮЩИХ НЕРАВЕНСТВУ $X_{2k} > 3^L$ НА ЭТАПЕ S203
S211
ВЫЧЕСТЬ 3^L ИЗ ИЗВЛЕЧЕННЫХ СЛУЧАЙНЫХ ЧИСЕЛ
S212
ИЗВЛЕЧЬ СЛУЧАЙНЫЕ ЧИСЛА, УДОВЛЕТВОРЯЮЩИЕ НЕРАВЕНСТВУ $X < 3^{L'}$, ГДЕ СЛУЧАЙНОЕ ЧИСЛО X ПОЛУЧЕНО ПУТЕМ ВЫЧИТАНИЯ 3^L
S213
ВЫВЕСТИ ИЗВЛЕЧЕННЫЕ СЛУЧАЙНЫЕ ЧИСЛА В ТРОИЧНОМ ПРЕДСТАВЛЕНИИ
S214
КОНЕЦ ИЗВЛЕЧЕНИЯ

Фиг. 21

СПОСОБ СТРУКТУРИРОВАНИЯ №1 (ПРИМЕНИТЕЛЬНО К МНОГОЧЛЕНУ F)

$$F(x_1,\cdots,x_N)=\begin{cases}\sum_{I,J} a_{1IJ}x_Ix_J+\sum_{I} b_{1I}x_I=y_1\\ \vdots\\ \sum_{I,J} a_{MIJ}x_Ix_J+\sum_{I} b_{MI}x_I=y_M\end{cases}$$

СТРУКТУРА ДАННЫХ A СТРУКТУРА ДАННЫХ B

Фиг. 22

СПОСОБ СТРУКТУРИРОВАНИЯ №1 (ПРИМЕНИТЕЛЬНО К МНОГОЧЛЕНУ G)

$$G(x_1,\cdots,x_N,y_1,\cdots,y_N)=\begin{cases}\sum_{I,J}\left(a_{1IJ}+a_{1JI}\right)x_I y_J=z_1\\ \vdots\\ \sum_{I,J}\left(a_{MIJ}+a_{MJI}\right)x_I y_J=z_M\end{cases}$$

СТРУКТУРА ДАННЫХ

Фиг. 23

СПОСОБ СТРУКТУРИРОВАНИЯ №1 (ПРИМЕНИТЕЛЬНО К КУБИЧЕСКОМУ МНОГОЧЛЕНУ F)

$$F(x_1,\cdots,x_N)=\begin{cases}\underbrace{\sum_{IJL} a_{1IJL}x_Ix_Jx_L}_{\text{СТРУКТУРА ДАННЫХ A}}+\underbrace{\sum_{IJ} b_{1IJ}x_Ix_J}_{\text{СТРУКТУРА ДАННЫХ B}}+\underbrace{\sum_{I} c_{1I}x_I}_{\text{СТРУКТУРА ДАННЫХ C}}=y_1\\ \vdots\\ \underbrace{\sum_{IJL} a_{MIJL}x_Ix_Jx_L}_{\text{СТРУКТУРА ДАННЫХ A}}+\underbrace{\sum_{IJ} b_{MIJ}x_Ix_J}_{\text{СТРУКТУРА ДАННЫХ B}}+\underbrace{\sum_{I} c_{MI}x_I}_{\text{СТРУКТУРА ДАННЫХ C}}=y_M\end{cases}$$

Фиг. 24

СПОСОБ СТРУКТУРИРОВАНИЯ №1 (ПРИМЕНИТЕЛЬНО К КУБИЧЕСКОМУ МНОГОЧЛЕНУ G_X)

$$G_x(x_1,\cdots,x_N,y_1,\cdots,y_N)=\begin{cases}\sum_{IJL}\left(a_{1IJL}+a_{1ILJ}+a_{1LJI}\right)y_I y_J x_L+\sum_{I,J}\left(b_{1IJ}+b_{1JI}\right)x_I y_J=z_1\\ \vdots\\ \sum_{IJL}\left(a_{MIJL}+a_{MILJ}+a_{MLJI}\right)y_I y_J x_L+\sum_{IJ}\left(b_{MIJ}+b_{MJI}\right)x_I y_J=z_N\end{cases}$$

СТРУКТУРА ДАННЫХ A

СТРУКТУРА ДАННЫХ B

Фиг. 25

Стр.: 90

RU 2 595 924 C2

СПОСОБ СТРУКТУРИРОВАНИЯ №1 (ПРИМЕНИТЕЛЬНО К КУБИЧЕСКОМУ МНОГОЧЛЕНУ G_y)

$$G_y(x_1,\cdots,x_N,y_1,\cdots,y_N)=\begin{cases}\sum_{IJL}\left(a_{1IJL}+a_{1ILJ}+a_{1LJI}\right)x_I x_J y_L = z_1\\ \vdots\\ \sum_{IJL}\left(a_{MIJL}+a_{MILJ}+a_{MLJI}\right)x_I x_J y_L = z_N\end{cases}$$

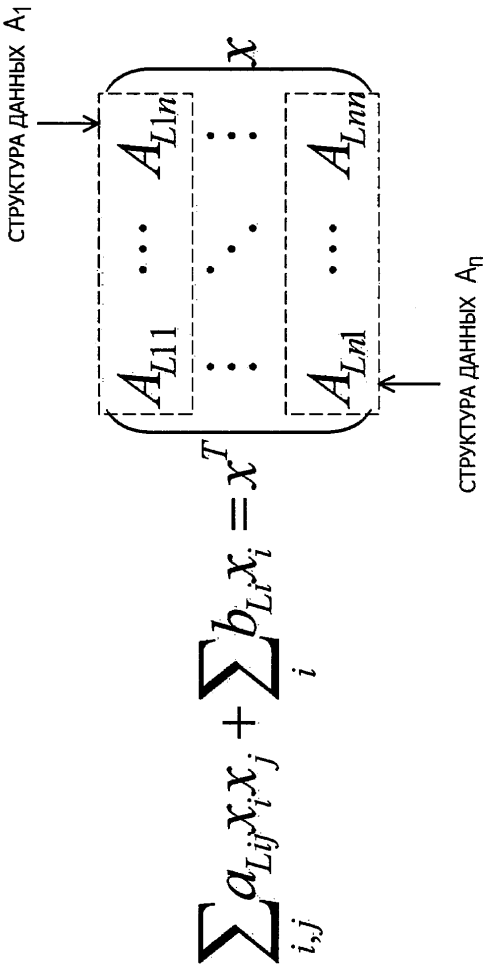
СТРУКТУРА ДАННЫХ

Фиг. 26

Стр.: 91

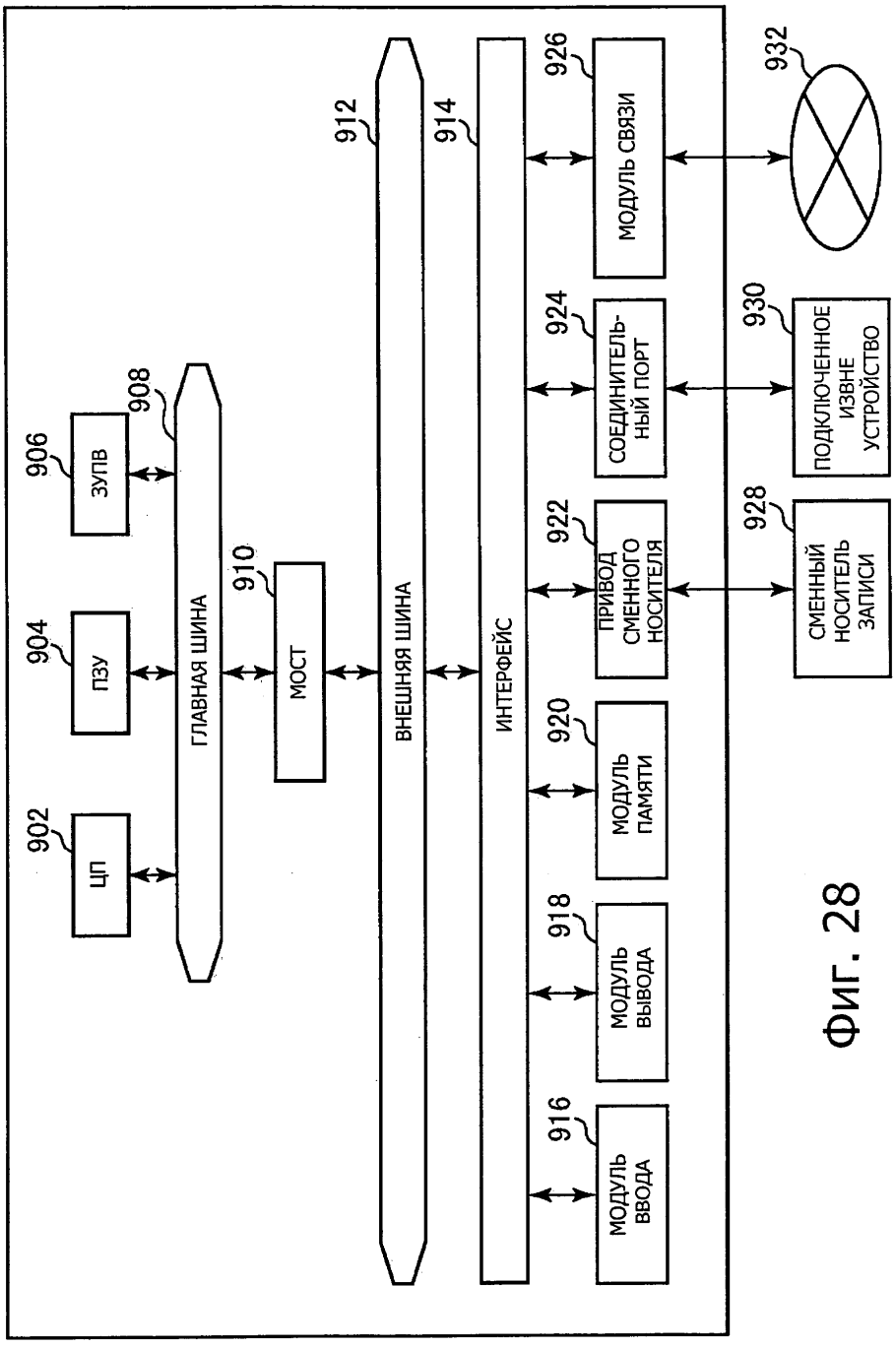
RU 2 595 924 C2

СПОСОБ СТРУКТУРИРОВАНИЯ №2



$$\sum_{i,j} a_{Lij} x_i x_j + \sum_i b_{Li} x_i = x^T \begin{pmatrix} A_{L11} & \cdots & A_{L1n} \\ \vdots & \ddots & \vdots \\ A_{Ln1} & \cdots & A_{Lnn} \end{pmatrix} x$$

Фиг. 27


902
ЦП
904
ПЗУ
906
ЗУПВ
908
ГЛАВНАЯ ШИНА
910
МОСТ
912
ВНЕШНЯЯ ШИНА
914
ИНТЕРФЕЙС
916
МОДУЛЬ ВВОДА
918
МОДУЛЬ ВЫВОДА
920
МОДУЛЬ ПАМЯТИ
922
ПРИВОД СМЕННОГО НОСИТЕЛЯ
924
СОЕДИНИТЕЛЬНЫЙ ПОРТ
926
МОДУЛЬ СВЯЗИ
928
СМЕННЫЙ НОСИТЕЛЬ ЗАПИСИ
930
ПОДКЛЮЧЕННОЕ ИЗВНЕ УСТРОЙСТВО
932

Фиг. 28