МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2023 № 61

Зарегистрирован в Федеральной службе по надзору
в сфере связи и массовых коммуникаций

Свидетельство о регистрации ПИ № ФС 77-33762 от 16 октября 2008 г.

Подписной индекс в объединённом каталоге «Пресса России» 38696

УЧРЕДИТЕЛЬ
Томский государственный университет

Адрес редакции и издателя: 634050, г. Томск, пр. Ленина, 36
E-mail: pank@mail.tsu.ru

В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надёжности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Редактор-переводчик *Т. В. Бутузова*
Верстка *И. А. Панкратовой*

Подписано к печати 18.09.2023. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 14,8. Тираж 300 экз.
Заказ № 5583. Цена свободная. Дата выхода в свет 27.09.2023.

Отпечатано на оборудовании
Издательства Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел.: 8(3822)53-15-28, 52-98-49

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

УДК 512.54; 003.26 DOI 10.17223/20710410/61/1

ДИСКРЕТНЫЕ ДИФФЕРЕНЦИРОВАНИЯ И ИНТЕГРИРОВАНИЯ И ИХ ВОЗМОЖНЫЕ ПРИЛОЖЕНИЯ К АЛГЕБРЕ И КРИПТОГРАФИИ[1]

С. К. Волошин*, В. А. Романьков**

*Омский государственный университет им. Ф. М. Достоевского, г. Омск, Россия
** Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

E-mail: savva.voloshin@gmail.com romankov48@mail.ru

Определяются обобщённые операции дискретных дифференцирования и интегрирования. Приводятся некоторые их свойства. Даётся краткий обзор полученных с использованием этих понятий результатов в алгебре и криптографии. Предлагается новая схема зашифрования сообщения на основе этих операций. Показывается, как их можно использовать для аутентификации и распределения ключа.

Ключевые слова: *дискретные дифференцирование и интегрирование, схема зашифрования, аутентификация, распределение ключа.*

DISCRETE DIFFERENTIATIONS AND INTEGRATIONS AND THEIR POSSIBLE APPLICATIONS TO ALGEBRA AND CRYPTOGRAPHY

S. K. Voloshin*, V. A. Roman'kov**

*Dostoevsky Omsk State University, Omsk, Russia
**Sobolev Institute of Mathematics SB RAS, Omsk, Russia

Generalized operations of discrete differentiation and integration are defined. Some of their properties are given. A brief review of the results obtained earlier with the use of these concepts in algebra and cryptography is given. A new message encryption scheme based on these operations is proposed. We also show how they can be used for authentication and key distribution.

Keywords: *discrete differentiations and integrations, encryption scheme, autentification, key distribution.*

Введение

В настоящее время ведётся интенсивный поиск новых инструментов для разработки криптографических схем и протоколов, в том числе устойчивых к атакам с помощью квантовых компьютеров. В данной работе в качестве такого инструмента предлагаются обобщённые дискретные дифференцирования и интегрирования. Впервые эти

[1]Работа второго автора выполнена в рамках госзадания ИМ СО РАН, проект FWNF–2022–0003.

операции определены авторами в [1] и использованы вторым автором в [2]. До этого было известно только обычное дискретное дифференцирование, применённое в работах [3, 4], в которых впервые доказано существование обычного дискретного интегрирования. Указаны некоторые возможности такого использования. Отмечено, что данные операции находят применение в алгебре.

Далее используются следующие обозначения: $\mathbb{Z}$ — кольцо целых чисел; $\mathbb{Z}_n$ — кольцо вычетов по модулю n; $\mathbb{F}_q$ — конечное поле порядка q; $\mathrm{GL}_s(K)$ — группа обратимых $k\times k$-матриц над кольцом K; $\mathrm{M}_k(K)$ — кольцо $k\times k$-матриц над кольцом K.

1. Определение и основные свойства обобщённых дискретных дифференцирований и интегрирований

Пусть K — произвольное коммутативное кольцо с единицей. Наибольший интерес для наших приложений представляют конечные поля $\mathbb{F}_q$, $q=p^r$, порядка q характеристики p и кольца вычетов вида $\mathbb{Z}_n$, $n=pq$, где p и q — различные (большие) простые числа. Операции обобщённых дискретных дифференцирований и интегрирований определяются на множестве бесконечных двусторонних последовательностей $\bar a=(\ldots,a_{-1},a_0,a_1,\ldots)$ элементов K. Они индуцируют соответствующие операции на односторонних бесконечных последовательностях и на конечных наборах элементов K. Сложение и умножение таких последовательностей задаются покомпонентно.

В общем случае дискретное дифференцирование δ_α определено набором элементов $\alpha=(\alpha_0,\alpha_1,\ldots,\alpha_k)\in K^{k+1}$. По определению

$$\delta_\alpha(\bar a)=(\ldots,b_{-1},b_0,b_1,b_2,\ldots),\ \text{где } b_i=\alpha_0a_i+\alpha_1a_{i+1}+\ldots+\alpha_ka_{i+k},\ i\in\mathbb{Z}. \qquad (1)$$

Обычное дискретное дифференцирование определяется набором $\alpha=(-1,1)$ или, проще говоря, формулой $b_i=a_{i+1}-a_i$, $i\in\mathbb{Z}$. Если понятно из контекста, о каком наборе параметров α идёт речь, или указывается свойство, справедливое для всех параметров, пишем δ, не указывая набор.

Ясно, что δ является аддитивной функцией, то есть для любых $\bar a,\bar b\in K^\infty$ выполнены равенства $\delta(\bar a\pm\bar b)=\delta(\bar a)\pm\delta(\bar b)$.

С любым набором $\alpha=(\alpha_0,\ldots,\alpha_k)$ связан многочлен $f_\alpha(x)=\alpha_0+\alpha_1x+\ldots+\alpha_kx^k$ с коэффициентами из кольца K, и наоборот, любому такому многочлену соответствует набор коэффициентов α, по которому определяется дифференцирование δ_α. Многочлен $f_\alpha(x)$ назовём *определяющим для дифференцирования* δ_α или просто *определяющим*, если ясно, о каком дифференцировании идет речь.

Пусть δ_α и δ_β — два дифференцирования K^∞, отвечающие наборам $\alpha=(\alpha_0,\alpha_1,\ldots,\alpha_k)$ и $\beta=(\beta_0,\beta_1,\ldots,\beta_l)$ соответственно. Непосредственно доказывается, что суперпозиция этих дифференцирований, взятых в любом порядке, является дифференцированием, соответствующим произведению определяющих многочленов $f_\alpha(x)$ и $g_\beta(x)$. Отсюда следует, что для любой последовательности $\bar a\in K^\infty$ справедливо равенство

$$\delta_\beta(\delta_\alpha(\bar a))=\delta_\alpha(\delta_\beta(\bar a)).$$

Другими словами, любые два обобщённых дифференцирования перестановочны между собой. В [1] этот результат отмечен для частного случая конечного поля $\mathbb{F}_q$.

Следующий результат доказан в [1, теорема 1] также для случая конечного поля $\mathbb{F}_q$. Он справедлив для любого коммутативного кольца K с единицей.

Теорема 1 [1]. Пусть дифференцирование δ_α соответствует набору $\alpha=(\alpha_0,\alpha_1,\ldots,\alpha_k)\in K^{k+1}$, у которого элементы α_0 и α_k обратимы в кольце K. Тогда для любой

последовательности $\bar{b} = (\ldots, b_{-1}, b_0, b_1, \ldots) \in K^\infty$ существует такая последовательность $\bar{a} = (\ldots, a_{-1}, a_0, a_1, \ldots) \in K^\infty$, что выполнено равенство

$$\delta_\alpha(\bar{a}) = \bar{b}.$$

Другими словами, любая последовательность $\bar{b} \in K^\infty$ *интегрируема*. Последовательность $\bar{a}$ однозначно определяется набором компонент $(a_0, \ldots, a_{k-1})$, значения которых можно задать произвольно.

Доказательство. Значения $a_0, a_1, \ldots, a_{k-1}$ задаём произвольным образом. Элемент a_k определяем так, чтобы выполнялось равенство $b_0 = \sum\limits_{i=0}^{k} \alpha_i a_i$, а именно:

$$a_k = \alpha_k^{-1} b_0 - \alpha_k^{-1} \sum_{i=0}^{k-1} \alpha_i a_i.$$

Далее последовательно вычисляем элементы a_{k+j}, $j = 1, 2, \ldots$, из соотношений $b_j = \sum\limits_{i=0}^{k} \alpha_i a_{j+i}$:

$$a_{k+j} = \alpha_k^{-1} b_j - \alpha_k^{-1} \sum_{i=0}^{k-1} \alpha_i a_{j+i}. \tag{2}$$

Аналогично вычисляем a_{-1-j} для $j = 0, 1, \ldots$ из соотношений

$$a_{-1-j} = \alpha_0^{-1} b_{-1-j} - \alpha_0^{-1} \sum_{i=1}^{k} \alpha_i a_{-1-j+i}. \tag{3}$$

Утверждение теоремы проверяется непосредственно. ■

Первообразной или *интегралом* $\iota_\alpha(\bar{b})$ последовательности $\bar{b}$ относительно набора α назовём множество всех последовательностей $\bar{a} \in K^\infty$, таких, что $\delta_\alpha(\bar{a}) = \bar{b}$.

Обозначим через $\mathrm{Ann}(\delta_\alpha)$ аннулятор дифференцирования δ_α в K^∞. Ясно, что $\iota_\alpha(\bar{b}) = \bar{a} + \mathrm{Ann}(\delta_\alpha)$, где $\bar{a}$ — любая (частная) последовательность, для которой $\delta_\alpha(\bar{a}) = \bar{b}$. Такая частная последовательность задаётся формулами (2) и (3).

Аналогично обозначаем $\iota_\alpha = \iota_{f_\alpha}$, если по данному многочлену $f_\alpha(x)$ можно определить интегрирование (коэффициенты α_0 и α_k обратимы в K). Конкретный элемент из ι_α однозначно определяется набором $\bar{a}_k = (a_0, a_1, \ldots, a_{k-1}) \in K^{k+1}$, который будем называть набором *констант*. Соответствующее значение по формулам (2) и (3) обозначаем $\iota_{\alpha,\bar{a}_k}$ и называем *определённым интегралом*, отвечающим выбору $\bar{a}_k$. Для любого $\bar{a}_k$ и произвольного набора α, по которому определяется интегрирование, выполнено равенство

$$\delta_\alpha(\iota_{\alpha,\bar{a}_k}(\bar{b})) = \bar{b}.$$

Поэтому мы будем также использовать формулу

$$\delta_\alpha(\iota_\alpha(\bar{b})) = \bar{b}.$$

Другими словами, дифференцирование δ_α определяет правый обратный элемент к интегрированию ι_α. Заметим, что формула

$$\iota_\alpha(\bar{a}_k)(\delta_\alpha(b)) = \bar{b}$$

выполнена только в том случае, когда $\bar{a}_k = \bar{b}_k$, то есть когда набор констант $\bar{a}_k$ совпадает с начальным набором $\bar{b}_k$ последовательности $\bar{b}$.

Замечание 1. Дифференцирование (1) индуцирует отображение любого бесконечного правостороннего интервала $\bar{a}_r(i) = (a_i, a_{i+1}, \ldots)$, $i \in \mathbb{Z}$, результатом которого является правосторонний интервал $\bar{b}_r(i) = (b_i, b_{i+1}, \ldots)$. Более того, областью определения может быть любой конечный интервал $(a_i, a_{i+1}, \ldots, a_{i+k+t})$, $t = 0, 1, 2, \ldots$, длины не меньше чем $k+1$, где k — степень определяющего многочлена. Результатом будет интервал $b_i, b_{i+1}, \ldots, b_{i+t}$ меньшей на величину k длины. Эти отображения будем также называть *дифференцированиями*. Для них выполнено утверждение о перестановочности дифференцирований. В дальнейшем мы не употребляем специальные обозначения для различных типов интервалов.

Интегрирования индуцируют отображения (*интегрирования*) как бесконечных правосторонних последовательностей, так и конечных наборов длины не меньше $k+1$. Считаем, что в этом случае набор начальных констант индексирован как крайний левый набор данной последовательности. Далее рассматриваются только правые интегрирования, по определению не вычисляющие элементы с меньшими индексами, чем у данной последовательности. В этом случае условие обратимости накладывается только на старший коэффициент α_k определяющего многочлена. Результатом такого отображения является соответственно бесконечная правосторонняя последовательность или конечный интервал длины на k больше интегрируемого.

Дифференцирования и интегрирования на нитях и их комбинации с линейными преобразованиями

Пусть по-прежнему K обозначает коммутативное кольцо с единицей. Допустим, задано s последовательностей (нитей) $a_1, \ldots, a_s$ одинаковых типов и равных соответствующих параметров, то есть это либо двусторонние бесконечные последовательности, либо односторонние правонаправленные бесконечные последовательности, либо наборы одинаковой длины. Дифференцирование δ_α и интегрирование i_α действуют одновременно на каждую из этих последовательностей. Считаем, что для интегрирований можно брать различные наборы констант. Пусть φ обозначает линейное преобразование нитей. Его можно задать матрицей A_φ. Далее рассматриваем только невырожденные преобразования, то есть $A_\varphi \in \mathrm{GL}_s(K)$. Очевидно, что операции дифференцирования и интегрирования перестановочны с линейными преобразованиями нитей. Суперпозицию дифференцирования и линейного преобразования обозначаем через $\delta_{\alpha,\varphi}$ и называем *скрученным дифференцированием*. Аналогично вводится понятие *скрученного интегрирования* $\iota_{\alpha,\varphi}$. Имеет место формула

$$\delta_{\alpha,\varphi^{-1}}(\iota_{\alpha,\varphi}(\bar{b})) = \bar{b}.$$

2. Возможные применения дискретных дифференцирований и интегрирований в алгебре

Понятия дискретных дифференцирования и интегрирования можно определить на произвольной группе G, в том числе некоммутативной. Они существенно использовались для получения основных результатов работы [4]. Отметим также, что в [5] неявно доказано существование первообразной.

При этом используется мультипликативная запись операции в группе G. Определяющий многочлен $f_\alpha(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_k x^k$ берётся с коэффициентами из кольца целых чисел $\mathbb{Z}$. Дифференцирование определяется формулой

$$\delta_\alpha(\bar{a}) = (\ldots, b_{-1}, b_0, b_1, b_2, \ldots), \quad b_i = a_i^{\alpha_0} a_{i+1}^{\alpha_1} \cdots a_{i+k}^{\alpha_k}, \quad i \in \mathbb{Z}.$$

Обычное (не обобщённое) дискретное дифференцирование определяется, как и в случае кольца, набором $\alpha = (-1, 1)$ или формулой $b_i = a_i^{-1} a_{i+1}$, $i \in \mathbb{Z}$. Интегрирование определено для любого определяющего многочлена с обратимыми коэффициентами α_0 и α_k формулой

$$\iota_\alpha(\bar{b}) = (\ldots, a_{-1}, a_0, a_1, a_2, \ldots),$$

где $\bar{a}_k = (a_0, a_1, \ldots, a_{k-1}) \in G^k$ — произвольный набор констант;

$$a_{k+j} = (a_{k+j-1}^{-\alpha_{k-1}} a_{k+j-2}^{-\alpha_{k-2}} \ldots a_{k+j}^{-\alpha_0} b_{k+j})^{\alpha_k^{-1}}, \quad j \in \mathbb{N} \cup \{0\};$$
$$a_{-j} = (a_{-j+1}^{\alpha_1} a_{-j+2}^{\alpha_2} \ldots a_{-j+k}^{\alpha_k} b_{-j})^{\alpha_0^{-1}}, \quad j \in \mathbb{N}.$$

3. Возможные применения дискретных дифференцирований и интегрирований в криптографии

В работах [6–13] и ряде других публикаций (см. библиографию в [9]) представлены методы криптографического анализа, показавшие уязвимость всех основных схем алгебраической криптографии. Поэтому актуален поиск новых криптографических инструментов для создания таких схем. В этом направлении ведётся и настоящее исследование.

На основе новых понятий обобщённых дискретных дифференцирования и интегрирования в работах [2, 3] предложена новая схема скрытого компактного хранения данных группы пользователей в общей открытой базе в виде таблицы. Компонентами таблицы служат элементы коммутативного кольца с единицей K, кодирующие данные. База не имеет подразделов, относящихся к данным индивидуальных пользователей. Соответствующая таблица является покомпонентной суммой индивидуальных таблиц, построенных определённым алгоритмом. Каждый из пользователей может извлечь из базы свои данные с помощью индивидуального ключа. Ключ выдаётся в момент регистрации пользователя в системе, когда создаётся таблица, полученная на основе его данных. Ключ представляет собой пару многочленов с коэффициентами из K с обратимыми старшими коэффициентами. Построение таблицы и алгоритмы извлечения из неё своих данных индивидуальными пользователями осуществляются эффективно. В то же время конкретный пользователь не имеет возможности получить данные других пользователей. Потенциальный нарушитель не может получить никаких данных. Схема позволяет изменять и удалять данные без замены ключей. Предполагается свободный доступ к базе данных. Возможно многократное использование ключей, что является основным достоинством схемы.

3.1. Передача зашифрованного сообщения

Алиса хочет передать сообщение, представленное конечной последовательностью s наборов $\bar{A} = (\bar{a}_1, \ldots, \bar{a}_s)$, где $\bar{a}_j = (a_{j,0}, \ldots, a_{j,l})$, фиксированной длины $l+1$ с элементами из коммутативного кольца с единицей K.

Перемешивание нитей проводится Алисой и Бобом и осуществляется умножением на матрицы $P_A, P_B \in \mathrm{GL}(K)$ соответственно. На последующих шагах корреспонденты используют обратные преобразования с матрицами P_A^{-1}, P_B^{-1} соответственно. Матрицы должны быть перестановочны между собой. Поэтому корреспонденты сначала договариваются о множестве M попарно перестановочных матриц из $\mathrm{GL}_s(K)$. Самым распространённым способом служит выбор матрицы T и определение в качестве M множества значений всех многочленов вида $u(x) \in K[x]$ от T. Тогда в алгоритме корреспонденты выбирают случайные относительно равномерного распределения обратимые матрицы $P(A)$ и $P(B)$ из M.

При выборе в качестве K конечного поля $\mathbb{F}_q$ вероятность того, что случайная матрица окажется обратимой, при относительно малом s по отношению к q близка к 1. Объясним это. В [14, Lemma 9 (Invertibility Lemma)] доказано следующее утверждение: Пусть для произвольного поля $\mathbb{F}$ матрицы $T_0, T_1, \ldots, T_r \in M_k(\mathbb{F})$ обладают тем свойством, что их линейная оболочка содержит матрицу из $\mathrm{GL}_k(\mathbb{F})$; S — конечное подмножество в $\mathbb{F}$. Если $\alpha_1, \ldots, \alpha_r$ выбираются равномерно и независимо из S, то вероятность того, что матрица $\alpha_1 T_1 + \ldots + \alpha_r T_r$ обратима, не меньше чем $1 - k/|S|$.

В нашем случае полагаем $\mathbb{F} = \mathbb{F}_q, S = \mathbb{F}_q$, $T_0 = T^0 = E$, $T_1 = T$, $\ldots$, $T_{s-1} = T^{s-1}$. По теореме Кронекера — Капелли бóльшие, чем $s-1$, степени матрицы T линейно выражаются через выписанные степени, поэтому M является линейной оболочкой данного множества матриц. По лемме вероятность случайного выбора из M обратимой матрицы не меньше чем $1 - s/q$.

Следующий протокол представляет схему передачи зашифрованного сообщения от Алисы к Бобу:

— Алиса выбирает параметр k и многочлен $f_\alpha(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_k x^k \in K[x]$, $\alpha_0, \alpha_k \in K^*$, определяющий операции дифференцирования и интегрирования. Эти данные секретны.

— Подобным образом поступает и Боб, выбирая параметр m и определяя свои операции многочленом $g_\beta(x) = \beta_0 + \beta_1 x + \ldots + \beta_m x^m \in K[x]$, $\beta_0, \beta_m \in K^*$. Эти данные также секретны. Ключом в данном протоколе служит пара $Key = (f_\alpha(x), g_\beta(x))$.

— Алиса выбирает набор нитей C, состоящий из s наборов констант $\bar{c}_j = (c_{j,0}, \ldots, c_{j,l})$, $j = 1, \ldots, s$, интегрирует вправо в соответствии с этим набором покомпонентно $\bar{A}$, получая $\iota_{\alpha,\bar{C}}(\bar{A})$. Затем она выбирает параметр r и случайную последовательность A' наборов $\bar{a}'_j = (a_{j,-r}, \ldots, a_{j,-1})$, $j = 1, \ldots, s$, и дописывает эти наборы перед соответствующими наборами из $\iota_{\alpha,\bar{C}}(A)$, получая последовательность нитей $(A'||\iota_{\alpha,\bar{C}}(A))$. Далее она выбирает случайную матрицу $P_A \in M$, применяет линейное преобразование с этой матрицей к полученным нитям и пересылает Бобу результат

$$\tilde{A} = (\bar{b}_1, \ldots, \bar{b}_s)$$

— последовательность из s наборов длины $l + 1 + r + k$.

— Боб выбирает набор нитей D, состоящий из s наборов констант $\bar{d}_j = (d_{j,0}, \ldots, d_{j,l})$, $j = 1, \ldots, s$, интегрирует вправо в соответствии с этим набором покомпонентно $\tilde{A}$, получая $\iota_{\beta,\bar{D}}(\tilde{A})$. Далее Боб выбирает параметр t и случайную последовательность $\bar{A}''$ наборов $\bar{a}''_j = (a_{j,-r-t}, \ldots, a_{j,-r-1})$, $j = 1, \ldots, s$, дописывает эти наборы в начало последовательности, получая

$$(A''||\iota_{\beta,\bar{D}}(\tilde{A}).$$

Далее он выбирает случайную матрицу $P_B \in M$, применяет линейное преобразование с этой матрицей к полученным нитям и пересылает Алисе результат

$$\tilde{B} = (f_1, \ldots, f_s)$$

— последовательность наборов длины $l + 1 + r + t + k + m$.

— Алиса удаляет из каждой нити начальный набор из r компонент, дифференцирует покомпонентно полученную последовательность, применяет к нитям линейное преобразование с матрицей P_A^{-1} и передаёт Бобу результат

$$\tilde{A}' = (\bar{g}_0, \ldots, \bar{g}_s)$$

— последовательность наборов длины $l + 1 + t + m$.

— Боб удаляет из каждой нити начальный набор из t компонент, дифференцирует покомпонентно полученную последовательность, применяет к нитям линейное преобразование с матрицей P_B^{-1} и получает сообщение Алисы $\bar{A}$.

Корректность вычислений обусловлена тем, что линейные преобразования перестановочны с дифференцированиями и интегрированиями, дифференцирования перестановочны, а используемые линейные преобразования также перестановочны между собой. Дописывание в начало случайных интервалов с последующим удалением интервалов той же длины не изменяет полученного результата, так как начальные значения при дифференцировании и правом интегрировании не влияют на вычисление последующих компонент. В то же время такая операция затрудняет определение в продифференцированном или проинтегрированном наборе нитей позиции, с которой произведено данное действие. Это также затрудняет применение метода Гаусса.

Основания стойкости зашифрования. В предлагаемом протоколе не содержится механизма аутентификации, поэтому он не защищён от атаки «противник посередине». Для противодействия данной атаке требуются дополнительные средства.

Передаваемое сообщение может быть прочитано взломщиком, если ему удастся найти хотя бы один из многочленов $f_\alpha(x)$ или $g_\beta(x)$. Прежде всего рассмотрим ситуацию, когда известен конечный набор $\bar{a} = (a_0, a_1, \ldots, a_{l+n}$ и результат дифференцирования $i_\alpha(\bar{a}) = (b_0, b_1, \ldots, b_l)$. Эти данные позволяют определить степень k определяющего многочлена $f_\alpha(x)$. Далее можно вычислить коэффициенты этого многочлена, решая систему линейных уравнений

$$\begin{cases} \alpha_0 a_0 + \alpha_1 a_1 + \ldots + \alpha_k a_k = b_0, \\ \alpha_0 a_1 + \alpha_1 a_2 + \ldots + \alpha_k a_{k+1} = b_1, \\ \ldots \\ +\alpha_0 a_{l-k} + \alpha_1 a_{l-k+1} \ldots + \alpha_k a_l = b_l. \end{cases}$$

Вычислительная сложность алгоритма Гаусса, основного метода решения системы из l линейных уравнений, составляет $O(l^3)$. Известные усовершенствования метода Гаусса не дают существенного уменьшения сложности вычислений. В то же время для вычисления компонент интервала длины l при известном определяющем многочлене требуется произвести $O(l)$ операций.

Рекомендуется использовать определяющие многочлены $f_\alpha(x)$ и $g_\beta(x)$ с разреженными множествами ненулевых коэффициентов. При этом сокращается время выполнения необходимых дифференцирований и интегрирований. Аналогичные рассуждения проходят для интегрирований.

В предложенном протоколе описанная ситуация не возникает из-за использования линейных преобразований и дописывания случайных наборов в начало передаваемых последовательностей с последующим удалением начальных наборов той же длины.

3.2. Аутентификация

Приведём описание протокола для схемы, основанной на дискретных дифференцировании и интегрировании. Аутентифицируется Алиса, проверяет Боб.

— Открытые данные Алисы состоят из коммутативного кольца с единицей K, последовательности из s нитей $\bar{A} = (\bar{a}_1, \ldots, \bar{a}_s)$, где $\bar{a}_j = (a_{j,0}, \ldots, a_{j,l}) \in K^{l+1}$, и её значений $\bar{B} = (\delta_{\alpha,\varphi}(A)$ при скрученном дифференцировании с секретными данными — определяющим многочленом $f_\alpha(x) \in K[x]$ и линейным преобразованием нитей φ, заданным матрицей $P_\varphi \in \mathrm{GL}_s(K)$. Эти данные должны либо быть подтверждены

сертификатом удостоверяющего центра, либо переданы Бобу в процессе предварительной начальной регистрации.

— При запросе Боба об аутентификации Алиса выбирает определяющий многочлен $g_\beta(x) \in K[x]$, линейное преобразование ψ, заданное матрицей $P_\psi \in \mathrm{GL}_s(K)$, вычисляет и предъявляет Бобу набор нитей $\bar{C} = \delta_{\beta,\psi}(\bar{B})$.

— Боб посылает Алисе метку c, равную 0 или 1.

— Если $c = 0$, Алиса посылает в ответ параметры скрученного дифференцирования $\delta_{\beta,\psi}$, Боб вычисляет $\bar{C}' = \delta_{\beta,\psi}(B)$ и проверяет справедливость равенства $\bar{C}' = \bar{C}$. Этот раунд аутентификации Алиса проходит, если равенство справедливо.

— Если $c = 1$, Алиса посылает в ответ параметры скрученного дифференцирования $\delta_{\gamma,\phi}$ — суперпозиции скрученных дифференцирований $\delta_{\alpha,\varphi}$ и $\delta_{\beta,\psi}$, Боб вычисляет $\bar{A}' = \delta_{\gamma,\phi}(A)$ и проверяет справедливость равенства $\bar{A}' = \bar{C}$. Этот раунд аутентификации Алиса проходит, если равенство справедливо.

— Подобно алгоритму аутентификации Фиата — Шамира (см., например, [12, с. 408] или [4, с. 143]) противник, не зная параметров скрученного дифференцирования $\delta_{\alpha,\varphi}$, может пройти раунд с $c = 1$, если специальным образом подготовится, а именно: вместо $\bar{C}$ он пошлёт значение $\delta_{\lambda,\xi}(\bar{A})$ для случайных параметров λ и ξ, которые он сможет предъявить при метке $c = 1$. Но тогда он не сможет аутентифицироваться при метке $c = 0$. Если он ожидает метку $c = 0$, то он просто выбирает свои параметры в качестве β и ψ, которые предъявляет при метке $c = 0$. Но тогда он не сможет пройти аутентификацию при метке $c = 0$. Таким образом, противник аутентифицируется с вероятностью $1/2$. Вероятность прохождения q раундов аутентификации равна $(1/2)^q$ и её можно сделать сколь угодно малой за счёт увеличения q.

Приведённая схема аутентификации является схемой с нулевым разглашением: в процессе не раскрывается главный секрет — параметры α и φ. Стойкость схемы основывается на трудноразрешимости задачи вычисления параметров скрученного дифференцирования.

3.3. Распределение ключа

Опишем протокол для схемы распределения ключа типа Диффи — Хеллмана с использованием скрученных дифференцирований. Сначала корреспонденты Алиса и Боб договариваются о выборе коммутативного кольца с единицей K и параметра s. Эти данные открыты. Корреспонденты договариваются также о выборе параметра l и конкретной последовательности из s нитей $\bar{A} = (\bar{a}_1, \ldots, \bar{a}_s)$, где $\bar{a}_j = (a_{j,0}, \ldots, a_{j,l}) \in K^{l+1}$. Эти данные также открыты.

— Алиса выбирает параметр k и многочлен $f_\alpha(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_k x^k \in K[x]$, определяющий операцию дифференцирования. Также она выбирает линейное преобразование нитей φ, заданное матрицей $P_A \in \mathrm{GL}_s(K)$. Эти данные секретны. Затем Алиса вычисляет и передаёт Бобу значение

$$K_A = \delta_{\alpha,\varphi}(\bar{A}).$$

— Боб выбирает параметр s и многочлен $g_\beta(x) = \beta_0 + \beta_1 x + \ldots + \alpha_m x^m \in K[x]$, определяющий операцию дифференцирования. Также она выбирает линейное преобразование нитей ψ, заданное матрицей $P_B \in \mathrm{GL}_s(K)$. Эти данные секретны. Затем Боб вычисляет и передаёт Алисе значение

$$K_B = \delta_{\beta,\psi}(\bar{A}).$$

— Алиса вычисляет распределенный ключ

$$K = \delta_{\alpha,\varphi}(K_B).$$

— Боб вычисляет распределенный ключ

$$K = \delta_{\beta,\psi}(K_A).$$

Корректность схемы (равенство значений, полученных на двух последних шагах) определяется перестановочностью операций дифференцирования. Стойкость схемы основывается на трудноразрешимости задачи вычисления параметров скрученного дифференцирования.

Заключение

В работе представлены определения дискретных операций дифференцирования и интегрирования последовательностей элементов произвольного коммутативного кольца с единицей, а также их скрученные версии на наборах таких последовательностей (нитях), использующие линейные преобразования нитей. Определены основные свойства этих операций. Дан краткий обзор их применений. В частности, предложены схемы передачи сообщения, распределения ключа и аутентификации с использованием этих операций.

ЛИТЕРАТУРА

1. *Волошин С. К., Романьков В. А.* Обобщенные дискретные операции дифференцирования и интегрирования // Вестник Омского университета. 2021. Т. 26. № 4. С. 4–8.
2. *Романьков В. А.* Обобщённая схема скрытого компактного хранения данных различных пользователей в общей открытой базе // Известия Иркутского государственного университета. Сер. Математика. 2022. Т. 20. С. 1–14.
3. *Романьков В. А.* О скрытом компактном способе хранения данных // Прикладная дискретная математика. Приложение. 2020. № 13. С. 56–59.
4. *Roman'kov V.* Embedding theorems for solvable groups // Proc. AMS. 2021. V. 149. P. 4133–4143.
5. *Neumann P. M.* On the structure of standard wreath products of groups // Math. Z. 1964. V. 84. P. 343–373.
6. *Романьков В. А.* Криптографический анализ некоторых схем шифрования, использующих автоморфизмы // Прикладная дискретная математика. 2013. № 3(21). С. 35–51.
7. *Myasnikov A. and Roman'kov V.* A linear decomposition attack // Groups Complexity Cryptology. 2015. V. 7. No. 1. P. 81–94.
8. *Roman'kov V. A.* A nonlinear decomposition attack // Groups Complexity Cryptologogy. 2017. V. 8. No. 2. P. 197–207.
9. *Roman'kov V.* Essays in algebra and cryptology: Algebraic cryptanalysis. Omsk: Omsk State University, 2018. 208 p.
10. *Романьков В. А.* Алгебраическая криптология. Омск: ОмГУ, 2020. 261 с.
11. *Ben-Zvi A., Kalka A., and Tsaban B.* Cryptanalysis via algebraic spans // Proc. 38th Ann. Intern. Cryptology Conf. Santa Barbara, CA, USA, 2018. Part 1. P. 255–274.
12. *Menezes A. J., Oorschot P. C., and Vanstone S.* Handbook of Applied Cryptography. Boca Raton: CRC Press, 1996. 816 p.
13. *Романьков В. А.* Введение в криптографию. Курс лекций. М.: Форум, 2012. 239 с.
14. *Tsaban B.* Polynomial time solutions of computational problems in noncommutative algebraic cryptography // J. Cryptology. 2015. V. 28. No. 3. P. 601–622.

REFERENCES

1. *Voloshin S. K. and Roman'kov V. A.* Obobshchennye diskretnye operatsii differentsirovaniya i integrirovaniya [Generalized discrete operations of differentiation and integration.] Vestnik OmSU, 2021, vol. 26, no. 4, pp. 4–8. (in Russian)
2. *Roman'kov V. A.* Obobshchennaya skhema skrytogo kompaktnogo khraneniya dannykh razlichnykh pol'zovateley v obshchey otkrytoy baze [Generalized scheme of hidden compact storage of data of various users in a common open database]. Izvestiya ISU. Matematika, 2022, vol. 20, pp. 1–14. (in Russian)
3. *Roman'kov V. A.* O skrytom kompaktnom sposobe khraneniya dannykh [About the hidden compact way to store data]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2020, no. 13, pp. 56–59. (in Russian)
4. *Roman'kov V.* Embedding theorems for solvable groups. Proc. AMS, 2021, vol. 149, pp. 4133–4143.
5. *Neumann P. M.* On the structure of standard wreath products of groups. Math. Z., 1964, vol. 84, pp. 343–373.
6. *Roman'kov V. A.* Kriptograficheskiy analiz nekotorykh skhem shifrovaniya, ispol'zuyushchikh avtomorfizmy [Cryptanalysis of some schemes applying automorphisms]. Prikladnaya Diskretnaya Matematika, 2013, no. 3(21), pp. 35–51. (in Russian)
7. *Myasnikov A. and Roman'kov V.* A linear decomposition attack. Groups Complexity Cryptology, 2015, vol. 7, no. 1, pp. 81–94.
8. *Roman'kov V. A.* A nonlinear decomposition attack. Groups Complexity Cryptology, 2017, vol. 8, no. 2, pp. 197–207.
9. *Roman'kov V.* Essays in Algebra and Cryptology: Algebraic Cryptanalysis. Omsk, OmskSU Publ., 2018. 208 p.
10. *Roman'kov V. A.* Algebraicheskaya Kriptologiya [Algebraic Cryptology]. Omsk, OmSU Publ., 2020. 261 p. (in Russian)
11. *Ben-Zvi A., Kalka A., and Tsaban B.* Cryptanalysis via algebraic spans. Proc. 38th Ann. Intern. Cryptology Conf., Santa Barbara, CA, USA, 2018, part 1, pp. 255–274.
12. *Menezes A. J., Oorschot P. C., and Vanstone S.* Handbook of Applied Cryptography. Boca Raton, CRC Press, 1996. 816 p.
13. *Roman'kov V. A.* Vvedenie v kriptografiyu. Kurs lektsiy [Introduction to Cryptography. Lecture Course]. Moscow, Forum, 2012. 239 p. (in Russian)
14. *Tsaban B.* Polynomial time solutions of computational problems in noncommutative algebraic cryptography. J. Cryptology, 2015, vol. 28, no. 3, pp. 601–622.

УДК 519.7 DOI 10.17223/20710410/61/2

NONLINEARITY OF APN FUNCTIONS: COMPARATIVE ANALYSIS AND ESTIMATES

V. G. Ryabov

NP "GST", Moscow, Russia

E-mail: 4vryabov@gmail.com

The main results of the paper relate to the nonlinearity of APN functions defined for a vectorial Boolean function as the Hamming distance from it to the set of affine mappings in the space of images of all vectorial Boolean functions in fixed dimension. For APN functions in dimension n, the lower nonlinearity bound of the form $2^n - \sqrt{2^{n+1} - 7 \cdot 2^{-2}} - 2^{-1}$ and the corresponding lower bound on the affinity order are obtained. The exact values of the nonlinearity of all APN functions up to dimension 5 are found, and also for one known APN 6-dimensional permutation and for all differentially 4-uniform permutations in dimension 4.

Keywords: *vectorial Boolean function, permutation, APN function, EA-equivalence, nonlinearity, differentially uniform.*

НЕЛИНЕЙНОСТЬ APN-ФУНКЦИЙ: СРАВНИТЕЛЬНЫЙ АНАЛИЗ И ОЦЕНКИ

В. Г. Рябов

НП «ГСТ», г. Москва, Россия

Нелинейность APN-функции определяется как расстояние Хэмминга от неё до множества аффинных отображений в пространстве значений векторных булевых функций фиксированной размерности. Для APN-функций размерности n получены нижняя граница нелинейности вида $2^n - \sqrt{2^{n+1} - 7 \cdot 2^{-2}} - 2^{-1}$ и соответствующая ей нижняя граница порядка аффинности. Найдены точные значения нелинейности всех APN-функций размерности, не превосходящей 5, а также для одной известной APN-подстановки размерности 6 и для всех дифференциально 4-равномерных подстановок размерности 4.

Ключевые слова: *векторная булева функция, подстановка, APN-функция, EA-эквивалентность, нелинейность, дифференциальная равномерность.*

1. Introduction

Denote by $\mathbb{F}_2^n$ the n-dimensional vector space over the two-element field $\mathbb{F}_2$, where n is a natural number, and by $P_2^{n,k}$ the set of all mappings of the space $\mathbb{F}_2^n$ into the space $\mathbb{F}_2^k$. The mapping $F \in P_2^{n,k}$ is called a vectorial Boolean function or simply a vectorial function, implying the Boolean case, and in the case $k = 1$ we will use similar terms without the adjective "vectorial". The subset of one-to-one mappings from $P_2^{n,n}$, called permutations, is denoted by S_2^n.

Any vectorial Boolean function is uniquely determined by an ordered set of coordinate Boolean functions. In turn, each coordinate function can be represented by a polynomial

of n variables over the field $\mathbb{F}_2$. For a vectorial function $F \in P_2^{n,k}$, the algebraic degree of nonlinearity $\deg F$ is usually defined as the maximum degree of the polynomials representing its coordinate functions. Under the condition $\deg F \leqslant 1$ the mapping F is affine. Denote by $A_2^{n,k}$ the subset of all affine mappings from the set $P_2^{n,k}$.

As noted in [1], two approaches to the definition of the nonlinearity of vectorial functions have become widespread. The first approach is based on using the Hamming distance. The Hamming distance from the function $f \in P_2^{n,1}$ to the set $A_2^{n,1}$ in the space $\mathbb{F}_2^{2^n}$, called its nonlinearity, is denoted by N_f. In [2], with an orientation towards the linear method of cryptanalysis, the nonlinearity of the vectorial function $F \in P_2^{n,k}$ with a set of coordinate functions $\boldsymbol{f} = (f_1, \ldots, f_k)$ is defined by the formula

$$NL_F = \min_{\boldsymbol{w} \in \mathbb{F}_2^k \setminus \{\mathbf{0}\}} N_{\langle \boldsymbol{w}, \boldsymbol{f} \rangle}, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of vectors, that is, it is the minimum of the nonlinearities of all nonzero linear combinations of coordinate functions of the mapping F. The Boolean case allows to give an equivalent definition of the nonlinearity of a vectorial function using the maximum absolute value of the Walsh — Hadamard transform coefficients of all nonzero linear combinations of its coordinate functions.

The second approach to determining the nonlinearity of the vectorial function F, associated with the differential method of cryptanalysis, is to compare for all possible $\alpha \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}$ and $\beta \in \mathbb{F}_2^k$ the cardinalities of subsets of variables for which the directed derivative satisfies the condition

$$D_\alpha F(\boldsymbol{x}) = F(\boldsymbol{x} \oplus \alpha) \oplus F(\boldsymbol{x}) = \beta, \tag{2}$$

where $\oplus$ is the addition operation in the corresponding space. Since in the Boolean case the equality $D_\alpha F(\boldsymbol{x}) = D_\alpha F(\boldsymbol{x} \oplus \alpha)$ is true, all elements of this spectrum have an even value. For $F \in P_2^{n,k}$, the value

$$\triangle_F = \max_{\substack{\alpha \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}, \\ \beta \in \mathbb{F}_2^k}} \big|\{\boldsymbol{x} : D_\alpha F(\boldsymbol{x}) = \beta\}\big|,$$

is considered in this approach as an indicator of nonlinearity. A mapping $F \in P_2^{n,k}$ for which the condition $\triangle_F \leqslant \delta$ is satisfied is called a differentially δ-uniform [3], and in the case $k = n$ and $\delta = 2$ it is called almost perfect nonlinear or APN function [4].

At the same time, within the framework of the first approach, one more indicator of the nonlinearity of the vectorial function can be naturally determined. Taking into account the isomorphism of the Abelian groups of the vector space $\mathbb{F}_2^k$ and the field $\mathbb{F}_{2^k}$, the classical Hamming distance in space $\mathbb{F}_{2^k}^{2^n}$ can be used to measure the remoteness of the functions F_1 and F_2 from $P_2^{n,k}$. Let's denote this distance by $\rho(F_1, F_2)$. For a vectorial function $F \in P_2^{n,k}$ let's define the nonlinearity indicator N_F using the formula

$$N_F = \min_{A \in A_2^{n,k}} \rho(F, A). \tag{3}$$

In [5–7] this indicator was called the second type of nonlinearity, and in [8] — the vectorial nonlinearity.

For $k = 1$, the nonlinearity indicators in the sense (1) and in the sense (3) are the same. However, starting from $k = 2$, they differ significantly. The indicator N_F also plays

an important role in cryptography and coding [7]. In particular, it is more relevant for the analysis of methods using multi-dimensional affine approximations of Boolean vectorial functions. For example, it can be used to get the lower bound on the minimum number of affinity domains in an arbitrary piecewise affine representation of a vectorial Boolean function, which in the domestic cryptographic literature is referred to as the affinity order and denoted by $\operatorname{ard} F$. Indeed, it is easy to see that the affinity order of the vectorial function $F \in P_2^{n,k}$ satisfies the inequality

$$\operatorname{ard} F \geqslant \frac{2^n}{2^n - N_F}. \tag{4}$$

Moreover, unlike the characteristics NL_F and $\triangle_F$, the indicator N_F is a metric, which makes it possible to speak mathematically correctly about the remoteness of a vectorial function from affine ones. In this regard, in [9, 10], relating to the case of arbitrary finite fields, the indicator of the form N_F was called the nonlinearity of the mapping F.

The nonlinearity in the sense (1) for APN functions has been studied by many authors. Here it is necessary to highlight the papers of C. Carlet (see, for example, [11–16]). For a vectorial function $F \in P_2^{n,n}$, the Sidelnikov — Chabaud — Vaudenay inequality implies an upper bound on the nonlinearity in the sense (1), namely

$$NL_F \leqslant 2^{n-1} - 2^{(n-1)/2}. \tag{5}$$

This bound is reached only for odd n for the so-called almost bent or AB functions. All AB functions are APN functions. The converse is not true in general, but it is true in particular case of odd n for quadratic functions. For other currently known APN functions, including the case of even n, the largest value of nonlinearity in the sense (1) is $2^{n-1} - 2^{n/2}$. Also of interest are the lower bounds given in [16], namely, $NL_F \geqslant 2^{n-1} - 2^{(3n-3)/4}$ for odd n and $NL_F \geqslant 2^{n-1} - 2^{(3n-2)/4}$ for even n. At the same time, there are a number of open problems regarding nonlinearity in the sense (1) for APN functions [13].

The nonlinearity in the sense (3) for APN functions has been studied to a lesser extent. From the results of [7] for a vectorial function $F \in P_2^{n,k}$ follows a chain of inequalities of the form $0 \leqslant NL_F \leqslant N_F \leqslant 2^n - 2^{n-k} - 1$. In [15][1], another upper bound of the form

$$N_F \leqslant 2^n - n - 1 \tag{6}$$

is obtained (for $k \leqslant 2n - 5$ or $k = n = 4$, a strict inequality holds). Using estimates of the size of the image set, the lower bound on the indicator N_F for differentially δ-uniform vectorial functions from $P_2^{n,k}$ of the form

$$N_F \geqslant 2^n - \sqrt{2^n + \delta\ (2^n - 1)} \tag{7}$$

is also obtained there, from which the lower bound on this indicator follows for all APN functions in dimension n of the form

$$N_F \geqslant 2^n - \sqrt{3 \cdot 2^n - 2}. \tag{8}$$

At the same time, the study of the behavior of nonlinearity in the sense (3) of vectorial Boolean functions, including APN functions, needs to be continued, which was, in particular, indicated in the open problem 11 of the eighth international Olympiad in cryptography

[1] In [15], as applied to the indicator N_F, the term nonlinearity is not used.

NSUCRYPTO2021 [17]. In the footsteps of solving this problem using estimates of the size of the Sidon set, G. P. Nagy at the end of 2022 posted material with new lower bounds on the Internet [8]. Its lower bound on the indicator N_F for differentially δ-uniform vectorial functions from $P_2^{n,k}$ has the form

$$N_F \geqslant 2^n - \sqrt{\delta \cdot 2^n} - 2^{-1},$$

from which the lower bound on this indicator follows for all APN functions in dimension n of the form

$$N_F \geqslant 2^n - \sqrt{2^{n+1}} - 2^{-1}.$$

This paper is devoted to the study of the nonlinearity of APN functions in the sense (3). In what follows, unless otherwise stated, by the nonlinearity of a vectorial function we mean the indicator N_F. The main task is to refine the bounds on the nonlinearity of the APN functions and find its exact values for the APN functions in small dimension ($n \geqslant 5$), as well as to compare the behavior of N_F with $\deg F$ and NL_F for such mappings. In parallel and independently of the studies of G. P. Nagy, without resorting to estimates of the size of the Sidon set, the author has obtained a lower bound on the nonlinearity of APN functions, which is presented in Section 2. A lower bound on the affinity order that follows from it is also given. In Section 3, the exact values of the nonlinearity for all APN functions up to dimension 5, as well as for one known APN 6-dimensional permutation, are found. Since none of the 4-dimensional permutations is an APN, in Section 4 the case of differentially 4-uniform permutations in dimension 4 is considered. In Section 5, open problems and conjectures related to the behavior of the nonlinearity of APN functions are presented.

2. Boundaries on nonlinearity of APN functions

In [18], the following necessary and sufficient condition for a Boolean vectorial function to be an APN was first obtained.

Proposition 1 [18]. Let a vectorial function $F \in P_2^{n,n}$. Then F is APN if and only if there is no 2-dimensional linear manifold[2] in the space of the domain of F on which the mapping F coincides with some affine one.

In [19] this condition is used as an alternative definition of APN functions. There are other formulations of this condition, for example, for pairwise distinct variables $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4 \in \mathbb{F}_2^n$, if the equality $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus \boldsymbol{x}_3 \oplus \boldsymbol{x}_4 = 0$ holds, then the inequality $F(\boldsymbol{x}_1) \oplus F(\boldsymbol{x}_2) \oplus F(\boldsymbol{x}_3) \oplus F(\boldsymbol{x}_4) \neq 0$ is true.

Theorem 1. Let F be the APN function in dimension n. Then the following inequality is true for its nonlinearity:

$$N_F \geqslant 2^n - \sqrt{2^{n+1} - 7 \cdot 2^{-2}} - 2^{-1}. \tag{9}$$

Proof. Let's prove the theorem by contradiction, assuming that the inequality

$$N_F < 2^n - \sqrt{2^{n+1} - 7 \cdot 2^{-2}} - 2^{-1} \tag{10}$$

is true. It follows from the definition of nonlinearity that there is at least one affine mapping $A \in A_2^{n,n}$ with which the vectorial function F coincides on $2^n - N_F$ variables of the domain of F and A. Let $\mathbf{C}_{F,A} = \{\boldsymbol{x} \in \mathbb{F}_2^n : F(\boldsymbol{x}) = A(\boldsymbol{x})\}$ and $C_{F,A} = |\mathbf{C}_{F,A}| = 2^n - N_F$. Then inequality (10) implies the inequality

$$C_{F,A} > \sqrt{2^{n+1} - 7 \cdot 2^{-2}} + 2^{-1}.$$

[2] In the original, a linear manifold is called an affine subspace.

The number of all possible unordered pairs of elements from the set $\mathbf{C}_{F,A}$ satisfies the chain of relations

$$\binom{C_{F,A}}{2} = \frac{C_{F,A}(C_{F,A}-1)}{2} > 2^n - 1.$$

Therefore, among nonzero vectors from the set $\{\boldsymbol{x}_1 \oplus \boldsymbol{x}_2 : \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbf{C}_{F,A},\ \boldsymbol{x}_1 \neq \boldsymbol{x}_2\}$ there will definitely be the same. The vectors $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$ and $\boldsymbol{x}_1 \oplus \boldsymbol{x}_3$, where $\boldsymbol{x}_2 \neq \boldsymbol{x}_3$, obviously differ. Therefore, there are pairwise distinct vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4 \in \mathbf{C}_{F,A}$, for which the equality $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2 = \boldsymbol{x}_3 \oplus \boldsymbol{x}_4$ is satisfied. These vectors form a 2-dimensional linear manifold on which F coincides with A. In accordance with Proposition 1, the vectorial function F is not APN. ■

It is easy to see that the lower bound on the nonlinearity of APN functions, obtained in Theorem 1, for $n > 4$ refines the estimate (8) from [15].

Corollary 1. Under the conditions of Theorem 1, for odd $n \geqslant 3$, the following inequality is true:

$$N_F \geqslant 2^n - 2^{(n+1)/2}. \tag{11}$$

Indeed, in the case of odd $n \geqslant 3$, for the difference of an integer $2^{(n+1)/2}$ and the root from expression (9), the chain of relations is valid

$$\sqrt{2^{n+1}} - \sqrt{2^{n+1} - 7 \cdot 2^{-2}} = \frac{7}{2^2(\sqrt{2^{n+1}} + \sqrt{2^{n+1} - 7 \cdot 2^{-2}})} < \frac{7}{30}.$$

Corollary 2. Under the conditions of Theorem 1, the following inequality is true:

$$\operatorname{ard} F \geqslant \frac{2^n}{\sqrt{2^{n+1} - 7 \cdot 2^{-2}} + 2^{-1}}; \tag{12}$$

and in the case of odd n

$$\operatorname{ard} F \geqslant 2^{(n-1)/2}. \tag{13}$$

Estimates (12) and (13) refine the lower bound on the affinity order from [20] for APN functions.

Inequality (6) can be used as the upper nonlinearity bound for APN functions.

3. Nonlinearity of APN functions up to dimension 5

Results on the nonlinearity of APN functions in small dimensions are given for classes of extended affine (EA) equivalence, since unordered sets of algebraic degrees of nonlinearity and absolute values of the Walsh — Hadamard coefficients for nonzero linear combinations of coordinate functions, cardinalities of subsets of variables satisfying condition (2), and also, Hamming distances to all affine mappings are invariants [9] for EA-equivalent vectorial Boolean functions (under CCZ-equivalence, only the spectrum of absolute Walsh — Hadamard values remains as an invariant). Accordingly, all the above nonlinearity indicators, including the algebraic degree, are also invariants in the case of EA-equivalence.

The results obtained in this section are based on results [21, 22], where all classes of EA-equivalent APN functions up to dimension 5 are presented through the canonical element, which is the representative of the class with the smallest truth table in the lexicographic sense. To shorten the notation, the 2^n-ary number system will be used.

For $n = 1$, all Boolean functions are APN and simultaneously affine.

For $n = 2$, there is a single class of EA-equivalent APN functions presented in the Table 1. Along with the nonlinearity value N_F found here, the values of the degree of nonlinearity $\deg F$ and the nonlinearity in the sense (1) NL_F are also given.

Table 1

Nonlinearity of APN functions in dimension 2

$\boldsymbol{x}$	0	1	2	3	$\deg F$	NL_F	N_F
$F(\boldsymbol{x})$	0	0	0	1	2	0	1

The value of the nonlinearity coincides with the lower bound (9) and the upper bound (6). The affinity order of all APN functions in dimension 2 is 2. There are no permutations in this EA-class and there is the APN function represented by the power function x^3 over a field $\mathbb{F}_4$. Vectorial functions in dimension n over the field $\mathbb{F}_2$, represented by one-dimensional power functions of the form x^d over the field $\mathbb{F}_{2^n}$, are commonly called power vectorial functions or simply power functions[3] with exponent d.

For $n = 3$, the class of EA-equivalent APN functions is also unique and is presented together with the nonlinearity indicators in the Table 2.

Table 2

Nonlinearity of APN functions in dimension 3

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	$\deg F$	NL_F	N_F
$F(\boldsymbol{x})$	0	0	0	1	0	2	4	7	2	2	4

In this case, according to (5), all APN functions are AB. The value of the nonlinearity coincides with the lower bound (11) and the upper bound (6). In accordance with (4), the affinity order of all such mappings is greater than or equal to 2. This class contains permutations, including power functions with exponents 3, 5, and 6.

For $n = 4$, there are 2 classes of EA-equivalent APN functions (these classes are CCZ-equivalent), presented in the Table 3.

Table 3

Nonlinearity of APN functions in dimension 4

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\deg F$	NL_F	N_F
$F_1(\boldsymbol{x})$	0	0	0	1	0	2	4	7	0	4	6	3	8	14	10	13	2	4	10
$F_2(\boldsymbol{x})$	0	0	0	1	0	2	4	7	0	4	6	3	8	14	11	12	3	4	10

The values of nonlinearity for both EA-classes coincides with the lower bound (9) and the upper bound (6). In accordance with (4), the affinity order of all APN functions in dimension 4 is greater than or equal to 3. There are no permutations in these classes. The first class contains power functions with exponents 3, 6, 9, and 12. In the second class, there are no power functions, but there are APN functions found in [23].

For $n = 5$, there are already 7 classes of EA-equivalent APN functions, presented in the Table 4 (the first, third and seventh classes, as well as the second, fourth and sixth classes are CCZ-equivalent).

[3]The term monomial functions is also used.

Table 4

Nonlinearity of APN functions in dimension 5

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$F_1(\boldsymbol{x})$	0	0	0	1	0	2	4	7	0	4	8	13	16	22	28	27	0	8	16	25
$F_2(\boldsymbol{x})$	0	0	0	1	0	2	4	7	0	4	8	13	16	22	28	27	0	8	16	25
$F_3(\boldsymbol{x})$	0	0	0	1	0	2	4	7	0	4	8	13	16	22	29	26	0	8	16	25
$F_4(\boldsymbol{x})$	0	0	0	1	0	2	4	7	0	4	8	13	16	22	29	26	0	8	16	25
$F_5(\boldsymbol{x})$	0	0	0	1	0	2	4	8	0	3	6	12	7	16	25	23	0	7	3	22
$F_6(\boldsymbol{x})$	0	0	0	1	0	2	4	8	0	3	6	16	8	21	26	29	0	5	12	27
$F_7(\boldsymbol{x})$	0	0	0	1	0	2	4	8	0	3	6	16	8	21	26	29	0	6	15	24

$\boldsymbol{x}$ *cont.*	20	21	22	23	24	25	26	27	28	29	30	31	$\deg F$	NL_F	N_F
$F_1(\boldsymbol{x})$ *cont.*	5	15	17	26	22	26	14	3	3	13	31	16	2	12	25
$F_2(\boldsymbol{x})$ *cont.*	5	15	17	26	27	23	3	14	14	0	18	29	2	12	25
$F_3(\boldsymbol{x})$ *cont.*	5	15	19	24	7	11	27	22	26	20	1	14	3	12	25
$F_4(\boldsymbol{x})$ *cont.*	5	15	19	24	10	6	22	27	23	25	12	3	3	12	25
$F_5(\boldsymbol{x})$ *cont.*	28	19	9	0	19	8	15	28	21	9	29	2	4	10	25
$F_6(\boldsymbol{x})$ *cont.*	20	6	31	16	7	31	8	22	9	26	17	11	3	12	25
$F_7(\boldsymbol{x})$ *cont.*	18	3	17	30	2	29	14	20	25	13	9	23	3	12	25

The calculated values of nonlinearity for all 7 EA-classes are the same. The resulting value exceeds the lower bound (11) by 1 and coincides with the the upper bound (6). In accordance with (4), the affinity order of all APN functions in dimension 5 is greater than or equal to 5. All APN functions from the first, second, sixth and seventh classes are AB, and from the fifth class are not AB. These 5 classes contain permutations. The third and fourth classes don't contain any permutations, but contain the AB functions found in [23].

In this case, all power functions with exponents from 1 to 30 are permutations. In order to determine whether they are APN permutations and obtain the distribution of power APN permutations over the indicated 5 classes of EA-equivalence, which is absent in [21], let's recall known results. H. Dobbertin [24] conjectured that the six known infinite families of power APN functions presented in Table 5 exhaust the entire set of power APN functions (in accordance with later works, the Niho case for $n \equiv 3 \pmod 4$ was added to the original table from [24]).

Table 5

Known infinite families of power APN functions

Name	Exponent	Conditions
Gold	2^k+1	$(k,n)=1,\ 1 \leqslant k < n/2$
Kasami	$2^{2k}-2^k+1$	$(k,n)=1,\ 2 \leqslant k < n/2$
Welch	$2^{(n-1)/2}+3$	n odd
Niho	$2^{(n-1)/2}+2^{(n-1)/4}-1$ $2^{(n-1)/2}+2^{(3n-1)/4}-1$	$n \equiv 1 \pmod 4$ $n \equiv 3 \pmod 4$
Dobbertin	$2^{4n/5}+2^{3n/5}+2^{2n/5}+2^{n/5}-1$	$n \equiv 0 \pmod 5$
Inverse	2^n-2	n odd

The power functions from the Welch and Niho families, and also in the case of odd n from the Gold and Kasami families, are AB functions. At the same time, the mappings from the Dobbertin and Inverse families are not AB. All power functions from the Gold family are quadratic.

The equivalence of exponents was also discussed in [24], which is defined as follows: if a power function x^d is an APN, then a power function x^h is also an APN, where for $0 \leqslant i < n$ modulo comparison $h \equiv 2^i d \pmod{(2^n-1)}$ is true, and also in the case when

x^d is a permutation, one more comparison $hd \equiv 2^i \pmod{(2^n - 1)}$ is true. In this sense, each exponent presented above gives in fact an equivalence class of exponents for which the power function is the APN. Unfortunately, this equivalence is sometimes forgotten to be mentioned by some authors, which narrows the reader's understanding about possible exponents of power APN functions.

Dobbertin's conjecture has not yet been proven, but it has been checked for all values of n up to 34. It was shown in [25–27] that the equivalence of exponents corresponds to the CCZ-equivalence of APN functions. It follows from [21] that the first and seventh classes, as well as the second and sixth classes are CCZ-equivalent. In addition, the first class contains x^5, the second class contains x^3, the fifth class contains x^{15}, the sixth class contains x^{11}, and the seventh class contains x^7. Then, after calculating the equivalent exponents for the CCZ-equivalent power APN functions and knowing their algebraic degrees, we obtain the following proposition.

Proposition 2. All non-affine power 5-dimensional permutations are APN and the following distribution of power APN permutations over 5 classes of EA-equivalence of APN functions in dimension 5 takes place:

— exponents 5 (Gold, Niho), 9, 10, 18, 20 correspond to the first class;
— exponents 3 (Gold), 6, 12, 17, 24 correspond to the second class;
— exponents 15, 23, 27, 29 (Dobbertin), 30 (Inverse) correspond to the fifth class;
— exponents 11, 13 (Kasami), 21, 22, 26 correspond to the sixth class;
— exponents 7 (Welch), 14, 19, 25, 28 correspond to the seventh class.

For $n \geqslant 6$, the situation with finding the nonlinearity of APN functions becomes much more complicated. Firstly, a complete partition of such functions into EA-equivalence classes is currently unknown, while the number of already known EA-classes even for $n = 6$ is measured in hundreds (the most advanced results in this direction are presented in [28, 29]). Second, the complexity of computing a nonlinearity for a mapping from $P_2^{n,n}$ is $O(2^{n^2+2n})$ additive operations in the field $\mathbb{F}_{2^n}$, and thus computing such a nonlinearity for n greater than or equal to 6 is itself a difficult task.

Consider a special case of the APN 6-dimensional permutation S presented in [30] (Table 6).

Table 6

APN 6-dimensional permutation

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(\boldsymbol{x})$	0	54	48	13	15	18	53	35	25	63	45	52	3	20	41	33
$\boldsymbol{x}$ *cont.*1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$S(\boldsymbol{x})$ *cont.*1	59	36	2	34	10	8	57	37	60	19	42	14	50	26	58	24
$\boldsymbol{x}$ *cont.*2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$S(\boldsymbol{x})$ *cont.*2	39	27	21	17	16	29	1	62	47	40	51	56	7	43	44	38
$\boldsymbol{x}$ *cont.*3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$S(\boldsymbol{x})$ *cont.*3	31	11	4	28	61	46	5	49	9	6	23	32	30	12	55	22

Permutation S, like the majority of known APN functions in dimensional 6, has a relatively high nonlinearity in the sense (1) equal to 24. Computer calculation of its nonlinearity gives a value 55, which exceeds the lower bound (9) by 2 and is inferior to the upper bound (6) also by 1. In accordance with (4), the affinity order of permutation S is greater than or equal to 8.

The obtained values of nonlinearity for APN functions in small dimensions allow us to assume that, in contrast to the nonlinearity in the sense (1), all APN functions have the same nonlinearity.

Since there are no APN 4-dimensional permutations, let's consider further the behavior of nonlinearity for differentially 4-uniform permutations in dimension 4.

4. Nonlinearity of differentially 4-uniform permutations in dimension 4

From the results [22], it follows that there are 13 EA-equivalent classes of differentially 4-uniform mappings from $P_2^{4,4}$ containing 4-dimensional permutations (the second and third, fourth and twelfth, fifth and sixth EA-classes in addition are pairwise CCZ-equivalent). As in the case of APN functions, we represent these classes in the Table 7 through their canonical elements in the hexadecimal notation, with three indicators for each of them, including the nonlinearity values found here.

Table 7

Nonlinearity of differentially 4-uniform permutations in dimension 4

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\deg F$	NL_F	N_F
$F_1(\boldsymbol{x})$	0	0	0	0	0	1	2	3	0	2	4	8	0	12	5	7	3	2	9
$F_2(\boldsymbol{x})$	0	0	0	0	0	1	2	3	0	4	8	13	0	5	14	10	3	4	9
$F_3(\boldsymbol{x})$	0	0	0	0	0	1	2	3	0	4	8	13	0	6	11	12	3	4	9
$F_4(\boldsymbol{x})$	0	0	0	0	0	1	2	3	0	4	8	13	0	6	12	11	3	4	9
$F_5(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	3	6	2	8	6	15	3	2	9
$F_6(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	3	6	3	8	7	15	3	2	9
$F_7(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	3	8	2	7	13	5	3	2	9
$F_8(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	3	8	4	11	12	14	3	4	9
$F_9(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	3	8	4	13	10	14	3	4	9
$F_{10}(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	3	8	4	13	14	10	3	4	9
$F_{11}(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	6	8	2	9	13	14	3	4	9
$F_{12}(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	1	6	8	2	13	8	15	3	4	9
$F_{13}(\boldsymbol{x})$	0	0	0	0	0	1	2	4	0	2	8	15	1	10	15	6	3	4	9

Using the results [22], it can be shown based on the number of matches of canonical elements with zero function that all EA-classes containing 4-dimensional permutations with differential uniformity greater than or equal to 6 give nonlinearity less than or equal to 9. Since, as can be seen from the Table 7, the permutations of all 13 classes have the same nonlinearity equal to 9, we can say that differentially 4-uniform permutations have the maximum possible nonlinearity in the class S_2^4, which exceeds the lower bound (7) by 1 and is inferior to the upper bound (6) by 2. In accordance with (4), the affinity order of all differentially 4-uniform 4-dimensional permutations, as for APN functions in this dimension, is greater than or equal to 3.

At the same time, the nonlinearity in the sense (1) for permutations of the first, fifth, sixth, and seventh classes is inferior to that for permutations of the remaining nine classes, equal to 4. The latter, as is known, is the maximum possible nonlinearity in the sense (1) for 4-dimensional permutations. Thus, we obtain the following proposition.

Proposition 3. There are 9 pairwise not EA-equivalent (7 pairwise not CCZ-equivalent) classes of APN functions in dimension 4 containing permutations with three optimal nonlinearity indicators, namely: $\triangle_S = 4$, $NL_S = 4$ and $N_S = 9$.

In [31], all 4-dimensional permutations with two optimal nonlinearity indicators ($\triangle_S$=4, $NL_S = 4$) were divided into 16 affine equivalence classes. We represent this partition in

terms of canonical representatives within the extended affine equivalence classes in the Table 8. The left column shows the number of the EA-class from the Table 7.

Table 8

Classes of 4-dimensional permutations with optimal nonlinear indicators

No. EA-class	$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	S_1	0	1	2	13	4	7	15	6	8	11	12	9	3	14	10	5
	S_2	0	1	2	13	4	7	15	6	8	14	9	5	10	11	3	12
3	S_3	0	1	2	13	4	7	15	6	8	11	14	3	5	9	10	12
	S_4	0	1	2	13	4	7	15	6	8	11	14	3	10	12	5	9
4	S_5	0	1	2	13	4	7	15	6	8	14	12	11	9	3	10	5
8	S_6	0	1	2	13	4	7	15	6	8	14	12	9	5	11	10	3
9	S_7	0	1	2	13	4	7	15	6	8	12	9	11	10	14	5	3
	S_8	0	1	2	13	4	7	15	6	8	12	11	9	10	14	5	3
10	S_9	0	1	2	13	4	7	15	6	8	12	14	11	10	9	3	5
	S_{10}	0	1	2	13	4	7	15	6	8	14	11	10	5	9	12	3
	S_{11}	0	1	2	13	4	7	15	6	8	14	11	10	9	3	12	5
11	S_{12}	0	1	2	13	4	7	15	6	8	14	11	3	5	9	10	12
	S_{13}	0	1	2	13	4	7	15	6	8	14	11	5	10	9	3	12
12	S_{14}	0	1	2	13	4	7	15	6	8	14	12	11	3	9	5	10
13	S_{15}	0	1	2	13	4	7	15	6	8	12	5	3	10	14	11	9
	S_{16}	0	1	2	13	4	7	15	6	8	12	11	9	10	14	3	5

Note also that permutations from the second and third EA-classes have 3 quadratic nonzero linear combinations of coordinate functions, permutations from the fourth, eleventh and twelfth EA-classes have 1 such quadratic combination, and for permutations from the eighth, ninth, tenth and thirteenth EA-classes, all nonzero linear combinations of coordinate functions are cubic. In addition, all power 4-dimensional permutations, namely x^7, x^{11}, x^{13} and x^{14}, are in the same thirteenth EA-class.

5. Conclusion

The nonlinearity of a vectorial function shows the minimum number of mismatches between its images and the images of an arbitrary affine mapping. Here we study the behavior of this nonlinearity for the class of mappings of the space $\mathbb{F}_2^n$ into itself, which have an optimal nonlinearity of a different form, namely, APN functions. For comparison and completeness, the behavior of the nonlinearity defined as the maximum nonlinearity of all nonzero combinations of coordinate functions is also given.

Among the most significant results is the lower bound on the nonlinearity of APN functions, obtained in Theorem 1 and Corollary 1. The lower bound obtained here, together with the upper bound from [15], leave a rather narrow range for possible nonlinearity values of APN functions, which is presented in Table 9 for $n \leqslant 8$.

Table 9

Bounds on nonlinearity of APN functions

n	1	2	3	4	5	6	7	8
Lover bound (9) or (11)	0	1	4	10	24	53	112	233
Exact value (Section 3)	0	1	4	10	25	(55) ?	?	?
Upper bound (6)	0	1	4	10	25	56	119	246

In addition, the lower nonlinearity bound makes it possible to obtain a lower bound on the affinity order of such mappings (Corollary 2), which guarantees that in an arbitrary

piecewise affine representation of any APN function F there is at least the obtained number of affinity domains. This number directly affects the complexity of solving the system of nonlinear equations given by F [20].

The results obtained for the nonlinearity of APN functions in small dimension allow us to formulate some problems and make conjectures about its behavior in the general case. As has been shown, all APN functions of fixed dimension up to 5 have the same nonlinearity value (in contrast to the nonlinearity defined as the minimal nonlinearity of nonzero combinations of coordinate functions). In this regard, the following question arises.

Problem 1. Do all APN functions in fixed dimension really have the same nonlinearity value?

It was also shown here that all APN functions in dimension up to 5 have the maximum possible nonlinearity among all mappings in the corresponding dimension. Therefore, if the answer to the first question is yes, then the second question arises.

Problem 2. Is the value of the nonlinearity of APN functions the maximum possible among all mappings in the corresponding dimension?

In [7] it was conjectured that the nonlinearity of all vectorial Boolean functions from $P_2^{n,k}$ is less than or equal to $(1-2^{-k})(2^n-2^{n/2})$, and, accordingly, for $k=n$, the conjectured upper bound has the form

$$N_F \leqslant 2^n - 2^{n/2} - 1 + 2^{-n/2}. \tag{14}$$

From the results obtained above, it is easy to see that the studied 6-dimensional permutation also has a nonlinearity value coinciding with (14). In a sense, this confirms the conjecture that all APN functions have the same nonlinearity, which is the maximum possible among all mappings in corresponding dimension.

In the paper, the distribution of power APN permutations over 5 classes of EA-equivalence of APN functions in dimension 5 is obtained (Proposition 2).

All possible 9 classes of EA-equivalent differentially 4-uniform vectorial functions in dimension 4, containing permutations and having optimal two other nonlinearity indicators are also presented (Proposition 3). Using Table 7 and Table 8, it is much easier to find combinations of not EA-equivalent 4-dimensional permutations with all three optimal nonlinearity indicators.

REFERENCES

1. *Glukhov M. M.* O priblizhenii diskretnykh funktsiy lineynymi funktsiyami [On the approximation of discrete functions by linear functions]. Matematicheskiye Voprosy Kriptografii, 2016, vol. 7, no. 4, pp. 29–50. (in Russian)
2. *Nyberg K.* On the construction of highly nonlinear permutations. LNCS, 1993, vol. 658, pp. 92–98.
3. *Nyberg K.* Differentially uniform mappings for cryptography. LNCS, 1994, vol. 765, pp. 55–64.
4. *Nyberg K. and Knudsen L. R.* Provable security against a differential attack. LNCS, 1993, vol. 740, pp. 566–574.
5. *Chen L. and Fu F.* On the nonlinearity of multi-output Boolean functions. Acta Scientiarum Naturalium Universitatis Nankaiensis, 2001, vol. 34, no. 4, pp. 28–33. (in Chinese)
6. *Liu J. and Chen L.* On nonlinearity of the second type of multi-output Boolean functions. Chinese J. Eng. Math., 2014, vol. 31, no. 1, pp. 9–22. (in Chinese)
7. *Liu J., Mesnager S., and Chen L.* On the nonlinearity of S-boxes and linear codes. Cryptography and Communications, 2017, vol. 9, no. 1, pp. 345–361.
8. *Nagy G. P.* Thin Sidon sets and the nonlinearity of vectorial Boolean functions. `https://arxiv.org/pdf/2212.05887.pdf`, 2022.

9. *Ryabov V. G.* O priblizhenii vektornykh funktsiy nad konechnymi polyami i ikh ogranicheniy na lineynyye mnogoobraziya affinnymi analogami [On approximation of vectorial functions over finite fields and their restrictions to linear manifolds by affine analogues]. Diskretnaya Matematika, 2022, vol. 34, no. 2, pp. 83–105. (in Russian)
10. *Ryabov V. G.* K voprosu o priblizhenii vektornykh funktsiy nad konechnymi polyami affinnymi analogami [On the question of approximation of vectorial functions over finite fields by affine analogues]. Matematicheskiye Voprosy Kriptografii, 2022, vol. 13, no. 4, pp. 125–146. (in Russian)
11. *Carlet C. and Ding C.* Nonlinearities of S-boxes. Finite Fields Appl., 2007, vol. 13, no. 1, pp. 121–135.
12. *Carlet C.* Relating three nonlinearity parameters of vectorial functions and building APN functions from bent functions. Designs Codes Cryptography, 2011, vol. 59, no. 1–3, pp. 89–109.
13. *Carlet C.* Open questions on nonlinearity and on APN functions. LNCS, 2015, vol. 9061, pp. 83–107.
14. *Carlet C.* On the properties of the Boolean functions associated to the differential spectrum of general APN functions and their consequences. IEEE Trans. Inform. Theory, 2021, vol. 67, no. 10, pp. 6926–6939.
15. *Carlet C.* Bounds on the nonlinearity of differentially uniform functions by means of their image set size, and on their distance to affine functions. IEEE Trans. Inform. Theory, 2021, vol. 67, no. 12, pp. 8325–8334.
16. *Carlet C., Heuser A., and Picek S.* Trade-offs for S-Boxes: cryptographic properties and side-channel resilience. LNSC, 2017, vol. 10355, pp. 393–414.
17. *Gorodilova A. A., Tokareva N. N., Agievich S. V., et al.* An overview of the Eight International Olympiad in Cryptography "Non-Stop University CRYPTO". Sibirskiye Elektronnyye Matematicheskiye Izvestiya, 2022, vol. 19, no. 1, pp. A.9–A.37.
18. *Hou X.* Affinity of permutations of F_2^n. Discrete Appl. Math., 2006, vol. 154, no. 2, pp. 313–325.
19. *Carlet C.* Vectorial Boolean functions for cryptography. In Y. Crama & P. Hammer (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering (Encyclopedia of Mathematics and its Applications), Cambridge, Cambridge University Press, 2010, pp. 398–470.
20. *Gorshkov S. P. and Dvinyaninov A. V.* Nizhnyaya i verkhnyaya otsenki poryadka affinnosti preobrazovaniy prostranstv bulevykh vektorov [Lower and upper bounds on the affinity order of transformations of spaces of Boolean vectors]. Prikladnaya Diskretnaya Matematika, 2013, no. 2(20), pp. 14–18. (in Russian)
21. *Brinkmann M. and Leander G.* On the classification of APN functions up to dimension five. Designs Codes Cryptography, 2008, vol. 49, no. 1–3, pp. 273–288.
22. *Brinkmann M.* Extended Affine and CCZ Equivalence up to Dimension 4. `https://eprint.iacr.org/2019/316.pdf`, 2019.
23. *Budaghyan L., Carlet C., and Pott A.* New classes of almost bent and almost perfect nonlinear polynomials. IEEE Trans. Inform. Theory, 2006, vol. 52, no. 3, pp. 1141–1152.
24. *Dobbertin H.* Almost perfect nonlinear power functions on GF(2^n): the Niho case. Information and Computation, 1999, vol. 151, no. 1–2, pp. 57–72.
25. *Yoshiara S.* Equivalences of power APN functions with power or quadratic APN functions. J. Algebraic Combinatorics, 2016, vol. 44, no. 3, pp. 561–585.
26. *Dempwolff U.* CCZ equivalence of power functions. Designs Codes Cryptography, 2018, vol. 86, no. 3, pp. 665–692.

27. *Dempwolff U.* Correction to: CCZ equivalence of power functions. Designs Codes Cryptography, 2022, vol. 90, no. 2, pp. 473–475.
28. *Calderini M.* On the EA-classes of known APN functions in small dimensions. `https://eprint.iacr.org/2019/369.pdf`, 2019.
29. *Calderini M.* On the EA-classes of known APN functions in small dimensions. Cryptography and Communications, 2020, vol. 12, no. 5, pp. 821–840.
30. *Browning K. A., Dillon J. F., McQuistan M. T., and Wolfe A. J.* An APN permutation in dimension six. Finite Fields: Theory and Appl., 2010, pp. 33–42.
31. *Leander G. and Poschmann A.* On the classification of 4 bit S-boxes. LNCS, 2007, vol. 4547, pp. 156–176.

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 003.26 + 004.056 + 001.99 DOI 10.17223/20710410/61/3

ЗАЩИЩЁННОЕ ФОРМИРОВАНИЕ ПУБЛИЧНЫХ ПАРАМЕТРОВ И УСТРАНЕНИЕ УЯЗВИМОСТЕЙ КРАТКИХ НЕИНТЕРАКТИВНЫХ АРГУМЕНТОВ С НУЛЕВЫМ РАЗГЛАШЕНИЕМ

И. В. Мартыненков

АО «КВАНТ-ТЕЛЕКОМ», г. Москва, Россия

E-mail: mivpost@yandex.ru

Рассматриваются известные методы устранения уязвимостей кратких неинтерактивных аргументов с нулевым разглашением на основе корректировки уравнений верификации доказательств, значений публичных параметров в виде главных ссылочных строк и ключей формирования доказательств. Описаны способы защищённого формирования главных ссылочных строк с использованием доверенной третьей стороны и многостороннего взаимодействия.

Ключевые слова: *краткие неинтерактивные аргументы, публичные параметры, главные ссылочные строки, защищённость.*

SECURE FORMATION OF PUBLIC PARAMETERS AND ELIMINATION OF VULNERABILITIES OF ZERO-KNOWLEDGE SUCCINT NON-INTERACTIVE ARGUMENTS OF KNOWLEDGE

I. V. Martynenkov

JSC “KVANT-TELECOM”, Moscow, Russia

The methods of eliminating vulnerabilities of zero-knowledge succinct non-interactive arguments of knowlede are considered. The methods are based on the security of public parameters formation in the form of common reference strings using a trusted third party and multilateral interaction. The multilateral formation of the common reference strings uses the only honest party with a fixed and unlimited set of participants, as well as verification of the reliability of the results. Examples of increasing the level of security of zero-knowledge succinct non-interactive arguments of knowledge based on the correction of proof verification equations and the values of the common reference strings, eliminating redundant elements from the common reference strings and the keys of proof formation are given. The protocols that develop the construction of the common reference strings from static to updatable and universal versions are mentioned.

Keywords: *succinct non-interactive arguments, public parameters, common reference strings, security.*

Введение

Описаны способы устранения уязвимостей протоколов zk-SNARK [1], основанных на защищённом формировании публичных параметров в виде главных ссылочных строк (Common Reference String, CRS) с использованием доверенной третьей стороны и многостороннего взаимодействия. Для протоколов zk-SNARK [2, 3], используемых в криптовалюте Zcash [4], рассматривается многостороннее формирование CRS [5] с единственной честной стороной. Для протокола zk-SNARK [6] на примере [7] рассмотрено более защищённое формирование CRS с неограниченным набором сторон в расширяемом онлайн-режиме, верификацией результатов и единственной честной стороной. Повышение уровня защищённости протокола zk-SNARK [6] также представлено переработанной версией [8], в которой выполнена корректировка уравнений верификации и значений, включаемых в CRS. Согласно источникам [9, 10], указывается ошибка описания протокола zk-SNARK [3], основанного на протоколе zk-SNARK «Pinocchio» [2]. Уязвимость [9, 10] опирается на избыточные элементы, включаемые в CRS и ключ формирования доказательств, которые позволяют строить доказательства для произвольных открытых входов. Приводится метод [9] устранения указанной уязвимости.

В качестве упоминания приводится набор протоколов [11–13], развивающий формирование CRS [7] от статической до обновляемой и универсальной версий, в том числе для протокола zk-SNARK [6] и криптовалюты Zcash [4]. Согласно источникам [14, 15], кратко отмечена возможность повышения уровня защищённости протокола zk-SNARK [2] для случая сговора всех сторон формирования CRS при условии предварительной проверки доказательств.

1. Протокол надёжного формирования CRS с фиксированным набором сторон

В [14] представлен метод формирования CRS для случая сговора всех задействованных сторон, при котором нарушители способны реконструировать «лазейку» τ и подделывать доказательства π. В работе [5] устраняются проблемы защищённости [14] и приведён многосторонний протокол формирования CRS для протоколов zk-SNARK [2, 3], используемых в криптовалюте Zcash [4]. Если хотя бы одна сторона является честной, то конструирование мошеннических доказательств становится невозможным. Дополнительно обеспечивается свойство нулевого разглашения (Zero-Knowledge, ZK), даже если все стороны являются нарушителями.

В протоколе формирования CRS [5] принимают участие n сторон, координатор и верификатор. Роль последних может выполняться одним сервером. Проверка корректности результатов протокола происходит после его полного выполнения, что может оказаться недостатком и потребует перевыполнения всего тяжеловесного процесса формирования CRS. Протокол состоит из четырёх раундов, где каждая сторона P_i, $i = 1, \ldots, n$, может отправить своё сообщение после получения сообщения от P_{i-1}, а P_1 начинает работу после приёма сообщения инициализации. В пределах каждого раунда стороны могут отправлять сообщения для координатора параллельно. Таким образом, протокол формирования CRS [5] имеет вид:

Раунд № 1. Для $i = 1, \ldots, n$ стороны P_i выполняют следующие шаги:

1. Выводятся наборы случайных элементов из $\mathbb{F}_r^*$ (для наглядности здесь и далее i-индексы вида $\rho_{B,i}$ не приводятся, а подразумеваются по умолчанию):

$$\begin{gathered}\mathsf{secrets}_i = \{\tau, \rho_A, \rho_B, \alpha_A, \alpha_B, \alpha_C, \beta, \gamma\},\\ \mathsf{elements}_i = \{\tau, \rho_A, \rho_B, \alpha_A, \alpha_B, \alpha_C, \beta, \gamma, \rho_A\alpha_A, \rho_B\alpha_B, \rho_A\rho_B, \rho_A\rho_B\alpha_C, \beta\gamma\}_i.\end{gathered} \tag{1}$$

2. На основе (1) вычисляется набор элементов, для чего P_i выбирает случайные $f_1, f_2, f_3 \in \mathbb{G}_2 \setminus \{0\}$, $f_4, f_5, f_6, f_7, f_8 \in \mathbb{G}_1 \setminus \{0\}$, а затем сохраняет наборы элементов $\mathbb{G}_2$ и $\mathbb{G}_1$ соответственно:

$$\begin{gathered}\mathsf{e}_i^{(2)} = \{f_1, f_1\rho_A, f_1\rho_A\alpha_A, f_1\rho_A\rho_B\alpha_C, f_1\rho_A\rho_B, f_1\rho_A\rho_B\alpha_B, f_2, f_2\beta, f_2\beta\gamma, f_3, f_3\tau\},\\ \mathsf{e}_i^{(1)} = \{f_4, f_4\alpha_A, f_5, f_5\alpha_C, f_6, f_6\rho_B, f_7, f_7\rho_A, f_8, f_8\gamma\}.\end{gathered}$$

3. Вычисляется результирующий вектор $\mathsf{e}_i = \left(\mathsf{e}_i^{(1)} \parallel \mathsf{e}_i^{(2)}\right)$.
4. Стороны P_i широковещательно передают $h_i = \mathsf{COMMIT}(\mathsf{e}_i)$, соответствующее вычислению хеш-функции (в работе [5] используется BLAKE-2).

Относительно e_i требуется, чтобы для каждого $s \in \mathsf{secrets}_i$ (1) при $s \in \mathbb{F}_r^*$ всегда присутствовала необходимая s-пара (p, q), где $s \cdot p = q$. Для конкретной группы используется термин $\mathbb{G}$-s-пара. Например, по $f_1, f_1\rho_A, f_1\rho_A\rho_B$ строятся ρ_A-пара $(f_1, f_1\rho_A)$, ρ_B-пара $(f_1\rho_A, f_1\rho_A\rho_B)$ и $\rho_A\rho_B$-пара $(f_1, f_1\rho_A\rho_B)$.

Раунд № 2.

Ч а с т ь 1 . Выполняется раскрытие обязательств, при котором стороны P_i широковещательно передают e_i, а верификатор проверяет соответствие $h_i = \mathsf{COMMIT}(\mathsf{e}_i)$. Далее $\mathsf{elements}_i$ (1) принимают наименование зафиксированных элементов, так как к текущему моменту выполнения протокола для каждого $s \in \mathsf{elements}_i$ стороны отправили s-пары для $\mathbb{G}_1$ и $\mathbb{G}_2$ в виде rp_s^1 и rp_s^2. Для представленных ниже значений переданные сторонами P_i элементы и s-пары имеют следующий вид:

$$\begin{aligned}
\tau &: (\mathsf{rp}_\tau^1, \mathsf{rp}_\tau^2) = (g, \tau g), & \gamma &: (\mathsf{rp}_\gamma^1, \mathsf{rp}_\gamma^2) = (g, \gamma g),\\
\rho_A &: (\mathsf{rp}_{\rho_A}^1, \mathsf{rp}_{\rho_A}^2) = (g, \rho_A g), & \rho_A\alpha_A &: (\mathsf{rp}_{\rho_A\alpha_A}^1, \mathsf{rp}_{\rho_A\alpha_A}^2) = (g, \rho_A\alpha_A g),\\
\rho_B &: (\mathsf{rp}_{\rho_B}^1, \mathsf{rp}_{\rho_B}^2) = (g, \rho_B g), & \rho_B\alpha_B &: (\mathsf{rp}_{\rho_B\alpha_B}^1, \mathsf{rp}_{\rho_B\alpha_B}^2) = (\rho_A g, \rho_A\rho_B\alpha_B g),\\
\alpha_A &: (\mathsf{rp}_{\alpha_A}^1, \mathsf{rp}_{\alpha_A}^2) = (\rho_A g, \rho_A\alpha_A g), & \rho_A\rho_B &: (\mathsf{rp}_{\rho_A\rho_B}^1, \mathsf{rp}_{\rho_A\rho_B}^2) = (g, \rho_A\rho_B g),\\
\alpha_B &: (\mathsf{rp}_{\alpha_B}^1, \mathsf{rp}_{\alpha_B}^2) = (\rho_A\rho_B g, \rho_A\rho_B\alpha_B g), & \rho_A\rho_B\alpha_C &: (\mathsf{rp}_{\rho_A\rho_B\alpha_C}^1, \mathsf{rp}_{\rho_A\rho_B\alpha_C}^2) = (g, \rho_A\rho_B\alpha_C g),\\
\alpha_C &: (\mathsf{rp}_{\alpha_C}^1, \mathsf{rp}_{\alpha_C}^2) = (\rho_A\rho_B g, \rho_A\rho_B\alpha_C g), & \beta\gamma &: (\mathsf{rp}_{\beta\gamma}^1, \mathsf{rp}_{\beta\gamma}^2) = (g, \beta\gamma g).\\
\beta &: (\mathsf{rp}_\beta^1, \mathsf{rp}_\beta^2) = (\gamma g, \beta\gamma g),
\end{aligned} \tag{2}$$

Ч а с т ь 2 . Проверяется факт фиксации сторонами P_i пар элементов из $\mathbb{G}_1$ и $\mathbb{G}_2$, входящих в rp_s^1 и rp_s^2, соответствующих s-парам одинаковых элементов $s \in \mathbb{F}_r^*$. Для $i \in \{1, \ldots, n\}$, $s \in \mathsf{elements}_i$ верификатор запускает проверку

$$\mathsf{SameRatio}(\mathsf{rp}_s^1, \mathsf{rp}_s^2) = \mathsf{SameRatio}((p, q), (f, H)). \tag{3}$$

Проверка (3) успешна, если при $p, q \in \mathbb{G}_1$ и $f, H \in \mathbb{G}_2$ все четыре элемента не равны 0 и $e(p, H) = e(q, f)$ для билинейного спаривания $e(ag_1, bg_2) = g_T^{ab}$.

Ч а с т ь 3 . Выполняется доказательство и проверка знаний дискретных логарифмов. Сторона P_1 вычисляет дайджест сообщений раунда № 1 $h = \mathsf{COMMIT}(h_1 \parallel \ldots \parallel \parallel h_n)$ и широковещательно его передаёт. Затем для $i = 1, \ldots, n$ выполняются следующие шаги:

1. Для $s \in \mathsf{secrets}_i$ (1) фиксируется $h_{i,s} = (h \parallel \mathsf{rp}_s^1)$. Все стороны P_i и верификатор имеют элементы для вычисления текущего шага.
2. Для каждого $s \in \mathsf{secrets}_i$ сторонами P_i широковещательно передаются доказательства $\pi_{i,s}$:
$$\pi_{i,s} = \mathsf{NIZK}(\mathsf{rp}_s^1, h_{i,s}). \tag{4}$$
Функция NIZK (4) основана на протоколе Шнорра [16, 17], которая по s-паре $\mathsf{rp}_s = (f, H = sf)$ и строке h выводит рандомизированную строку — доказательство знания строки s. Для функции (4) выбирается случайное a и $R = af$. Вычисляется $c = \mathsf{COMMIT}(R \parallel h)$, которое интерпретируется как элемент $\mathbb{F}_r$, например взятием первых $\log r$ бит, а также $u = a + cs$. Результатом является пара (R, u).
3. Для каждого $s \in \mathsf{secrets}_i$ верификатором проводится проверка
$$1/0 = \mathsf{VERIFY} - \mathsf{NIZK}(\mathsf{rp}_s^1, \pi_{i,s}, h_{i,s}). \tag{5}$$
Функция (5) проверяет, действительно ли доказательство соответствует заданному h. Для этого вычисляется $c = \mathsf{COMMIT}(R \parallel h)$ и доказательство принимается, если $uf = R + cH$.

Ч а с т ь 4 . Для $\tau = \tau_1, \ldots, \tau_n$ (2) вычисляется вектор
$$\mathsf{POWERS}_r = ((1, \tau, \tau^2, \ldots, \tau^d)g_1, (1, \tau, \tau^2, \ldots, \tau^d)g_2).$$

1. Стороной P_1 выполняется широковещательная передача $V_1 = (1, \tau, \tau^2, \ldots, \tau^d)g_1$ и $V_1' = (1, \tau, \tau^2, \ldots, \tau^d)g_2$, а для $i = 2, \ldots, n$ — широковещательная передача сторонами P_i значений $V_i = \mathsf{powerMult}(V_{i-1}, \tau_i)$ и $V_i' = \mathsf{powerMult}(V_{i-1}', \tau_{i-1})$. Например, для вектора $V \in \mathbb{G}^{d+1}$ и $a \in \mathbb{F}_r$ функция $\mathsf{powerMult}(V, a)_{i \in \{0,\ldots,d\}} = \{a^i V : i = 0, \ldots, d\} \in \mathbb{G}^{d+1}$.
2. Для $i \in \{2, \ldots, n\}$ верификатор подтверждает корректность ранее переданных векторов. Выполняются проверки
$$\begin{gathered}\mathsf{SameRatioSeq}(V_1, \mathsf{rp}_{\tau_1}^{(2)}), \quad \mathsf{SameRatioSeq}(V_1', (V_{1,0}, V_{1,1})), \\ \mathsf{SameRatioSeq}(V_i, (V_{i,0}', V_{i,1}')), \quad \mathsf{SameRatioSeq}(V_i', (V_{i,0}, V_{i,1})), \\ \mathsf{SameRatio}((V_{i-1,1}, V_{i,1}), \mathsf{rp}_{\tau_i}^{(2)}).\end{gathered} \tag{6}$$
Если проверки (6) выполняются, то на выход подаются компоненты ключа доказательства ($PK_H = V_n, PK_H' = V_n'$). Функция подтверждения соответствия двух пар элементов s-парам SameRatio() описана в (3). Функция SameRatioSeq() для $V \in \mathbb{G}_1^d$ и $\mathsf{rp}_s \in \mathbb{G}_2^2$ проверяет, что каждые два последовательных элемента V являются s-парой. Она соответствует вызову $\mathsf{SameRatio}(V', \mathsf{rp}_s)$ для $V' = ((V_0, V_1), (V_1, V_2), \ldots, (V_{d-1}, V_d))$.
3. Выполняется проверка корней $Z(x)$. Для обеспечения нулевого разглашения требуется, чтобы τ являлось корнем $Z(x) = x^d - 1$. Для этого верификатор и все P_i проверяют равенство $Z(\tau)g_1 = (\tau^d - 1)g_1 = V_{n,d} - V_{n,0} \neq 1$, иначе протокол перезапускается.
4. Используется набор векторов квадратичной арифметической программы (Quadratic Arithmetic Program, QAP). Координатор вычисляет QAP-векторы доказательства $\mathbf{A}, \mathbf{B}, \mathbf{B_2}, \mathbf{C}$ в точке τ, где, в отличие от [3], $A_{m+1} = B_{m+1} = C_{m+1} = Z[\tau]g_1 = (\tau^d - 1)g_1$:
$$\begin{gathered}\mathbf{A} = g_1\{A_i(\tau) : i = 0, \ldots, m+1\}, \quad \mathbf{B} = g_1\{B_i(\tau) : i = 0, \ldots, m+1\}, \\ \mathbf{B}_2 = g_2\{B_i(\tau) : i = 0, \ldots, m+1\}, \quad \mathbf{C} = g_1\{C_i(\tau) : i = 0, \ldots, m+1\}.\end{gathered} \tag{7}$$

Раунд № 3.

Часть 1. Координатор широковещательно передает векторы $\mathbf{A}, \mathbf{B}, \mathbf{B_2}, \mathbf{C}$ (7).

Часть 2. Для получения различных элементов ключей выполняется подпротокол выработки случайных коэффициентов RCPC. Для элементов $\alpha = (\alpha_1, \ldots, \alpha_n) \in$ $\in \mathsf{elements}_i$ (1) вычисляются элементы ключей

$$\begin{gathered} PK_A = \mathsf{RCPC}(\mathbf{A}, \rho_A), \quad PK_B = \mathsf{RCPC}(\mathbf{B}_2, \rho_B), \quad PK_C = \mathsf{RCPC}(\mathbf{C}, \rho_A\rho_B), \\ PK'_A = \mathsf{RCPC}(\mathbf{A}, \rho_A\alpha_A), \quad PK'_B = \mathsf{RCPC}(\mathbf{B}, \rho_B\alpha_B), \quad PK'_C = \mathsf{RCPC}(\mathbf{C}, \rho_A\rho_B\alpha_C), \\ temp_B = \mathsf{RCPC}(\mathbf{B}, \rho_B), \quad VK_Z = \mathsf{RCPC}(g_2 Z(\tau) = g_2(\tau^d - 1), \rho_A\rho_B), \\ VK_A = \mathsf{RCPC}(g_2, \alpha_A), \quad VK_B = \mathsf{RCPC}(g_1, \alpha_B), \quad VK_C = \mathsf{RCPC}(g_2, \alpha_C). \end{gathered} \tag{8}$$

Для расчётов (8) не требуется s-пара для каждого $s \in \mathsf{secrets}_i$ каждой группы. Например, требуется только $\mathbb{G}_2$-ρ_A-пара, которая используется для вычисления PK_A, а $\mathbb{G}_1$-ρ_A-пара не используется.

Подпротокол RCPC (8) покоординатно умножает вектор элементов из $\mathbb{G}_1$ на скаляр $\alpha \in \mathbb{F}_r^*$. Перед вызовом подпротокола для каждого $i \in \{1, \ldots, n\}$ стороны P_i передали $\mathbb{G}_2$-α_i-пары rp_{α_i}, доступные верификатору. Таким образом, $\mathsf{RCPC}(V, \alpha)$ принимает $V \in \mathbb{G}_1^d$ и $\alpha_i \in \mathbb{F}_r^*$ для $i \in \{1, \ldots, n\}$. Сторона P_1 вычисляет $V_1 = \alpha_1 V$, а для $i = 2, \ldots, n$ стороны P_i широковещательно передают $V_i = \alpha_i V_{i-1}$. В итоге стороны выводят $V_n = \alpha V$.

Раунд № 4. Вычисляются компоненты ключей, содержащие элемент β, для чего стороны P_i или только координатор рассчитывают $V = PK_A + temp_B + PK_C$. Затем стороны вычисляют

$$\begin{gathered} PK_K = \mathsf{RCPC}(V, \beta), \quad VK_\gamma = \mathsf{RCPC}(g_2, \gamma), \\ VK^{(1)}_{\beta\gamma} = \mathsf{RCPC}(g_1, \beta\gamma), \quad VK^{(2)}_{\beta\gamma} = \mathsf{RCPC}(g_2, \beta\gamma). \end{gathered}$$

В заключение верификатор запускает функцию verifyRCPC() на входных параметрах в виде результатов работы раундов № 3, 4. Верификация успешна, если все вызовы подтверждены.

Для векторов $S, T \in \mathbb{G}_1^d$ и $\mathbb{G}_2$-α-пары rp_α функция $\mathsf{sameRatio}((S, T), \mathsf{rp}_\alpha)$ (3) возвращает $\mathsf{sameRatio}(V, \mathsf{rp}_\alpha)$, где $V_i = (S_i, T_i)$. Функция $\mathsf{verifyRCPC}(V, \alpha)$ принимает V, а также $V_1, \ldots, V_n \in \mathbb{G}_1^d$ и $\mathbb{G}_2$-α_i-пары rp_{α_i} для $i \in \{1, \ldots, n\}$. Затем запускается $\mathsf{sameRatio}((V, V_1), \mathsf{rp}_{\alpha_1})$, а для $i = 2, \ldots, n$ запускается $\mathsf{sameRatio}((V_{i-1}, V_i), \mathsf{rp}_{\alpha_i})$. Если все итерации успешны, верификация подтверждается.

В работе [15] отмечается, что злонамеренный выбор секретных параметров, зависящих от $\tau, \rho_A, \rho_B, \alpha_A, \alpha_C, \gamma, \beta \in \mathbb{F}_r^*$, может нарушить свойство нулевого разглашения протокола [2]. Даже при отсутствии честных сторон P_i формирование CRS способом [5] устраняет данную уязвимость протокола [2] при условии проверки доказательства перед его отправкой.

2. Протокол надёжного формирования CRS с неограниченным и расширяемым набором сторон

По сравнению с работой [5], в [7] рассмотрено более защищённое формирование CRS протокола zk-SNARK [6]. Алгоритмы формирования доказательства и верификации соответствуют протоколу zk-SNARK [6]. Протокол [7] работает с неограниченным набором сторон в расширяемом онлайн-режиме без их предварительного выбора, не требует хранения конфиденциальных значений в течение длительного времени и предварительной фиксации случайных значений. Доказательства будут достоверны, если

хотя бы одна из N сторон является честной. Дополнительную роль играет ненадёжный координатор — детерминированная функция, причём любая сторона способна проверить корректность его сообщений. В частности, верификатор формирования CRS [7] независимо вычисляет сообщения координатора и проверяет их достоверность. В роли верификатора может выступать как координатор, так и любая сторона протокола.

В [7] арифметическая схема $\mathbf{C}$ над $\mathbb{F}_p$ рассматривается как чередующиеся слои умножения/деления $\mathbf{C}_1, \ldots, \mathbf{C}_d$ и их линейные комбинации $\mathbf{L}_1, \ldots, \mathbf{L}_d$. Вход схемы $\mathbf{x}$ разделяется на непересекающиеся наборы $\mathbf{x}^1, \ldots, \mathbf{x}^d$, соответствующие слоям. Целью вычислений является вывод $\mathbf{C}(\mathbf{x})\mathbf{g}$ для произвольного входа $\mathbf{x}$. Детальнее, рассматривается произведение $\mathbf{x} = (\mathbf{x}_1 \cdot \ldots \cdot \mathbf{x}_N \cdot \mathbf{x}')$, где $\mathbf{x}_i \in (\mathbb{F}_p^*)^t$ — вход стороны P_i, а $\mathbf{x}'$ — выход функции рандомизации RB. Уровни схемы обозначаются как $\mathbf{C}_1, \mathbf{L}_1, \ldots, \mathbf{C}_d, \mathbf{L}_d$, и протокол выполняет d рабочих фаз. Для выполнения одной фазы фиксируется уровень $l \in \{1, \ldots, d\}$, $\mathbf{C} = \mathbf{C}_l$ и $\mathbf{L} = \mathbf{L}_l$. Предполагается, что для всех вентилей $\mathbf{g}$ предыдущих уровней $\mathbf{C}_1, \mathbf{L}_1, \ldots, \mathbf{C}_{l-1}, \mathbf{L}_{l-1}$ уже вычислен набор значений $[\mathbf{g}] \in \mathbf{G} = \mathbb{G}_1 \times \mathbb{G}_2$.

По сравнению с протоколом zk-SNARK [6] в CRS [7] добавлены элементы

$$\{x^i : i = n, \ldots, 2n-2\}, \quad \{\alpha x^i : i = 1, \ldots, n-1\}, \quad \{\beta x^i : i = 1, \ldots, n-1\} \tag{9}$$

и убраны элементы $\{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\gamma : i = 0, \ldots, l\}$, γ, которые являются линейной комбинацией компонентов CRS и добавленных элементов (9). В [7] протокол zk-SNARK [6] вычисляется с помощью схемы с двумя слоями. При этом слой $\mathbf{C}_1$ имеет вход $\mathbf{x}^1 = \{x, \alpha, \beta\}$ и вычисляет (9). Слой $\mathbf{L}_1$ вычисляет $\{x^i t(x) : i = 0, \ldots, n-2\}$ как линейные комбинации x^i, $i \in \{0, \ldots, 2n-2\}$, где $\deg(t(x)) = n$, а также $\{(\beta u_i(x) + \alpha v_i(x) + w_i(x)) : i = 0, \ldots, m\}$ как линейные комбинации результатов $\mathbf{C}_1$. Слой $\mathbf{C}_2$ имеет вход $\mathbf{x}^2 = \{\delta\}$ и вычисляет δ, $\{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta : i = l+1, \ldots, m\}$, $\{x^i t(x)/\delta : i = 0, \ldots, n-2\}$.

Для упрощения надёжное формирование CRS [7] для протокола zk-SNARK [6] представлено для одной группы и аналогично для других групп. Используются следующие значения: n — степень полиномов QAP [6]; $[x]$ — набор значений всех открытых входов; $\{1, \ldots, l\}$ — индексы открытых входов; $\{l+1, \ldots, m\}$ — индексы секретных входов. Значение M является вычисляемым выходом в $\mathbb{G}_1$, $\mathbb{G}_2$ или $\mathbf{G} = \mathbb{G}_1 \times \mathbb{G}_2$; P_j — стороны протокола при $j \in \{1, \ldots, N\}$; $[M]^j$ — значение M после внесения сторонами $P_1, \ldots, P_j$ долей распределённых вычислений; $[M]^0$ — начальное значение; $\mathbf{g} = (g_1, g_2)$ — порождающие элементы групп $\mathbb{G}_1, \mathbb{G}_2$. Выполняются два раунда протокола [7].

Раунд № 1. Используются следующие вспомогательные функции. Формирование доказательства знания $\alpha \in \mathbb{F}_p^*$ выполняется функцией $\mathsf{POK}(\alpha, \mathbf{v})$, которая строит $r = R([\alpha]_1, \mathbf{v}) \in \mathbb{G}_2^*$ и на выход подает $([\alpha]_1, \alpha r)$, где R — хеш-функция; $[\alpha]_1 = \alpha g_1$ при $\langle g_1 \rangle = \mathbb{G}_1$. Верификация доказательства знания $\alpha \in \mathbb{F}_p^*$ выполняется за счёт функции $\mathsf{CheckPOK}(a, \mathbf{v}, b)$ при $a \in \mathbb{G}_1^*$, $b \in \mathbb{G}_2^*$, которая строит $r = R(a, \mathbf{v}) \in \mathbb{G}_2^*$ и на выход подаёт $\mathsf{SameRatio}((g_1, a), (r, b))$. Функция $\mathsf{SameRatio}$ (3) соответствует [5].

Функция $\mathsf{Consistent}(A, B, C)$ [7] использует $(A, B) \in \mathbb{G}_1^2$ или $(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2 = \mathbf{G}^2$, а также $C \in \mathbb{G}_2^*$ или $C \in (\mathbb{G}_2^*)^2$. Если $C \in (\mathbb{G}_2^*)^2$, то вычисляется $r = \mathsf{SameRatio}((A_1, B_1), (C_1, C_2))$, иначе $r = \mathsf{SameRatio}((A_1, B_1), (g_2, C))$. Если $(A, B) \in \mathbb{G}_1^2$, то на выход подаётся r, иначе выводится r и $\mathsf{SameRatio}((A_1, B_1), (A_2, B_2))$. Верификатор протокола для $j \in \{1, \ldots, N\}$ вычисляет $r_{\alpha,j} = R([\alpha_j]_1, \mathsf{transcript}_{1,j-1})$, $r_{\beta,j} = R([\beta_j]_1, \mathsf{transcript}_{1,j-1})$, $r_{x,j} = R([x_j]_1, \mathsf{transcript}_{1,j-1})$. Функция $\mathsf{transcript}_{1,j-1}$ выводит сообщения протокола до момента отправки стороной P_j нового сообщения.

Функция RB принимает временной слот J, целое число k и возвращает случайное $a \in \mathbb{F}_p^*$. Пусть $(J-1)$ — временной интервал, в который P_N отправляет сообщение.

Выполняются следующие шаги:

1. Для каждого входа x схемы $\mathbf{C}$ выполняется инициализация значений

$$\{[x^i]^0 = \mathbf{g} : i = 1, \dots, n-1\}, \quad \{[x^i]^0 = g_1 : i = n, \dots, 2n-2\},$$
$$\{[\alpha x^i]^0 = g_1 : i = 0, \dots, n-1\}, \quad [\beta]^0 = \mathbf{g}, \quad \{[\beta x^i]^0 = g_1 : i = 1, \dots, n-1\}.$$

2. Для $j \in \{1, \dots, N\}$ стороны протокола P_j вычисляют набор значений

$$[\alpha_j]_1, \quad [\beta_j]_1, \quad [x_j]_1,$$
$$y_{\alpha,j} = \mathsf{POK}(\alpha_j, \mathsf{transcript}_{1,j-1}), \quad y_{\beta,j} = \mathsf{POK}(\beta_j, \mathsf{transcript}_{1,j-1}),$$
$$y_{x,j} = \mathsf{POK}(x_j, \mathsf{transcript}_{1,j-1}),$$
$$\{[x^i]^j = x_j^i [x^i]^{j-1} : i = 1, \dots, 2n-2\}, \quad \{[\alpha x^i]^j = \alpha_j x_j^i [\alpha x^i]^{j-1} : i = 0, \dots, n-1\},$$
$$\{[\beta x^i]^j = \beta_j x_j^i [\beta x^i]^{j-1} : i = 0, \dots, n-1\}.$$

3. Координатор выводит $(x', \alpha', \beta') = \mathsf{RB}(J, 3)$, определяются значения

$$\{[x^i] = x'^i [x^i]^N : i = 1, \dots, 2n-2\}, \quad \{[\alpha x^i] = \alpha' x'^i [\alpha x^i]^N : i = 0, \dots, n-1\},$$
$$\{[\beta x^i] = \beta' x'^i [\beta x^i]^N : i = 0, \dots, n-1\}.$$

4. Достоверность полученных значений выполняется проверками

$$\mathsf{CheckPOK}([\alpha_j]_1, \mathsf{transcript}_{1,j-1}, y_{\alpha,j}), \quad \mathsf{CheckPOK}([\beta_j]_1, \mathsf{transcript}_{1,j-1}, y_{\beta,j}),$$
$$\mathsf{CheckPOK}([x_j]_1, \mathsf{transcript}_{1,j-1}, y_{x,j}),$$
$$\mathsf{consistent}([\alpha]^{j-1} - [\alpha]^j, (r_{\alpha,j}, y_{\alpha,j})), \quad \mathsf{consistent}([\beta]^{j-1} - [\beta]^j, (r_{\beta,j}, y_{\beta,j})),$$
$$\mathsf{consistent}([x]^{j-1} - [x]^j, (r_{x,j}, y_{x,j})),$$
$$\{\mathsf{consistent}([x^{i-1}]^j - [x^i]^j, [x]^j) : i = 1, \dots, 2n-2\},$$
$$\{\mathsf{consistent}([x^i]_1^j - [\alpha x^i]^j, [\alpha]^j) : i = 1, \dots, n-1\},$$
$$\{\mathsf{consistent}([x^i]_1^j - [\beta x^i]^j, [\beta]^j) : i = 1, \dots, n-1\}.$$

5. На выход раунда № 1 подаётся набор значений

$$M_1 = \big\{\{[x^i] : i = 0, \dots, n-1\}, \ \{[x^i]_1 : i = n, \dots, 2n-2\},$$
$$\{[\alpha x^i]_1 : i = 0, \dots, n-1\}, \ [\beta], \ [\delta], \ \{[\beta x^i]_1 : i = 1, \dots, n-1\}\big\}. \tag{10}$$

Раунд № 2. Выполняются следующие шаги:

1. Для каждого входа x схемы $\mathbf{C}$ вычисляются

$$K_i = \{\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta : i = l+1, \dots, m\},$$
$$H_i = \{t(x)x^i/\delta : i = 0, \dots, n-2\}.$$

2. Выполняется инициализация необходимых для раунда № 2 значений:

$$\{[K_i]^0 = K_i' : i = l+1, \dots, m\}, \quad \{[H_i]^0 = H_i' : i = l+1, \dots, m\}, \quad [\delta]^0 = \mathbf{g}.$$

3. Для $j \in \{1, \ldots, N\}$ стороны протокола P_j вычисляют набор значений

$$[\delta_j]_1, \quad y_{\delta,j} = \mathsf{POK}(\delta_j, \mathsf{transcript}_{2,j-1}), \quad [\delta]^j = [\delta]^{j-1}/\delta_j,$$
$$\{[K_i]^j = ([K_i]^{j-1})/\delta_j : i = l+1, \ldots, m\}, \quad \{[H_i]^j = ([H_i]^{j-1})/\delta_j : i = 0, \ldots, n-2\}.$$

4. Аналогично раунду № 1, пусть $(J-1)$ — временной интервал, в который P_N отправляет сообщение. Выводится $\delta' = \mathsf{RB}(J, 1)$ и определяются значения

$$[\delta] = [\delta]^N/\delta', \quad [K_i]_1 = [K_i]^N/\delta', \quad [H_i]_1 = [H_i]^N/\delta'.$$

5. Верификатор для $j \in \{1, \ldots, N\}$ вычисляет $r_{\delta,j} = R([\delta_j]_1, \mathsf{transcript}_{2,j-1})$. Достоверность полученных значений выполняется проверками

$$\mathsf{CheckPOK}([\delta_j]_1, \mathsf{transcript}_{2,j-1}, y_{\delta,j}), \quad \mathsf{consistent}([\delta]^{j-1} - [\delta]^j, (r_{\delta,j}, y_{\delta,j})),$$
$$\{\mathsf{consistent}([K_i]^j - [K_i]^{j-1}, [\delta_j]) : i = l+1, \ldots, m\},$$
$$\{\mathsf{consistent}([H_i]^j - [H_i]^{j-1}, [\delta_j]) : i = 0, \ldots, n-2\}.$$

6. На выход раунда № 2 подаётся следующий набор значений:

$$M_2 = \big\{[\delta], \ \{[K_i]_1 : i = l+1, \ldots, m\}, \ \{[H_i]_1 : i = 0, \ldots, n-2\}\big\}. \tag{11}$$

Результирующая CRS [7] основана на M_1 (10) и M_2 (11) и принимает вид

$$\{[x^i] : i = 0, \ldots, n-1\}, \{[x^i]_1 : i = n, \ldots, 2n-2\}, \{[\alpha x^i]_1 : i = 0, \ldots, n-1\},$$
$$[\beta], \{[\beta x^i]_1 : i = 1, \ldots, n-1\}, \{[x^i t(x)/\delta]_1 : i = 0, \ldots, n-2\},$$
$$\{[(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta]_1 : i = l+1, \ldots, m\}.$$

По сравнению с работой [7], в [11] представлено альтернативное доказательство защищённости информации, содержащейся в CRS протокола zk-SNARK [6]. Для этого используется 2-раундовая процедура формирования и проверки корректности публичных значений. Протокол [11] формирует надёжную CRS и сохраняет конфиденциальность секретного входа для случая с единственной честной стороной. Защищённость сохраняется даже при доступе злоумышленников ко всем этапам рерандомизации CRS, кроме одного. В результате протокол [11] формирует новые обновляемые публичные параметры CRS для Zcash [4] и проводит их верификацию, развивая идеи [7] на основе протокола zk-SNARK [6]. Свойство обновляемости CRS позволяет динамичному набору сторон вносить в публичные параметры секретную случайность неограниченное количество раз.

На каждом этапе протокола [7] требуется наличие хотя бы одной честной стороны. При этом второй этап [7] зависит от первого, поэтому формируемая CRS не может обновляться. В протоколе [12] любая сторона в любое время способна обновить и подтвердить корректность обновлённой CRS. Это верно, если одна из старых CRS достоверна и/или одна из сторон обновления является честной. Таким образом, CRS [12] может непрерывно рерандомизироваться с сохранением результатов всех предыдущих обновлений, удовлетворяя доверию всех сторон, заинтересованных в надёжности формируемых публичных значений. Кроме того, CRS [12] не привязывается к конкретной дискретной функции, поэтому является универсальной CRS (Universal CRS, UCRS) и подходит для различных дискретных функций. В определённом смысле протокол формирования CRS [12] всё ещё использует доверенную третью сторону, однако уверенность в защищённости CRS повышается, потому что только одна предыдущая сторона должна уничтожать введённую секретную случайность.

Например, согласно [12], CRS протокола zk-SNARK [18] может быть обновлена, в то время как CRS протокола zk-SNARK «Pinocchio» [2] не является обновляемой и требует вмешательства доверенной третьей стороны. Источник [12] также демонстрирует способ нарушения защищённости исторически базового протокола zk-SNARK «Pinocchio» [2].

Работа [13] продолжает развитие идей [12], где используются обновляемые универсальные структурированные CRS (Structured UCRS, SUCRS) и приводится метод, в котором ненадёжные стороны повышают производительность пакетной верификации доказательств. Для произвольных дискретных функций доказательство протокола [13] составляет 256 байт, все компоненты которого принадлежат одной группе.

3. Устранение уязвимости протокола zk-SNARK Э. Бен-Сассона, А. Кьезы, Э. Тромера, М. Вирзы

Источник [9] раскрывает ошибку в описании протокола zk-SNARK [3]. По сравнению с исходной версией протокола zk-SNARK «Pinocchio» [2], схема [3] включает в CRS избыточные элементы, которые важно не раскрывать. Уязвимость позволяет при наличии корректного доказательства для некоторого открытого входа создавать корректные доказательства для любого открытого входа. В [9] представлено также доказательство надёжности протокола zk-SNARK [3] при исключении из CRS данных элементов и удовлетворении QAP определённым алгебраическим условиям.

Источник [10] описывает отличия протоколов zk-SNARK [2] и [3]. Показана уязвимость [3], в которой нарушитель представляет ложные открытый вход и доказательство, принимаемые верификатором. Согласно [10], устранение уязвимостей [3] требует издержек и компромиссов производительности. На основе результатов [10] в исправленной версии работы [3] в виде [19] атака была ослаблена. Однако в [9] представлена более серьёзная уязвимость протокола zk-SNARK [3] на основе избыточных элементов ключа формирования доказательства. Протоколы zk-SNARK [5, 20, 21] и реализация [22] основаны на [3], поэтому унаследовали проблемы защищённости. Реализация [23] не использует избыточные элементы, поэтому не подвержена атаке [9].

Для описания устранения уязвимости [9] используется нотация протокола zk-SNARK [3], где m — размер QAP; d — степень QAP; n — размер открытого входа $x \in \mathbb{F}^n$; QAP имеет форму $\big\{\{A_i(X), B_i(X), C_i(X) : i = 0, \ldots, m\}, Z(X)\big\}$; степени $A_i, B_i, C_i \in \mathbb{F}[X]$ не выше d; $Z \in \mathbb{F}[X]$ и имеет степень d; $[x]_i = g_i^x$ при $\langle g_i \rangle = \mathbb{G}_i$.

Таким образом, в [3] элементы $\mathsf{pk}'_{\mathsf{A,i}} = [\alpha_A \rho_A A_i(\tau)]_1$ не используются доказывающим и верификатором, однако позволяют доказывающему заменять открытые входы на основе корректного доказательства. Это возможно за счёт введения множителя для компоненты доказательства π_A, что изменяет значение открытого входа, связанного с исходным доказательством, с $x = (x_1, \ldots, x_n) \in \mathbb{F}^N$ на $x' = (x'_1, \ldots, x'_n) \in \mathbb{F}^N$. Первое уравнение верификации [3] $e(\pi'_A, g_2) = e(\pi_A, [\alpha_A]_2)$ для компонент доказательства $\pi_A = [\rho_A A_{mid}(\tau)]_1, \pi'_A = [\alpha_A \rho_A A_{mid}(\tau)]_1$, где $A_{mid} = \sum_{i=0}^{m} x_i A_i - \sum_{i=0}^{n} x_i A_i$, призвано недопустить нежелательное поведение, но избыточные элементы также позволяют злонамеренному доказывающему добавить аналогичный множитель к π'_A. В результате для изменённого открытого входа верификация становится успешной. Подробности атаки приведены в [9].

Уязвимость протокола zk-SNARK [3] устраняется исключением из ключа доказательства набора $\mathsf{pk}'_{\mathsf{A,i}} = [\alpha_A \rho_A A_i(\tau)]_1$ и использованием QAP с линейно независимыми полиномами $\{A_i : i = 0, \ldots, n\}$ [10], которые не пересекаются в индексах $i \in \{0, \ldots, n\}$

и $i \in \{n+1, \ldots, m\}$. В результате изменения протокола zk-SNARK [3] с учётом модификаций [9] принимают следующий вид:

Алгоритм формирования ключей

1. Выводятся случайные секретные значения $\tau, \rho_A, \rho_B, \alpha_A, \alpha_B, \alpha_C, \gamma, \beta \in \mathbb{F}_p^*$.
2. Для $i \in \{0, \ldots, d\}$ вычисляются τ^i.
3. Для $i \in \{0, \ldots, m\}$ вычисляются компоненты ключа доказательства:

$$\rho_A A_i(\tau), \quad \rho_B B_i(\tau), \quad \alpha_B \rho_B B_i(\tau), \quad \rho_A \rho_B C_i(\tau),$$
$$\alpha_C \rho_A \rho_B C_i(\tau), \quad \beta(\rho_A A_i(\tau) + \rho_B B_i(\tau) + \rho_A \rho_B C_i(\tau)).$$

4. Формируется набор $(\alpha_A, \alpha_B, \alpha_C, \gamma, \beta\gamma, \rho_A \rho_B Z(\tau))$ и для $i \in \{n+1, \ldots, m\}$ вычисляются $\alpha_A \rho_A A_i(\tau)$.

Алгоритм верификации

Вычисляется $\mathsf{PI}(x) = \rho_A A_0(\tau) + \sum\limits_{i=1}^{n} x_i \rho_A A_i(\tau)$ и проверяется выполнение следующих уравнений:

$$\pi'_A = \alpha_A \pi_A, \quad \pi'_B = \alpha_B \pi_B, \quad \pi'_C = \alpha_C \pi_C,$$
$$\gamma \pi_K = \beta\gamma(\mathsf{PI}(x) + \pi_A + \pi_B + \pi_C), \quad (\mathsf{PI}(x) + \pi_A)\pi_B = \pi_C + \pi_H Z(\tau) \rho_A \rho_B.$$

4. Устранение уязвимости протокола zk-SNARK Й. Грота

Протокол zk-SNARK [8] основан на программах квадратичной арифметики (Square Arithmetic Program, SAP) и билинейном спаривании. В [8] приняты меры по устранению проблем защищённости протокола zk-SNARK [6, 1], в котором уравнение верификации для известного полинома $f(\phi)$ и секретных α, β, δ имеет вид $e(A, B) = e(g^{\alpha}, h^{\beta}) e(g^{f(\phi)}, h) e(C, h^{\delta})$. В данном случае противник может изменить текущее доказательство в другое доказательство того же состояния ϕ. При этом он имеет два способа рандомизации компонент доказательства (A, B, C).

В первом случае противник для некоторого r переназначает компоненты доказательства в виде $A' = A^r$, $B' = B^{1/r}$, $C' = C$. Атака устраняется добавлением проверки $e(A, h) = e(g, B)$, а для значения $r = -1$ уравнение верификации корректируется для использования $e(Ag^{\alpha\delta}, Bh^{\beta\delta})$ вместо $e(A, B)$. Во втором случае противник выполняет переназначение вида $A' = A$, $B' = Bh^{r\delta}$, $C' = A^r C$. Для устранения атаки в CRS включаются h^{δ}, $g^{\gamma\delta}$ и $h^{\gamma\delta}$, а g^{δ} исключается. Но если противник устанавливает $B' = Bh^{r\delta}$, то единственным возможным значением, удовлетворяющим представленному ниже уравнению верификации, будет $A' = Ag^{r\delta}$ вместо доступного $A' = A$, которое противник вычислить не способен ввиду отсутствия g^{δ} в CRS. Значение $r = \Phi(\tau)$ соответствует значению полинома Φ в секретной «лазейке» τ. Результирующие уравнения верификации представлены ниже в алгоритме верификации доказательств.

В работе [8] используется частный случай QAP под названием SAP, где $u_i(x) = v_i(x)$ для всех i. Формальное определение SAP имеет следующий вид:

$$R = \big(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, l, \{u_i(x), w_i(x) : i = 0, \ldots, m\}, t(x)\big).$$

Билинейная группа определяет конечное поле $\mathbb{Z}_p$ для простого $p > 2^{\lambda-1}$, $1 \leqslant l \leqslant m$, $u_i(x), w_i(x), t(x) \in \mathbb{Z}_p[x]$ и $u_i(x), w_i(x)$ имеют строго более низкую степень, чем $n = \deg(t(x))$. Множество $S = \{u_i(x) : i = 0, \ldots, l\}$ линейно независимо, всякий полином

$u_i(x) \in S$ линейно независим от множества $\{u_j(x) : j = l+1, \ldots, m\}$. В результате при $s_0 = 1$ программа SAP определяет следующее бинарное отношение:

$$R = \Big\{(\phi, w) : \phi = (s_1, \ldots, s_l) \in \mathbb{Z}_p^l, w = (s_{l+1}, \ldots, s_m) \in \mathbb{Z}_p^{m-l}, \\ \exists h(x) \in \mathbb{Z}_p[x] \Big(\deg(h) \leqslant n-2 \;\&\; \big(\sum_{i=0}^{m} s_i u_i(x)\big)^2 = \sum_{i=0}^{m} s_i w_i(x) + h(x)t(x)\Big)\Big\}. \tag{12}$$

Алгоритм формирования ключей $(\mathsf{crs}_{\mathsf{sap}}, \rho) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{sap})$

На основе отношения R (12) и параметра защиты λ вырабатываются CRS и ρ — ключ проверки корректности CRS. Алгоритм Setup выполняет следующие шаги:

1. Выводятся случайные элементы для формирования ключа верификации $\tau = (\alpha, \beta, \gamma, \delta, x) \in \mathbb{Z}_p^5$, такие, что $t(x) \neq 0$.
2. Вычисляется CRS

$$\mathsf{crs}_{\mathsf{sap}} = \big(g^\alpha, g^\gamma, g^x, g^{\alpha\delta}, g^{\gamma\delta t(x)}, g^{\gamma^2\delta t(x)^2}, g^{(\alpha+\beta)\gamma\delta t(x)}, h, h^\beta, h^\delta, h^{\beta\delta}, h^{\gamma\delta t(x)}, h^{\delta^2}, \\ \{g^{\gamma\delta x^i}, h^{\gamma\delta x^i}, g^{\gamma^2\delta t(x) x^i} : i = 0, \ldots, n-1\}, \{g^{\gamma\delta w_i(x) + \delta(\alpha+\beta)u_i(x)} : i = 0, \ldots, l\}, \\ \{g^{\gamma^2\delta w_i(x) + \gamma\delta(\alpha+\beta)u_i(x)} : i = l+1, \ldots, m\}\big). \tag{13}$$

3. Вычисляется ключ ρ без раскрытия значения g^δ, предназначенный для проверки корректности CRS (13):

$$\rho = (g^\alpha, h^\beta, g^\gamma, h^\delta, g^x).$$

4. На выход подаётся тройка $(\mathsf{crs}, \tau, \rho)$.

Строка CRS (13) содержит $m + 2n + 5$ элементов из $\mathbb{G}_1$ и $n + 3$ элементов из $\mathbb{G}_2$. Критически важно, чтобы CRS не содержала значений вида g, g^b, в которых b является небольшой величиной, например битом, за счёт чего нарушитель может восстановить данную переменную и решить задачу дискретного логарифмирования.

Алгоритм проверки CRS-структуры $0/1 \leftarrow \mathsf{UpdateVerify}(1^\lambda, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, \rho)$

1. Выполняется проверка набора уравнений

$$\begin{aligned}
&e(g^{\alpha\delta}, h) = e(g^\alpha, h^\delta), && e(g^{\gamma\delta}, h^\beta) = e(g^\gamma, h^{\beta\delta}),\\
&e(g^{\gamma\delta}, h) = e(g^\gamma, h^\delta), && e(g^{\gamma\delta}, h^\delta) = e(g^\gamma, h^{\delta^2}),\\
&e(g^{\gamma\delta t(x)}, h^{\gamma\delta}) = && e(g^{\gamma\delta t(x)}, h^{\gamma\delta}) = e(g^{\gamma\delta}, h^{\gamma\delta t(x)}),\\
&\quad = e(g^{\gamma\delta(t(x)-t_0)\gamma\delta/x}, h^{\gamma\delta x}) e(g^{-t_0\gamma\delta}, h^{\gamma\delta}), &&\\
&e(g^{\gamma^2\delta t(x)^2}, h^\delta) = e(g^{\gamma\delta t(x)}, h^{\gamma\delta t(x)}), && e(g^{(\alpha+\beta)\gamma\delta t(x)}, h^\delta) =\\
&e(g^\gamma, h^\delta) = e(g, h^{\gamma\delta}), && \quad = e(g^{\alpha\delta}, h^{\gamma\delta t(x)}) e(g^{\gamma\delta t(x)}, h^{\beta\delta}).
\end{aligned} \tag{14}$$

2. Выполняется проверка набора уравнений для различных индексов i:

$$\begin{aligned}
0 \leqslant i \leqslant n-2 \;&:\; e(g^x, h^{\gamma x^i}) = e(g, h^{\gamma x^{i+1}}),\quad e(g^{\gamma\delta x^i}, h^{\gamma\delta x}) = e(g^{\gamma\delta x^{i+1}}, h^{\gamma\delta}),\\
1 \leqslant i \leqslant n-1 \;&:\; e(g^{\gamma^2\delta t(x) x^i}, h^{\gamma\delta}) = e(g^{\gamma\delta t(x)}, h^{\gamma\delta x^i}),\\
0 \leqslant i \leqslant l \;&:\; e(g^{\gamma\delta w_i(x) + \delta(\alpha+\beta)u_i(x)}, h^{\gamma\delta}) =\\
&\quad = e(g^{\gamma\delta}, h^{\gamma\delta w_i(x)}) e(g^{\alpha\delta}, h^{\gamma\delta u_i(x)}) e(g^{\gamma\delta u_i(x)}, h^{\beta\delta}),\\
l+1 \leqslant i \leqslant m \;&:\; e(g^{\gamma^2\delta w_i(x) + \gamma\delta(\alpha+\beta)u_i(x)}, h^\delta) =\\
&\quad = e(g^{\gamma\delta}, h^{\gamma\delta w_i(x)}) e(g^{\alpha\delta}, h^{\gamma\delta u_i(x)}) e(g^{\gamma\delta u_i(x)}, h^{\beta\delta}).
\end{aligned} \tag{15}$$

3. Если все проверки (14) и (15) выполняются, то на выход подаётся 1, иначе 0.

Алгоритм доказывающего $\pi \leftarrow \mathsf{Prove}(\mathsf{info}, \phi, w)$

Алгоритм доказывающего анализирует публичные ϕ и приватные w значения входов-выходов логических элементов схемы $(1, a_1, \ldots, a_m)$ и встраивает оценки SAP-полиномов $a_i u_i(x)$, полученные в случайной секретной точке x, в один элемент проверки в $\mathbb{G}_1$ и в один элемент проверки в $\mathbb{G}_2$. На их основе строится третий элемент доказательства, показателем которого является произведение показателей первых двух элементов; $\mathsf{info} = \left(\mathsf{bp}, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, e(g^{\alpha\delta}, h^{\beta\delta})\right)$. Алгоритм выполняет следующие шаги:

1. Выбирается случайный элемент $r \in \mathbb{Z}_p$ для обеспечения свойства ZK.
2. Выполняется выборка публичных и приватных значений входов-выходов логических элементов схемы $(\phi \parallel w)$: $(a_0, a_1, \ldots, a_m) \leftarrow \mathsf{Parse}(1, \phi, w)$.
3. Вычисляется многочлен-делитель

$$h(x) = \left(\left(\sum_{i=0}^{m} a_i u_i(x)\right)^2 - \sum_{i=0}^{m} a_i w_i(x)\right) / t(x).$$

4. Вычисляются компоненты доказательства $\pi = (A, B, C)$:

$$\begin{gathered} A = g^{\gamma\delta\left(rt(x) + \sum\limits_{i=0}^{m} a_i u_i(x)\right)}, \\ B = h^{\gamma\delta\left(rt(x) + \sum\limits_{i=0}^{m} a_i u_i(x)\right)}, \\ C = g^{\gamma\delta\left(\sum\limits_{i=l+1}^{m} a_i(\gamma w_i(x) + (\alpha+\beta)u_i(x)) + r(\alpha+\beta)t(x) + \gamma t(x)(r^2 t(x) + h(x) + 2r\sum\limits_{i=0}^{m} a_i u_i(x)\right)}. \end{gathered} \tag{16}$$

5. На выход подаётся доказательство $\pi = (A, B, C)$.

Размер доказательства (16) составляет два элемента из $\mathbb{G}_1$ и один элемент из $\mathbb{G}_2$.

Алгоритм верификатора $0/1 \leftarrow \mathsf{Verify}(\mathsf{info}, \phi, \pi)$

С использованием билинейного спаривания верификатор проверяет, что компоненты A и B доказательства π (16) имеют общие показатели степеней. Также проверяется, что показатель степени компонента C доказательства π является произведением показателей первых двух компонент. Алгоритм Verify выполняет следующие шаги:

1. Выполняется выборка открытых значений входов логических элементов схемы ϕ: $(a_0, a_1, \ldots, a_l) \leftarrow \mathsf{Parse}(1, \phi)$.
2. Выполняется разделение компонент доказательства (16):

$$(A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \leftarrow \mathsf{Parse}(\pi).$$

3. Выполняется проверка уравнений верификации

$$\begin{gathered} e(Ag^{\alpha\delta}, Bh^{\beta\delta}) = e(g^{\alpha\delta}, h^{\beta\delta}) e\left(g^{\sum\limits_{i=0}^{l} a_i\delta(\gamma w_i(x) + (\alpha+\beta)u_i(x))}, h^{\gamma\delta}\right) e(C, h^{\delta}), \\ e(A, h) = e(g, B). \end{gathered} \tag{17}$$

   Компонента A однозначно определяет B через второе уравнение (17), а пара (A, B) однозначно определяет C через первое уравнение (17).
4. Если все проверки (17) выполняются, то выводится 1, иначе 0.

Алгоритм моделирования доказательства $\pi \leftarrow \mathsf{Sim}(\mathsf{bp}, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, \tau, \phi)$

За счёт «лазейки» τ возможно вывести корректные доказательства. Для этого выполняются следующие шаги:

1. Выполняется выборка открытых значений входов логических элементов схемы ϕ: $(a_0, a_1, \ldots, a_l) \leftarrow \mathsf{Parse}(1, \phi)$.
2. Выбирается случайное значение $\mu \in \mathbb{Z}_p$.
3. Вычисляются компоненты доказательства $(A, B = g^{\mu\delta}, h^{\mu\delta})$.
4. Вычисляется компонента $C = g^{(\mu^2\delta + (\alpha+\beta)\mu\delta - \gamma\delta \sum\limits_{i=0}^{l} a_i(\gamma w_i(x) + (\alpha+\beta)u_i(x)))}$.
5. На выход подаётся доказательство $\pi = (A, B, C)$.

Заключение

Рассмотрены способы устранения уязвимостей протоколов zk-SNARK, основанных на нарушении защищённости формирования CRS [5, 7, 11–15]. Представленные протоколы построения надёжных CRS являются самостоятельными и тяжеловесными конструкциями, которыми возможно расширять протоколы zk-SNARK, например [2, 3, 6]. Проверка корректности формируемых CRS, например [5, 7], происходит после выполнения всего протокола, что в случае выявления частых нарушений может оказаться недостатком и требует повторного выполнения всего процесса построения CRS. Описаны более защищённые версии прототоколов zk-SNARK [6, 3] в виде модернизированных схем [8–10]. Отмечается [24], что рассмотренные в работе протоколы zk-SNARK [2, 3, 6] имеют фиксированный размер доказательств и постоянное количество уравнений верификации, что дополнительно обосновывает целесообразность их практического применения. Проблемы защищённости других производительных протоколов zk-SNARK [24], например [19, 20, 25–29], в публичном доступе не представлены, поэтому их также можно рассматривать в качестве кандидатов на практическое применение.

ЛИТЕРАТУРА

1. *Мартыненков И. В.* Краткие неинтерактивные аргументы с нулевым разглашением на основе наборов полиномов // Прикладная дискретная математика. 2023. № 59. С. 34–72.
2. *Parno B., Howell J., Gentry C., and Raykova M.* Pinocchio: Nearly practical verifiable computation // Proc. 34th IEEE Symp. Security and Privacy. Oakland, 2013. P. 238–252.
3. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct non-interactive Zero Knowledge for a von Neumann architecture // Proc. 23rd USENIX Security Symp. San Diego, CA, USA, 2014. P. 781–796.
4. *Hopwood D., Bowe S., Hornby T., and Wilcox N.* Zcash Protocol Specification. Version 2021.2.16 [NU5]. 2021. 213 p.
5. *Bowe S., Gabizon A., and Green M.D.* A Multi-Party Protocol for Constructing the Public Parameters of the Pinocchio zk-SNARK. Cryptology ePrint Archive. Paper 2017/602. 2017. 25 p. `https://ia.cr/2017/602.`
6. *Groth J.* On the size of pairing-based non-interactive arguments // LNCS. 2016. V. 9666. P. 305–326.
7. *Bowe S., Gabizon A., and Miers I.* Scalable Multi-Party Computation for zk-SNARK Parameters in the Random Beacon Model. Cryptology ePrint Archive. Paper 2017/1050. 2017. 24 p. `https://eprint.iacr.org/2017/1050.`
8. *Groth J. and Maller M.* Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. Cryptology ePrint Archive. 2017. 36 p. `https://eprint.iacr.org/2017/540.pdf.`
9. *Gabizon A.* On the Security of the BCTV Pinocchio zk-SNARK Variant. Cryptology ePrint Archive. Paper 2019/119. 2019. 9 p. `https://eprint.iacr.org/2019/119.`

10. *Parno B.* A Note on the Unsoundness of vnTinyRAM's SNARK. Cryptology ePrint Archive. Paper 2015/437. 2015. 4 p. https://eprint.iacr.org/2015/437.
11. *Maller M.* A Proof of Security for the Sapling Generation of zk-SNARK Parameters in the Generic Group Model. 2018. 12 p. https://github.com/zcash/saplingsecurity-analysis/blob/master/MaryMallerUpdated.pdf.
12. *Groth J., Kohlweiss M., Maller M., et al.* Updatable and Universal Common Reference Strings with Applications to zk-SNARKs. Cryptology ePrint Archive. Paper 2018/280. 2018. 38 p. https://eprint.iacr.org/2018/280.
13. *Maller M., Bowe S., Kohlweiss M., and Meiklejohn S.* Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings. Cryptology ePrint Archive. Paper 2019/099. 2019. 20 p. https://eprint.iacr.org/2019/099.
14. *Ben-Sasson E., Chiesa A., Green M., et al.* Secure sampling of public parameters for succinct zero knowledge proofs // IEEE Symp. SP 2015. San Jose, CA, USA, 2015. P. 287–304.
15. *Campanelli M., Gennaro R., Goldfeder S., and Nizzardo L.* Zero-knowledge contingent payments revisited: Attacks and payments for services // Proc. ACM SIGSAC Conf. CCS'17. N.Y.: ACM, 2017. P. 229–243.
16. *Schnorr C.* Efficient identification and signatures for smart cards // LNCS. 1990. V. 435. P. 239–252.
17. *Черёмушкин А. В.* Криптографические протоколы. Основные свойства и уязвимости. М.: Издательский центр «Академия», 2009. 272 с.
18. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments // LNCS. 2010. V. 6477. P. 321–340.
19. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. Updated version. 2019. 37 p. https://eprint.iacr.org/2013/879.pdf.
20. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data // Proc. 2015 IEEE Symp. Security and Privacy. San Jose, CA, USA, 2015. P. 271–286.
21. *Fuchsbauer G.* Subversion-Zero-Knowledge Snarks. Cryptology ePrint Archive. Paper 2017/587. 2017. 32 p. https://eprint.iacr.org/2017/587.
22. zkSNARKs implementation in JavaScript & WASM. https://github.com/iden3/snarkjs.
23. C++ library for zkSNARKs. https://github.com/scipr-lab/libsnark.
24. *Мартыненков И. В.* Способы повышения производительности кратких неинтерактивных аргументов с нулевым разглашением и анализ достигнутых результатов // Прикладная дискретная математика. 2023. № 60. С. 40–58.
25. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs // LNCS. 2013. V. 7881. P. 626–645.
26. *Ben-Sasson E., Chiesa A., Genkin D., et al.* SNARKs for C: Verifying program executions succinctly and in zero knowledge // LNCS. 2013. V. 8043. P. 90–108.
27. *Danezis G., Fournet C., Groth J., and Kohlweiss M.* Square span programs with applications to succinct NIZK arguments // LNCS. 2014. V. 8873. P. 532–550.
28. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Scalable zero knowledge via cycles of elliptic curves // LNCS. 2014. V. 8617. P. 276–294.
29. *Costello C., Fournet C., Howell J., et al.* Geppetto: Versatile verifiable computation // Proc. IEEE Symp. SP'15. IEEE Computer Society, USA, 2015. P. 253–270.

REFERENCES

1. *Martynenkov I. V.* Kratkie neinteraktivnye argumenty s nulevym razglasheniem na osnove naborov polinomov [Zero-knowledge succinct non-interactive arguments of knowledge based on sets of polynomials]. Prikladnaya Diskretnaya Matematika, 2023, no. 59, pp. 34–72. (in Russian)
2. *Parno B., Howell J., Gentry C., and Raykova M.* Pinocchio: Nearly practical verifiable computation. Proc. 34th IEEE Symp. Security and Privacy, Oakland, 2013, pp. 238–252.
3. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct non-interactive Zero Knowledge for a von Neumann architecture. Proc. 23rd USENIX Security Symp., San Diego, CA, USA, 2014, pp. 781–796.
4. *Hopwood D., Bowe S., Hornby T., and Wilcox N.* Zcash Protocol Specification. Version 2021.2.16 [NU5], 2021. 213 p.
5. *Bowe S., Gabizon A., and Green M.D.* A Multi-Party Protocol for Constructing the Public Parameters of the Pinocchio zk-SNARK. Cryptology ePrint Archive. Paper 2017/602, 2017. 25 p. `https://ia.cr/2017/602.`
6. *Groth J.* On the size of pairing-based non-interactive arguments. LNCS, 2016. vol. 9666, pp. 305–326.
7. *Bowe S., Gabizon A., and Miers I.* Scalable Multi-Party Computation for zk-SNARK Parameters in the Random Beacon Model. Cryptology ePrint Archive. Paper 2017/1050, 2017. 24 p. `https://eprint.iacr.org/2017/1050.`
8. *Groth J. and Maller M.* Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. Cryptology ePrint Archive, 2017. 36 p. `https://eprint.iacr.org/2017/540.pdf.`
9. *Gabizon A.* On the Security of the BCTV Pinocchio zk-SNARK Variant. Cryptology ePrint Archive. Paper 2019/119, 2019. 9 p. `https://eprint.iacr.org/2019/119.`
10. *Parno B.* A Note on the Unsoundness of vnTinyRAM's SNARK. Cryptology ePrint Archive. Paper 2015/437, 2015. 4 p. `https://eprint.iacr.org/2015/437.`
11. *Maller M.* A Proof of Security for the Sapling Generation of zk-SNARK Parameters in the Generic Group Model. 2018. 12 p. `https://github.com/zcash/saplingsecurity-analysis/blob/master/MaryMallerUpdated.pdf.`
12. *Groth J., Kohlweiss M., Maller M., et al.* Updatable and Universal Common Reference Strings with Applications to zk-SNARKs. Cryptology ePrint Archive. Paper 2018/280, 2018. 38 p. `https://eprint.iacr.org/2018/280.`
13. *Maller M., Bowe S., Kohlweiss M., and Meiklejohn S.* Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings. Cryptology ePrint Archive. Paper 2019/099, 2019. 20 p. `https://eprint.iacr.org/2019/099.`
14. *Ben-Sasson E., Chiesa A., Green M., et al.* Secure sampling of public parameters for succinct zero knowledge proofs. IEEE Symp. SP 2015, San Jose, CA, USA, 2015, pp. 287–304.
15. *Campanelli M., Gennaro R., Goldfeder S., and Nizzardo L.* Zero-knowledge contingent payments revisited: Attacks and payments for services. Proc. ACM SIGSAC Conf. CCS'17, N.Y., ACM, 2017, pp. 229–243.
16. *Schnorr C.* Efficient identification and signatures for smart cards. LNCS, 1990, vol. 435, pp. 239–252.
17. *Cheremushkin A. V.* Kriptograficheskie protokoly. Osnovnye svojstva i uyazvimosti [Cryptographic Protocols. Basic Properties and Vulnerabilities]. Moscow, Akademiya Publ., 2009. 272 p. (in Russian)
18. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments. LNCS, 2010, vol. 6477, pp. 321–340.

19. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. Updated version, 2019. 37 p. `https://eprint.iacr.org/2013/879.pdf`.
20. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data. Proc. 2015 IEEE Symp. Security and Privacy, San Jose, CA, USA, 2015, pp. 271–286.
21. *Fuchsbauer G.* Subversion-Zero-Knowledge Snarks. Cryptology ePrint Archive. Paper 2017/587, 2017. 32 p. `https://eprint.iacr.org/2017/587`.
22. zkSNARKs implementation in JavaScript & WASM. `https://github.com/iden3/snarkjs`.
23. C++ library for zkSNARKs. `https://github.com/scipr-lab/libsnark`.
24. *Martynenkov I. V.* Sposoby povysheniya proizvoditel'nosti kratkikh neinteraktivnykh argumentov s nulevym razglasheniem i analiz dostignutykh rezul'tatov [Ways to improve the performance of zero-knowledge succinct non-interactive arguments of knowledge and analysis of the results achieved]. Prikladnaya Diskretnaya Matematika, 2023, no. 60, pp. 40–58. (in Russian)
25. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs. LNCS, 2013, vol. 7881, pp. 626–645.
26. *Ben-Sasson E., Chiesa A., Genkin D., et al.* SNARKs for C: Verifying program executions succinctly and in zero knowledge. LNCS, 2013, vol. 8043, pp. 90–108.
27. *Danezis G., Fournet C., Groth J., and Kohlweiss M.* Square span programs with applications to succinct NIZK arguments. LNCS, 2014, vol. 8873, pp. 532–550.
28. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Scalable zero knowledge via cycles of elliptic curves. LNCS, 2014, vol. 8617, pp. 276–294.
29. *Costello C., Fournet C., Howell J., et al.* Geppetto: Versatile verifiable computation. Proc. IEEE Symp. SP'15, IEEE Computer Society, USA, 2015, pp. 253–270.

УДК 519.719.325 DOI 10.17223/20710410/61/4

СИСТЕМЫ С ОТКРЫТЫМИ КЛЮЧАМИ НА ОСНОВЕ ИДЕНТИФИКАЦИОННОЙ ИНФОРМАЦИИ

А. В. Черемушкин

Академия криптографии РФ, г. Москва, Россия

E-mail: avc238@mail.ru

Рассмотрены особенности практического применения криптографических систем с открытыми ключами на основе идентификаторов. Выделены математические задачи и конструкции, приведены основные подходы к построению систем шифрования, цифровой подписи, аутентификации сторон и ключевых систем с открытыми ключами на основе идентификаторов.

Ключевые слова: *криптография на основе идентификаторов, криптосистемы с открытыми ключами, эллиптическая кривая, билинейное спаривание.*

ID-BASED PUBLIC KEY CRYPTOGRAPHIC SYSTEMS

A. V. Cheremushkin

Academy of Cryptography of the Russian Federation, Moscow, Russia

A surway contains an analisys of practical aspects of ID-based public key cryptography. IDB-systems simplify a certificate management process, but trusted requirements for the key generation center (KGC) must be very stronger than for certification authority. When key escrow property is not assumed, users' private keys should be protected from malicious KGC. Many networks need hierarchical KGC architecture. In the paper, we describe a basic mathematical constructions applied in ID-based cryptosystems. We survey fundamental ID-based cryptographic primitives: Key extraction, Key Escrow, Encryption, Digital Signature, Identification Scheme and Key Agreement, which are based on the mathematical concepts of Integer Factorization, Quadratic Residues, Discret Logarithms, and Bilinear Pairings. We review several schemes to illustrate different approaches and practical solutions.

Keywords: *ID-based cryptography, public key cryptography, elliptic curve, bilinear pairing.*

Введение

Криптографическая система на основе идентификаторов (IDentity-Based cryptosystem, IDB-system) — это асимметричная криптографическая система, в которой открытые ключи вычисляются по общедоступному алгоритму на основе идентификационной информации их владельцев (в дальнейшем для краткости будем называть такие системы IDB-*системами*).

Личные ключи вырабатываются *центром генерации ключей* KGC (Key Generation Center) на основе идентификационной информации и выдаются владельцам открытых ключей при личной встрече либо с использованием защищённого канала. Идентификационная информация для формирования открытого ключа может включать:

идентификаторы пользователей информационной системы; любую персональную информацию (адрес электронной почты, фотографию, номер телефона, почтовый адрес и т. п.); любые термины и условия, такие, как действующая политика, время, выполняемая роль, любые факты, связанные с конкретной стороной. Поэтому часто такие системы определяют как системы, в которых личные ключи формируются на основе открытых, а последние могут представлять собой произвольные текстовые строки.

Поскольку ключи однозначно определяются по идентификационной информации, то необходимость в сертификатах открытых ключей отпадает, а следовательно, отпадает необходимость в создании инфраструктуры открытых ключей, содержащей множество удостоверяющих центров. Поэтому данная технология формирования ключей представляется весьма перспективной и удобной для практических применений.

Первые работы по данному направлению появились около сорока лет назад, но активный поиск новых конструкций и подходов к их построению продолжается и в настоящее время. Общее число публикаций по данному направлению уже составляет несколько сотен. Хорошие обзоры по различным способам построения и разнообразным приложениям IDB-систем содержатся в работах [7, 10, 15, 16, 20, 27, 38, 57].

Многие IDB-системы доведены до включения в международные стандарты. Так, например, в стандарте ISO/IEC 14888-2:2007 [69] описана IDB-схема цифровой подписи GQ1 на основе RSA. В стандарте ISO/IEC 14888-3:2018 [70] приведены три IDB-схемы цифровой подписи:

— системы IBS-1 и IBS-2 на основе GDH-групп[1] [12];
— китайская система IBS на основе билинейного спаривания [73].

В стандарте ISO/IEC 11770-3:2015 [71] описаны два IDB-протокола выработки общего ключа:

— на основе системы Смарта — Чена — Ченга (N. Smart, L. Chen, Z. Cheng) [17];
— на основе системы Фуджиока — Сузуки — Устаоглу (Fujioka, Suzuki, Ustaoglu),

и один IDB-протокол защищённой передачи ключа:

— на основе системы Сакаи — Касахары [55].

Стандарт IEEE P1363a-2004 [72] определяет четыре типа криптосистем:

— IBS-системы шифрования;
— IBS-системы инкапсуляции ключа (Key Encapsulation);
— IBS-системы цифровой подписи;
— IBS-системы одновременного шифрования и подписи (Signcryption).

Кроме того, рабочая группа IETF S/MIME выпустила несколько проектов, касающихся криптографических методов на основе идентификаторов.

Есть также несколько опубликованных RFC, которые доказывают интерес и доверие научного сообщества к этой криптографической технике (таблица).

[1] GDH (Gap Diffie — Hellman groups) — таким термином обозначают класс циклических групп, для которых вычислительная проблема Диффи — Хеллмана (CDHP) является трудной, в то время как проблема распознавания Диффи — Хеллмана (DDHP) оказывается простой.
Более точно: пусть G — аддитивная группа и $a, b, c \in \mathbb{Z}_p$.

1. Computation Diffie — Hellman Problem (CDHP): для (P, aP, bP) вычислить abP.
2. Decisional Diffie — Hellman Problem (DDHP): для (P, aP, bP, cP) распознать, когда $c = ab$ в $\mathbb{Z}_p$.

Группа G является GDH-группой, если DDHP разрешима за полиномиальное время, но никакой вероятностный полиномиальный алгоритм не сможет решить CDHP с непренебрежимо малым преимуществом за полиномиальное время.

Документы IETF, содержащие описание IDB-систем

Номер	Год	Название
RFC 1824	1995	IBC Protocol for Authenticated Key-Exchange
RFC 5091	2007	IBC Standard #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems
RFC 5408	2009	IBE Architecture and Supporting Data Structures
RFC 5409	2009	Using the BF and BB1 Algorithms with the Cryptographic Message Syntax
RFC 6267	2011	MIKEY-IBAKE: Identity-Based Authenticated Key Exchange (IBAKE) Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)
RFC 6508	2012	Sakai-Kasahara Key Encryption (SAKKE)
RFC 6509	2012	MIKEY-SAKKE: Sakai-Kasahara Key Encryption in Multimedia Internet KEYing (MIKEY)
RFC 6539	2012	IBAKE: Identity-Based Authenticated Key Exchange
RFC 7859	2016	Identity-Based Signatures for Mobile Ad Hoc Network (MANET) Routing Protocols

Далее работа построена следующим образом: в п. 1 обсуждаются особенности практического применения IDB-систем и возникающие при этом проблемы. В п. 2 приведены основные математические конструкции, применяемые для формирования личных ключей пользователей, соответствующих открытым ключам на основе их идентификаторов. В п. 3 рассмотрены конкретные примеры криптографических IDB-систем: системы шифрования (включая иерархические, анонимные, с использованием сертификатов и без них), цифровой подписи (включая signcryption- и higncryption-системы), аутентификации сторон, а также системы выработки общего ключа и замены ключа.

1. Особенности практического применения

Перечислим особенности практического применения IDB-систем, их недостатки и способы их устранения при построении приложений.

1.1. Масштабирование на несколько регионов

Нельзя утверждать, что из-за отсутствия необходимости в сертификатах потребность в инфраструктуре полностью отпадает. Для формирования личных ключей подписи пользователей необходима доверенная третья сторона — центр генерации ключей KGC. Он формирует личные ключи на основании идентификационной информации их владельцев, используя свой *главный* (*мастер-*) *ключ*, представляющий собой ключевую пару (открытый/закрытый мастер-ключ). При этом мастер-ключ должен быть один, и он должен принадлежать только центру генерации ключей, который должен быть также один.

Это накладывает ограничение на максимальное количество пользователей, так как они должны быть зарегистрированы и должны получить ключевые пары только в одном таком центре. Значит, круг использования этой технологии ограничен сотрудниками одного региона, одной организации, предприятия, либо клиентами одного банка или какого-либо другого поставщика услуг.

Для расширения возможностей данной технологии применяются так называемые иерархические IDB-системы HIDE (Hierarchical IDentity-based Encryption), в которых имеется множество локальных центров KGC, образующих древовидную структуру. Корнем дерева является главный центр, а центр каждого нижележащего уровня выдаёт ключи центрам следующего уровня. Пользователям соответствуют листья этого дерева. При этом открытый ключ каждого пользователя должен зависеть от идентификаторов центров, лежащих на пути от этой вершины до корня дерева. Такой способ формирования открытых ключей приводит к усложнению процедуры отзыва и обновления ключей.

1.2. Обеспечение анонимности в иерархических IDB-системах

В системах открытого шифрования условие анонимности получателя обычно обеспечивается путём шифрования идентификаторов получателя вместе с отправляемым сообщением. Поэтому убедиться в правильности адресата может только тот, кто владеет личным ключом получателя. С другой стороны, для формирования такого сообщения необходимо знать актуальную информацию об открытом ключе получателя, которая содержится в действующем сертификате открытого ключа. Этот сертификат должен быть своевременно получен отправителем, причём он пересылается, как правило, в открытом виде. Поэтому в коммуникационных сетях противник, анализируя трафик, имеет возможность получить информацию о получателе.

IDB-системы исключают такую возможность, поскольку там отсутствуют сертификаты, а открытый ключ вычисляется непосредственно из идентификационной информации получателя. Поэтому такие системы очень удобны для построения анонимных (иерархических) систем шифрования (Anonymous (Hierarchical) Identity-Based Encryption, A(H)IBE) для анонимных коммуникационных систем, в которых по шифртексту невозможно определить ни отправителя, ни получателя. Это понятие введено впервые в работе М. Абдала и др. [1]. Примеры A(H)IBE-схем получаются, например, на основе схемы Боне — Франклина.

Помимо коммуникационных систем, данные конструкции применяются при удалённом получении информации из баз данных. В той же работе [1] изучались *системы открытого шифрования с возможностью поиска ключевых слов* (Public-key Encryption with Keyword Search, PEKS). PEKS — это системы, в которых шифртекст ассоциирован с ключевым словом, причём выполняется требование о невозможности получения никакой информации об этом слове. Пользователь получает в центре KGC, помимо ключевой пары, ещё однонаправленную функцию для каждого используемого им ключевого слова. Для получения всех записей, содержащих ключевое слово, он может обратиться к администратору удалённой базы данных и передать ему соответствующую однонаправленную функцию. Администратор может выбрать все такие зашифрованные записи, причем он не сможет получить никакой информации ни о ключевом слове, ни о содержащейся в этих зашифрованных записях информации.

1.3. Депонирование ключа в IDB-системах

Критичным свойством IDB-систем является присущая им возможность создания системы депонирования личных ключей. Центр KGC, обладая мастер-ключом, имеет возможность вычислять ранее выданные личные ключи всех пользователей. Это позволяет без труда создать легитимную систему депонирования личных ключей, при которой при наличии решения суда правоохранительные органы могут запросить применяемые ключи у центра, а центр обязан предоставить личный ключ указанного пользователя. При этом центру нет необходимости хранить эти ключи, так как он может их заново вычислить по идентификационной информации.

1.4. Защита от нечестного центра KGC

В криптографических системах с инфраструктурой открытых ключей PKI на основе стандарта X.509 удостоверяющий центр, входящий в инфраструктуру PKI, отвечает только за подлинность соответствия между указанными в сертификате открытым ключом и идентификатором пользователя. Удостоверяющий центр выдаёт только сертификат ключа, а свой личный ключ пользователь может держать в секрете, не предъявляя его центру. Единственное, что пользователь должен сделать при получе-

нии сертификата открытого ключа, — это доказать наличие у него второй половины ключевой пары, что может быть произведено без раскрытия этого ключа.

В системах с открытыми ключами на основе идентификационной информации центр KGC сам формирует личные ключи всех пользователей. Поэтому большим недостатком IDB-систем является то, что нечестный центр KGC, обладая мастер-ключом, имеет возможность самостоятельно вычислять текущие и выданные ранее личные ключи всех пользователей по их идентификаторам, а поэтому и читать шифрованную переписку и подделывать цифровую подпись каждого пользователя.

Поэтому необходимо дополнить IDB-систему механизмами, позволяющими защитить пользователей от нечестного поведения центра KGC, например применяя протокол доказательства с нулевым разглашением. Такой подход используется в системах шифрования с извлечением ключа вслепую, в которых пользователь имеет возможность получить личный ключ, не раскрывая центру ни ключа, ни своего идентификатора, и в системах защиты цифровой подписи от подделки подписи со стороны центра [18], где с помощью такого доказательства пользователь, сохраняя в тайне свой личный ключ, может доказать на его основе арбитру, что это не его подпись.

1.5. IDB-системы на основе сертификатов (CBC-системы)

Другой способ защиты от нечестного центра KGC предоставляют *системы с открытыми ключами, построенные на основе сертификатов* (Certificate-Based Cryptography, CBC) и сохраняющие преимущества PKI и IDB-систем. Теперь каждый пользователь сам формирует свою ключевую пару и запрашивает сертификат в доверенном сертификационном центре CA (Certificate Authority). При этом центр CA формирует сертификат с помощью IDB-алгоритма, но при этом он не может восстановить личный ключ пользователя. Такие системы уже не относятся непосредственно к IDB-системам, но сочетают в себе преимущества обычных систем с открытыми ключами и IDB-систем.

1.6. Системы без сертификатов (CLC-системы)

Ещё одним направлением исследований является *открытая криптография без сертификатов* (CertificateLess Cryptography, CLC) [57], где также решается проблема депонирования ключа центром, унаследованная от систем на основе идентификаторов. В данном случае не требуются ни сертификаты, ни инфраструктура PKI. Вместо них доверенная сторона — центр KGC — формирует частичные личные ключи аналогично IDB-системам. Действующий личный ключ пользователя получается путём объединения полученного частичного ключа и выбранного им самим секрета. Поэтому он остаётся неизвестным и не хранится в центре KGC, что устраняет проблему депонирования ключа центром KGC.

Хотя такие системы имеют много общего с системами шифрования на основе сертификатов, каждый подход имеет свои достоинства и отличительные особенности.

1.7. Как заменить скомпрометированные ключи?

Ещё одной проблемой IDB-систем является отзыв и замена скомпрометированных ключей. Для любых систем с открытыми ключами, основанных на PKI или на ID, должна быть обеспечена процедура отзыва скомпрометированных ключей. В традиционных PKI это решается путём включения в сертификат предустановленного срока годности и ведения актуального списка аннулированных сертификатов. В IDB-системах с открытыми ключами на основе идентификаторов замена ключей представляет проблему, поскольку непонятно, как можно заменять имеющиеся идентификаторы. Самое простое практическое решение для облегчения процедуры отзы-

ва ключей предполагает дополнение идентификаторов некоторой заменяемой информацией, например, сроком действия, ключевым словом и т. п. Более того, можно предусмотреть регулярную замену ключей независимо от того, был ли ключ скомпрометирован или нет, например используя идентификационную информацию вида "`receiver-address`$\|$`current-date`" [20, 22], где `date` может быть днём, неделей, месяцем или годом.

Такой способ оказывается неудобным для систем с большим числом пользователей, поскольку они должны постоянно контактировать с одним доверенным центром KGC, нагрузка на который резко возрастает. Кроме того, важную роль приобретает проблема обеспечения аутентичности самой идентификационной информации, так как отправители могут путать похожие наборы данных и тем самым неправильно формировать открытые ключи получателя. В результате они будут ненамеренно отправлять зашифрованные сообщения не тем адресатам, причём последние смогут прочитать содержащуюся в них информацию.

Поэтому в нескольких работах предложены IDB-системы с процедурой отзыва ключей (IDB-cryptosistem with revocation) [6, 40, 42, 58, 39], где предлагаются специальные математические конструкции, ускоряющие и облегчающие этот процесс и позволяющие упростить работу центра KGC, заменив оценку трудоёмкости с линейной на логарифмическую от числа пользователей, и при этом сохранить простоту работы для самих пользователей.

1.8. Другие приложения

По данным Voltage, сегодня технология IBE защищает данные для более чем 100 миллионов пользователей по всему миру и совместима с такими широко распространенными продуктами, как Outlook, Yahoo, Gmail и др. В [33] содержится большой обзор применений IDB-систем для сенсорных сетей, где отмечается, что в приложениях с ограниченными доступными системными ресурсами факторы, касающиеся общей производительности системы, становятся гораздо более важными. Благодаря экономичности и низким требованиям к инфраструктуре, IDB-системы хорошо соответствуют требованиям к таким сетям. Их применение приводит к значительному сокращению накладных расходов на связь, а также к более рациональному использованию пропускной способности. Иерархические модели, которые могут быть получены из IBC, по-видимому, идеально вписываются в инфраструктуру сенсорной сети. Более того, ряд специальных, ключевых особенностей IBC (распределённый центр генерации ключей, системная иерархия, отзыв ключей, делегирование) решаются простым и элегантным способом.

Далее при описании протоколов будем использовать следующие обозначения:

— $a \in_R \mathbb{Z}_n$ — выбрать случайный элемент a из $\mathbb{Z}_n$;
— $F \stackrel{?}{=} G$ — проверить совпадение F и G;
— $\mathbb{G} = \langle P \rangle$ — циклическая группа $\mathbb{G}$ порождена элементом P.

В случаях, когда это не вызывает разночтений, будем для упрощения записи опускать знак конкатенации в аргументе хеш-функции $h(x, y, \ldots, z) = h(x\|y\| \ldots \|z)$.

2. Математические конструкции для формирования личных ключей на основе идентификаторов

Формирование личных ключей пользователей (key extraction), соответствующих открытым ключам на основе их идентификаторов, производится в центрах KGC. Перечислим применяемые при этом основные алгоритмы формирования ключевых пар.

2.1. IDB-системы на основе RSA

Первый способ построения IDB-системы предложил А. Шамир в 1984 г. [60]. За основу была взята система RSA, основанная на сложности задачи факторизации целых чисел. Пусть $h : \{0,1\}^* \to \mathbb{Z}_n$ — хеш-функция, $n = pq$.

Открытый ключ центра KGC — это пара (e, n), где e — большое простое число, не являющееся делителем числа $(p-1)(q-1)$.

Закрытый ключ центра — (p, q, d), где $d = e^{-1} \bmod (p-1)(q-1)$.

Пользователь A предъявляет центру свою идентификационную информацию $\mathrm{ID_A}$ и центр выдаёт ему ключевую пару $(pk_A, sk_A) = (h(\mathrm{ID_A}), pk_A^d)$, т. е. $h(\mathrm{ID_A}) = sk_A^e \bmod n$.

Значения d и $sk_A = pk_A^d \bmod n$ центр может вычислить, зная разложение числа n.

2.2. IDB-системы на основе квадратичных вычетов

В 2001 г. в работе [19] К. Кокс (C. Cocks) предложил схему на основе сложности задачи нахождения квадратного корня в кольце вычетов по составному модулю: $x \in \mathbb{Z}_n$, $a = x^2 \bmod n$, $n = pq$, p, q — различные большие простые числа, удовлетворяющие условию $p \equiv q \equiv 3 \pmod 4$.

Пусть $J(n)$ — множество всех элементов кольца $\mathbb{Z}_n$, которые имеют символ Якоби равный 1; $QR(n) \subset J(n)$ — множество всех квадратичных вычетов по модулю n; $u \in J(n) \setminus QR(n)$; $h\colon \{0,1\}^* \to J(n)$ — хеш-функция.

Тройка (n, u, h) составляет набор открытых параметров.

Заметим, что из условия $a \in J(n)$ следует, что $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$ или $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$. Поэтому либо a, либо $(-a)$ является квадратичным вычетом по модулю n.

При этом извлечь квадратный корень может только центр KGC, знающий числа p и q, являющиеся секретом KGC. Для этого он может воспользоваться формулой

$$r = a^{(n+5-(p+q))/8} \bmod n.$$

Такое r удовлетворяет условию $r^2 = a \pmod n$ или $r^2 = -a \pmod n$ в зависимости от того, что является квадратичным вычетом: a или $-a$.

Пользователь A предъявляет центру свою идентификационную информацию $\mathrm{ID_A}$ и центр выдаёт ему ключевую пару $(pk_A, sk_A) = (a, r)$, где:

— открытый ключ пользователя равен $a = h(\mathrm{ID_A}) \in J(n)$;

— личный ключ пользователя определяется как

$$r = \begin{cases} \sqrt{a} \bmod n, & a \in QR(n); \\ \sqrt{ua} \bmod n, & a \in J(n) \setminus QR(n). \end{cases} \tag{1}$$

2.3. IDB-системы на основе задачи логарифмирования в мультипликативной группе поля

К. Гюнтер (C. G. Günther) в [32] предложил способ построения IDB-системы на основе цифровой подписи центра KGC, вычисленной по схеме Эль Гамаля. Закрытым и открытым ключами центра KGC являются соответственно элементы $x_S \in \{1, \ldots, p-1\}$ и $y_S = g^{x_S} \in \mathbb{Z}_p^*$, где p — большое простое число.

Пользователь A получает в KGC цифровую подпись (u_A, v_A) для своего идентификатора ID_A, где

$$u_A = g^{k_A}; \quad v_A = (\mathrm{ID}_A - x_S u_A) k_A^{-1} \bmod (p-1); \quad k_A \in_R \mathbb{Z}_p^*; \quad (k_A, p-1) = 1.$$

Проверка подписи проводится с помощью уравнения

$$u_A^{v_A} = g^{\mathrm{ID}_A} y_S^{-u_A}.$$

Теперь ключевая пара пользователя A определяется как

$$(pk_A, sk_A) = ((\mathrm{ID}_A, u_A), v_A),$$

где в роли открытого ключа пользователя A выступает его идентификатор и первая половина подписи, а в роли личного ключа — вторая половина подписи $v_A = \log_{u_A}(g^{\mathrm{ID}_A} y_S^{-u_A})$. Поэтому проблема определения личного ключа по открытому ключу сводится к проблеме дискретного логарифмирования в $\mathbb{Z}_p^*$.

2.4. IDB-системы на основе группы точек эллиптической кривой

Фактически, все первоначально разработанные IDB-схемы на основе группы точек эллиптической кривой предполагают наличие операции *билинейного спаривания*. В общем случае эта операция определяется на двух абелевых группах $\mathbb{G}_1$ и $\mathbb{G}_2$ простого порядка q и принимает значение в третьей мультипликативной группе $\mathbb{G}_T$ того же порядка:

$$e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T.$$

При этом должны выполняться два свойства:

1) *билинейность*: при всех $w, x \in \mathbb{G}_1$ и $y, z \in \mathbb{G}_2$ выполнены тождества

$$e(w, x+z) = e(w,x)\, e(w,z), \quad e(w+x, z) = e(w,z)\, e(x,z);$$

2) *невырожденность*: для некоторых элементов $x \in \mathbb{G}_1$ и $y \in \mathbb{G}_2$ выполнено $e(x,y) \neq 1$.

В практических применениях используются в основном операции спаривания для групп точек эллиптической кривой. Поэтому будем рассматривать только такие операции. В этом случае группы $\mathbb{G}_1$ и $\mathbb{G}_2$ являются одинаковыми или различными циклическими подгруппами группы точек эллиптической кривой над конечным полем или его расширением, а $\mathbb{G}_T$ — это подгруппа мультипликативной группы поля. При рассмотрении групп точек эллиптических кривых над конечным полем будем обозначать точки большими латинскими буквами, а элементы поля — маленькими.

Если $P \in \mathbb{G}_1$ и $n \in \mathbb{N}$, то для кратных точек будем использовать обозначение

$$[n]P = \underbrace{P + \ldots + P}_{n}.$$

В [10] выделено два типа IDB-схем, различающихся по способу формирования ключевых пар участников:

— тип SOK (от Сакаи — Огиши — Казахара (R. Sakai, K. Ohgishi, M. Kasahara) [53]) предполагает наличие двух хеш-функций h_1 и h_2, принимающих значение соответственно в группах $\mathbb{G}_1$ и $\mathbb{G}_2$;

— тип SK (от Сакаи — Казахара (R. Sakai, M. Kasahara) [55]) использует только одну хеш-функцию h, принимающую числовое значение, которое получается представлением двоичного вектора — значения хеш-функции соответствующим натуральным числом.

Для обоих типов ключевая пара центра KGC имеет вид $([s]P, s)$, где s — закрытый ключ $(1 < s < q-1)$; $[s]P$ — открытый ключ; P — образующий элемент группы $\mathbb{G}_1$.

Для IDB-схем типа SOK ключевая пара пользователя A определяется как

$$(pk_A, sk_A) = (h_1(\mathrm{ID}_A), [s]h_1(\mathrm{ID}_A)).$$

Для IDB-схем типа SK ключевая пара пользователя A определяется как

$$(pk_A, sk_A) = ([s + h(\mathrm{ID}_A)]P, [s + h_1(\mathrm{ID}_A)]^{-1}P),$$

где открытый ключ участника A зависит от закрытого ключа s центра KGC, причём выполняется соотношение $e(pk_A, sk_A) = e(P, P)$.

2.5. IDB-системы на основе нечётких множеств

В 2005 г. А. Сахаи и Б. Уотерс (A. Sahai и B. Waters) [56] предложили новый способ построения IDB-системы на основе нечётких множеств, в которой идентификатор рассматривается как набор описательных атрибутов.

Нечёткая схема IBE позволяет использовать закрытый ключ, соответствующий открытому ключу с идентификатором ω, для расшифрования текста, зашифрованного на открытом ключе с идентификатором ω', тогда и только тогда, когда идентификаторы ω и ω' находятся близко друг к другу, что определяется метрикой, позволяющей оценить «перекрытие» множеств: $|\omega \cap \omega'| \geqslant d$ при некотором заданном параметре d.

Нечёткая схема IBE может быть применена для обеспечения шифрования с использованием в качестве идентификационной информации биометрических входных данных. Поскольку биометрические данные обладают некоторым шумом, их использование в обычных IDB-системах невозможно. Однако свойство устойчивости к ошибкам нечёткой схемы IBE позволяет расшифровать зашифрованные данные с помощью личного ключа, восстановленного из биометрических данных, который может отличаться от истинного личного ключа.

Кроме того, Fuzzy-IBE можно использовать для приложений, которые называются «шифрованием на основе атрибутов». В таких приложениях стороны отправляют зашифрованные сообщения всем пользователям, имеющим заданное множество атрибутов, например «члены комиссии», «сотрудники отдела» и т. п., которые составляют (нечёткую) идентификацию. Поэтому такие сообщения могут прочитать только те, у кого в идентификационной информации есть соответствующие атрибуты.

Преимуществом такого подхода является то, что документы могут храниться на обычном сервере без доверенных средств (удалённой) аутентификации.

Благодаря этому свойству IDB-системы на основе нечётких множеств применяются в системах, допускающих эффективную процедуру отзыва ключей. Далее в п. 3.1 описана нечёткая система шифрования Fuzzy-IBE (Fuzzy Identity-Based Encryption) из [56], а в п. 3.2 — её применение в системах с процедурой отзыва ключей из [38].

2.6. Иерархические IDB-системы

Для масштабирования IDB-систем с целью применения их в больших распределённых системах в [23] предложена иерархическая структура IDB-системы, в которой может быть несколько локальных центров KGC, образующих древовидную структуру, корнем которой является корневой центр генерации ключей rootPKG, а каждый центр выдаёт ключи только связанным с ним центрам следующего уровня (рис. 1). Листьям дерева соответствуют пользователи.

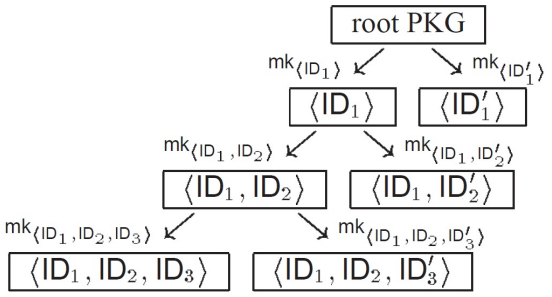

Рис. 1. Иерархическая структура локальных центров [23]

Каждый локальный центр PKG, соответствующий вершине, расположенной на i-м уровне, однозначно характеризуется набором идентификаторов $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$, соответствующих пути от корня к этой вершине. Поэтому в качестве открытого ключа для этого центра можно выбрать значение $h(\mathrm{ID}_1 \| \ldots \| \mathrm{ID}_i)$, где h — некоторая хеш-функция, а закрытый ключ $\mathrm{mk}_{(\mathrm{ID}_1,\ldots,\mathrm{ID}_i)}$ формируется на основе открытого ключа.

В п. 3.1 приведён пример иерархической IDB-системы шифрования HIDE [23].

Другие способы формирования ключевых пар пользователей рассмотрены далее при описании конкретных классов IDB-систем.

3. Примеры криптографических IDB-систем

3.1. IDB-системы шифрования (IBE)

IBE-схема на основе квадратичных вычетов

В [19] К. Кокс (C. Cocks) предложил IDB-схему шифрования, использующую описанный в п. 2.2 способ формирования ключевых пар на основе проблемы извлечения квадратного корня в кольце вычетов $\mathbb{Z}_n$, $n = pq$, где p и q — простые числа, удовлетворяющие условию $p \equiv q \equiv 3 \pmod 4$. Пользователь A предъявляет центру свою идентификационную информацию ID_A и получает ключевую пару $(pk_A, sk_A) = (a, r)$, где $a = h(\mathrm{ID}_A) \in J(n)$, а r находится по формуле (1).

Зашифрование происходит побитно: для зашифрования бита m (который закодирован $+1$ или -1) надо:

1) выбрать случайные элементы $t_0, t_1 \in \mathbb{Z}_n$;
2) вычислить

$$d_i = \frac{t_i^2 + u^i a}{t_i}, \quad c_i = m\left(\frac{t_i}{n}\right), \quad i \in \{0, 1\}.$$

Шифртекст состоит из двух элементов $((d_o, c_0), (d_1, c_1))$. Поэтому при зашифровании одного бита получится $O(\log n)$ битов шифртекста.

Расшифрование:

1) определить $i \in \{0, 1\}$, такое, что $r^2 = u^i a$;
2) вычислить $g = d_i + 2r$, которое можно записать так:

$$g = d_i + 2r = \frac{t_i^2 + r^2}{t_i} + 2r = \frac{(t_i + r)^2}{t_i} = \left[\frac{t_i + r}{t_i}\right]^2 t_i.$$

Отсюда следует, что $\left(\frac{g}{n}\right) = \left(\frac{t_i}{n}\right)$;

3) вычислить $m = c_i \left(\frac{t_i}{n}\right)$.

IBE-система Боне — Франклина на основе билинейного спаривания (BF-IDE)

Первая практически эффективная иерархическая система предложена в работе D. Boneh и M. Franklin в 2001 г. [9]. Она основана на операции спаривания для эллиптических кривых $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, где $\mathbb{G}_1$ — подгруппа группы точек эллиптической кривой; $\mathbb{G}_T$ — подгруппа мультипликативной группы поля. Пусть также имеются хеш-функции $h_1\colon \{0,1\}^* \to \mathbb{G}_1$ и $h_2\colon \mathbb{G}_T \to \{0,1\}^n$.

Центр KGC обладает ключевой парой (s, H), $H = [s]P$, $P \in \mathbb{G}_1$.

Участник A получает в центре KGC открытый ключ $Q_A = h_1(\mathrm{ID}_A)$ и личный ключ $S_A = [s]Q_A$. Для зашифрования сообщения $M \in \{0,1\}^*$, отправляемого участнику A, надо выбрать случайное значение $r \in \mathbb{Z}_q^*$ и сформировать шифртекст

$$C = (\,[r]P, M + h_2(e(Q_A, H)^r)\,) = (U, V).$$

Получив это сообщение, A вычисляет открытый текст по формуле

$$M = C + h_2(e(S_A, U)),$$

так как $e(S_A, U) = e([s]Q_A, [r]P) = e(Q_A, [s]P)^r = e(Q_A, H)^r$.

Данная схема в качестве конструктивного блока нашла многочисленные практические применения в протоколах для иерархических, облачных, широковещательных и др. IDB-систем [23, 22, 34, 2, 67, 14].

Позднее Д. Галиндо (D. Galindo) [21] обнаружил уязвимость этого протокола и предложил исправленный и улучшенный вариант.

Система Fuzzy-IBE на основе нечётких множеств

В нечёткой системе шифрования Fuzzy-IBE (Fuzzy Identity-Based Encryption), предложенной в 2005 г. А. Сахаи и Б. Уотерс (A. Sahai, B. Waters) [56], идентификатор рассматривается как набор описательных атрибутов.

Пусть $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ — операция билинейного спаривания для групп $\mathbb{G}_1$ и $\mathbb{G}_T$ простого порядка p и P_0 — образующий элемент группы $\mathbb{G}_1$. Определим коэффициент Лагранжа для $i \in \mathbb{Z}_p$ и $S \subset \mathbb{Z}_p$ равенством

$$\delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j} = \begin{cases} 1, & x = i \in S; \\ 0, & x \in \mathbb{Z}_p \setminus \{i\}. \end{cases} \tag{2}$$

Пусть идентификаторами являются подмножества ω некоторого множества $\mathcal{U}$. Множество $\mathcal{U}$ может быть множеством всевозможных атрибутов или множеством оцифрованных значений некоторой биометрической системы распознавания. Каждому элементу подмножества ω однозначно сопоставлен некоторый элемент из $\mathbb{Z}_p^*$. Пусть для простоты элементам из $\mathcal{U}$ соответствуют первые $|\mathcal{U}|$ элементов $\{1, \ldots, |\mathcal{U}|\}$ из $\mathbb{Z}_p^*$.

Выберем случайно и равновероятно $t_1, \ldots, t_{|\mathcal{U}|}, y \in_R \mathbb{Z}_p^*$. Открытыми параметрами центра являются

$$(T_1 = [t_1]P_0,\ \ldots,\ T_{|\mathcal{U}|} = [t_{|\mathcal{U}|}]P_0,\ Y = e(P_0, P_0)^y),$$

а мастер-ключ определяется как $(t_1, \ldots, t_{|\mathcal{U}|}, y)$.

Пользователь с идентификатором $\omega \subset \mathcal{U}$ получает личный ключ на основе случайного многочлена $q(x) \in \mathbb{Z}_p[x]$ степени $d-1$ с $q(0) = y$ как упорядоченный набор $(D_i)_{i\in\omega}$, где $D_i = \left[\dfrac{q(i)}{t_i}\right] P_0$, $i \in \omega$.

Зашифрование. Для зашифрования сообщения $m \in \mathbb{G}_T$ на открытом ключе ω' выбирают случайный элемент $r \in \mathbb{Z}_p$ и формируют шифртекст вида

$$C = (\omega',\ c' = mY^r,\ \{C_i = [r]T_i\}_{i\in\omega}).$$

Расшифрование. Если шифртекст получен на ключе ω', то для его расшифрования на ключе ω, удовлетворяющем условию $|\omega \cap \omega'| \geqslant d$, следует выбрать произвольное подмножество $\sigma \subseteq \omega \cap \omega'$ мощности d и вычислить

$$c' / \prod_{i\in\sigma} e(G_i, C_i)^{\delta_{i,\sigma}(0)} = m \cdot e(P_0, P_0)^{ry} / \prod_{i\in\sigma} e\left(\left[\frac{q(i)}{t_i}\right] P_0, [rt_i]P_0\right)^{\delta_{i,\sigma}(0)} =$$

$$= m \cdot e(P_0, P_0)^{ry} / \prod_{i\in\sigma} \left(e(P_0, P_0)^{rq(i)}\right)^{\delta_{i,\sigma}(0)} = m \cdot e(P_0, P_0)^{ry} / e(P_0, P_0)^{r\sum\limits_{i\in\sigma} q(i)\delta_{i,\sigma}(0)} = m.$$

Данное выражение вытекает из формулы интерполяции стоящего в показателе многочлена $q(x)$ степени $d-1$ по d точкам:

$$\sum_{i\in\sigma} q(i)\delta_{i,\sigma}(0) = q(0) = y.$$

Иерархические IBE-системы

В [23] предложена иерархическая структура IDB-системы — HIDE (Hierarchical IDentity-based Encryption), в которой локальные центры KGC образуют древовидную структуру, корнем которой является корневой центр генерации ключей rootPKG, а каждый центр выдаёт ключи только связанным с ним центрам следующего уровня (см. рис. 1). Конечные вершины, соответствующие листьям этого дерева, представляют пользователей.

Данная система была основана на конструкции Боне и Франклина [9].

Параметры корневого центра rootPKG:

1) Пусть p — k-битовое простое число, $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ — операция спаривания, где $\mathbb{G}_1, \mathbb{G}$ — циклические группы порядка p; P_0 — случайно выбранный образующий элемент группы $\mathbb{G}_1$.
2) При некотором $n > 0$ выбираются криптографические хеш-функции

$$h_1\colon \{0,1\}^* \to \mathbb{G}_1,$$
$$h_2\colon \mathbb{G}_T \to \{0,1\}^n,$$
$$h_3\colon \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_p,$$
$$h_4\colon \{0,1\}^n \to \{0,1\}^n.$$

3) Выбирается случайный элемент $s_0 \in \mathbb{Z}_p^*$ — корневой мастер-ключ, и формируется ключевая пара центра rootPKG $(s_0, Q_0 = [s_0]P_0)$.

Формирование ключей для центра, расположенного на i-м уровне.

При $i \geqslant 1$ каждый центр KGC_i, расположенный на i-м уровне, выбирает в качестве секрета элемент $s_i \in \mathbb{Z}_p^*$, $s_i \neq s$, и вычисляет $Q_i = [s_i]P_0$.

Центр KGC_i однозначно характеризуется набором идентификаторов $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$, соответствующих пути от корня к этому центру. В качестве открытого ключа он выбирает значение $P_i = h_1(\mathrm{ID}_1 \| \ldots \| \mathrm{ID}_i)$, а для получения закрытого ключа обращается в центр KGC_{i-1} с идентификатором $(\mathrm{ID}_1 \| \ldots \| \mathrm{ID}_{i-1})$.

Пусть центр KGC_{i-1} обладает секретом s_{i-1} и мастер-ключом

$$mk_{ID_{i-1}} = (S_{i-1}, Q_1, \ldots, Q_{i-1})$$

(при $i = 1$ центр KGC_1 обладает секретом s_1 и мастер-ключом $mk_1 = (S_1, Q_1)$, где S_1 — единичный элемент группы $\mathbb{G}_1$).

Для вычисления личного ключа центра KGC_i центр KGC_{i-1}:

1) вычисляет $P_i = h_1(\mathrm{ID}_1 \| \ldots \| \mathrm{ID}_i)$;
2) вычисляет $S_i = S_{i-1} + [s_{i-1}]P_i = \sum\limits_{j=1}^{i} [s_{j-1}]P_i$;
3) возвращает $(S_i, Q_1, \ldots, Q_{i-1})$, где $Q_j = [s_j]P_0$, $1 \leqslant j \leqslant i-1$.

Теперь подчинённый ему центр KGC_i получает закрытый ключ

$$mk_{ID_i} = (S_i, Q_1, \ldots, Q_i).$$

Для *зашифрования сообщения* $M \in \{0,1\}^*$ для центра KGC_i надо:

1) вычислить $P_i = h_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_j)$, $1 \leqslant j \leqslant i$;
2) вычислить $g = e(Q_0, P_1)$;
3) выбрать случайное $\sigma \in \{0,1\}^n$ и вычислить $r = h_3(\sigma, M)$;
4) определить

$$C = ([r]P_0,\ [r]P_2,\ \ldots,\ [r]P_i,\ \sigma \oplus h_2(rg),\ M \oplus h_4(\sigma)).$$

Расшифрование.

Получив сообщение $C = (U_0, U_2, \ldots, U_i, V, W)$, участник $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ должен:

1) вычислить

$$g' = \frac{e(U_0, S_t)}{\prod\limits_{j=2}^{i} e(Q_{j-1}, U_j)};$$

2) вычислить $\sigma = V \oplus h_2(g')$;
3) вычислить $M = W \oplus h_4(\sigma)$;
4) вычислить $r = h_3(\sigma, M)$. Если $U_0 \neq [r]P_0$, то прервать протокол. Если нет, то принять открытый текст M.

Заметим, что вместо гаммирования $W = M \oplus h_4(\sigma)$ можно использовать любой алгоритм блочного шифрования $W = E_{h_4(\sigma)}(M)$.

IBE-протокол Уотерса

Система шифрования, предложенная B. Waters в [66], использует симметричное спаривание эллиптических кривых $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, $\mathbb{G}_1 = \langle P_0 \rangle$. Закрытый ключ центра — случайный элемент $x \in_R \mathbb{Z}_p$, открытым ключом является $Q = [x]P_0$. Пусть пользователь A характеризуется идентификатором

$$\mathrm{ID}_A = (a_1, \ldots, a_n) \in \{0,1\}^n.$$

Обозначим $\omega = \{i\colon a_i = 1\} \subseteq \{1, \ldots, n\}$.

Для формирования ключа пользователя центр KGC выбирает случайные P_2, U', $U_1, \ldots, U_n \in_R \mathbb{G}_1$, и пусть $\mathbf{U} = (U_1, \ldots, U_n)$. Центр объявляет $(P_0, Q_0, P_2, \mathbf{U})$ открытыми параметрами пользователя и выдаёт ему личный ключ $d = e([x]P_2 + [r]V, [r]P_0)$, где $r \in_R \mathbb{Z}_p^*$ — случайное; $V = U' + \sum\limits_{i\in\omega} U_i$.

Зашифрование.

Для зашифрования сообщения $m \in \mathbb{Z}_p^*$ необходимо вычислить значение $V = U' + \sum\limits_{i\in\omega} U_i$, выбрать случайное $t \in_R \mathbb{Z}_p$ и сформировать шифртекст вида

$$C = (e(Q_0, P_2)^t \cdot m,\ [t]P_0, [t]V).$$

Расшифрование.

Для расшифрования шифртекста $C = (c_1, C_2, C_3)$ надо с использованием личного ключа $d = (d_1, d_2)$ вычислить $m = \dfrac{e(d_2, C_3)}{e(d_1, C_2)} \cdot c_1$.

Действительно,

$$\frac{e(d_2, C_3)}{e(d_1, C_2)} \cdot c_1 = \frac{e([r]P_0, [t]V)}{e([x]P_2 + [r]V, [t]P_0)} \cdot c_1 = \frac{e([r]P_0, [t]V)}{e([x]P_2, [t]P_0) \cdot e([r]V, [t]P_0)} \cdot c_1 =$$
$$= \frac{e(P_0, V)^{rt}}{e([x]P_0, P_2)^t \cdot e(P_0, V)^{rt}} \cdot e(Q_0, P_2)^t \cdot m = m.$$

Данная схема может быть модифицирована в иерархическую IDB-систему, в которой идентификатор центра i-го уровня имеет вид $\mathrm{ID} = (\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$. В этом случае для каждого уровня надо генерировать свои параметры U' и $\mathbf{U}$.

Системы IBE вслепую (Blind IBE)

Для схем шифрования на основе идентификаторов существует протокол извлечения ключа (key extraction protocol), в котором пользователь отправляет строку с идентификационными данными центру KGC, который затем возвращает соответствующий секретный ключ для этого идентификатора. Этот протокол может быть выполнен для нескольких известных схем IBE эффективно вслепую, то есть пользователь может получить секретный ключ, соответствующий его идентификатору, без того чтобы главный центр узнал что-либо об этом идентификаторе. Схемы, поддерживающие протокол извлечения вслепую, называются *системами IBE вслепую* (blind IBE).

В [28] М. Грин и С. Хохенбергер (M. Green, S. Hohenberger) предложили эффективные протоколы извлечения ключа вслепую, удовлетворяющие этому определению, для схем IBE Боне — Бойена [8] и Уотерса [66] (используя обобщение, предложенное независимо Неккаче (D. Naccache) [46] и Чатержи — Сархара (Chatterjee, Sarkar) [13]). Последний протокол похож на схему подписи вслепую Окамото [49].

Рассмотрим эффективный протокол извлечения ключа вслепую, названный BlindExtract, для следующих IBE-систем:

(1) Боне — Бойена [8];
(2) обобщённой версии протокола IBE Уотерса [66], предложенной независимо Нэккаче [46] и Чатержи — Сархаром [13].

Так как обе схемы основаны на одинаковых конструкциях, сначала опишем их общие элементы. Пусть $\mathbb{G}_1 = \langle P \rangle$, $|\mathbb{G}_1| = q$, $f : \{0,1\}^* \to \mathbb{G}_1$ и $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ — операция билинейного спаривания. Выбираем $\alpha \in_R \mathbb{Z}_q^*$ и $H, P_2 \in_R \mathbb{G}_1$ и полагаем $P_1 = [\alpha]P$, $msk = [\alpha]P_2$ — закрытый ключ центра KGC.

Пользователь A отправляет центру свой идентификатор ID_A и получает от центра личный ключ вида

$$(D_0, D_1) = ([\alpha]P_2 + [r]f(\mathrm{ID}_A),\ [r]P) \in \mathbb{G}_1^2,$$

где $r \in_R \mathbb{Z}_q$. Корректность этой пары проверяется тестом

$$e(P_1, P_2)\, e(D_1, f(\mathrm{ID}_A)) = e(D_0, P).$$

Для зашифрования текста $m \in \mathbb{G}_T$ вырабатывается случайное число $s \in_R \mathbb{Z}_q$ и формируется шифртекст

$$C = (e(P_1, P_2)^s \cdot m,\ [s]P,\ [s]f(\mathrm{ID}_A)).$$

При расшифровании шифртекста $C = (c, Y, Z) \in \mathbb{G}_T \times \mathbb{G}_1^2$ используется ключ $(D_0, D_1) \in \mathbb{G}_1^2$ и формируется открытый текст

$$m = c \cdot e(Z, D_1)/e(Y, D_0).$$

Протокол извлечения ключа вслепую BlindExtract для обеих схем IBE (1) и (2) имеет следующий вид (A — пользователь, T — центр):

$$\begin{array}{ll}
A: & y \in_R \mathbb{Z}_q, \\
A \to T: & H' = [y]P + [\mathrm{ID}_A]P_1, \\
A \leftrightarrow T: & \mathrm{ZKproof}(y, \mathrm{ID}_A), \\
T: & r \in_R \mathbb{Z}_q, \\
T: & D_0' = [\alpha]P_2 + [r](H' + H), \\
T: & D_1' = [r]P, \\
A \leftarrow T: & (D_0', D_1'), \\
A: & \text{проверяет } e(P_1, P_2) \cdot e(D_1', H' + H) = e(D_0', P), \\
A: & z \in_R \mathbb{Z}_q, \\
A: & D_0 = D_0' - [y]D_1' + [z]f(\mathrm{ID}_A),\ D_1 = D_1' + [z]P.
\end{array}$$

Через $\mathrm{ZKproof}(y, a)$ здесь обозначен протокол доказательства с нулевым разглашением знания такой пары (y, a), что выполняется равенство $H' = [y]P + [a]P_1$. Этот протокол может быть сделан неинтерактивным, выполняемым однократной передачей сообщения $([r_1]P + [r_2]P_1, x, s_1, s_2)$, где $r_1, r_2 \in_R \mathbb{Z}_q$, $x = h([r_1]P + [r_2]P_1)$, $s_1 = r_1 + xr$, $s_2 = r_2 + xa$. Проверка правильности доказательства осуществляется путём рассмотрения равенства

$$[r_1]P + [r_2]P_1 = [s_1]P + [s_2]P_1 + [x]H'.$$

Различие между системами (1) и (2) заключается в выборе способа записи идентификаторов и конструкции функции $f : \{0,1\}^* \to \mathbb{G}_1$.

Для системы IBE (1) функция f определяется как

$$f(\mathrm{ID}_A) = H + [\mathrm{ID}_A]P_1.$$

Для системы IBE (2) идентификаторы $\mathrm{ID} = (a_1, \ldots, a_n)$, $a_j \in \{0,1\}^l$, выбираются из пространства битовых строк длины $N = nl$, а функция $f : \{0,1\}^N \to \mathbb{G}_T$ определяется как

$$f(\mathrm{ID}_A) = H + \sum_{j=1}^{n} [a_j] U_j,$$

где элементы $U_j \in \mathbb{G}_T$ выбираются центром случайно (Д. Неккаче предлагал использовать значения $N = 160$ и $l = 32$ [46]). Для протокола доказательства с нулевым разглашением ZKproof значение H' вычисляется по формуле $H' = [y]P + \sum_{j=1}^{n} [a_j] U_j$, $0 \leqslant a_i < 2^l$. Личный ключ пользователя, соответствующий идентификатору ID, где $r \in_R \mathbb{Z}_q$, имеет следующий вид:

$$(D_0, D_1) = ([\alpha]P_2 + [r] f(\mathrm{ID}_A), [r]P) = \left([\alpha]P_2 + [r] \left(H + \sum_{j=1}^{n} [a_j] U_j \right), [r]P \right).$$

CBC-системы шифрования

Системы шифрования на основе сертификатов (Certificate-Based Cryptosystem, CBC) помогают избавиться от недостатка обычных IDB-систем шифрования, где центр KGC имеет возможность вычислять личные ключи всех пользователей, а поэтому и знакомиться с содержанием шифрованной переписки каждого пользователя. В то же время они не требует наличия дорогостоящей инфраструктуры PKI, позволяющей постоянно проверять актуальность сертификата.

Сертификат используется как составная часть ключа расшифрования, который составлен непосредственно из сгенерированного пользователем личного ключа и полученного в центре сертификации CA сертификата. Хотя CA знает сертификат, он, в отличие от KGC, не может расшифровать ни одного шифртекста.

Пусть имеется операция симметричного билинейного спаривания для группы $\mathbb{G}_1 = \langle P \rangle$ и две хеш-функции $h_1 : \{0,1\}^* \to \mathbb{G}_1$ и $h_2 : \mathbb{G}_T \to \{0,1\}^n$, где n — длина открытого сообщения, подлежащего зашифрованию.

Центр CA выбирает закрытый ключ $s_C \in_R \mathbb{Z}_p^*$ и открытый ключ $Q_C = [s_C]P$.

Пользователь A обладает ключевой парой $(sk_A, pk_A) = (s_A, [s_A]P)$. Для получения сертификата он обращается в центр CA, отправляя туда сообщение infoA, содержащее значение открытого ключа $Q_A = [s_A]P$ и другую идентификационную информацию. Центр CA вычисляет $\mathcal{Q}_A = h_1(Q_A \,\|\, \mathrm{infoA})$ и возвращает сертификат $\mathrm{cert}_A = [s_C]\mathcal{Q}_A$. В качестве дополнительного аргумента для h_1 может быть включён период действия сертификата.

Теперь A подписывает infoA, формируя значение $[s_A]P_A$, где $P_A = h_1(\mathrm{infoA})$, и вычисляет личный ключ $S_A = \mathrm{cert}_A + [s_A]P_A$. Заметим, что это значение образовано из подписей центра CA и пользователя A под сообщениями $\mathcal{Q}_A$ и P_A соответственно.

Зашифрование сообщения m с использованием infoA выполняется следующим образом: вычисляется $\mathcal{Q}_A = h_1(Q_A \,\|\, \mathrm{infoA})$ и $P_A = h_1(\mathrm{infoA})$, выбирается случайное $t \in_R \mathbb{Z}_p^*$ и формируется шифртекст

$$(U, V) = ([t]P,\ M \oplus h_2(e(Q_C, \mathcal{Q}_A)^t e([s_A]P, P_A)^t)).$$

Расшифрование сообщения (U, V), зашифрованного с помощью infoA и открытого ключа $\mathcal{Q}_A$, осуществляется с помощью личного ключа S_A по формуле

$$M = V \oplus h_2(e(U, S_A)).$$

CLC-системы шифрования

Рассмотрим пример *системы шифрования без сертификатов* (Certificateless Cryptosystem), предложенный в работе [2]. Пусть, как и выше, имеется операция симметричного билинейного спаривания для группы $\mathbb{G}_1 = \langle P \rangle$. Центр выбирает закрытый ключ $s \in_R \mathbb{Z}_p^*$, и пусть открытый ключ $Q = [s]P$. Пусть $h_1 : \{0,1\}^* \to \mathbb{G}_1$ и $h_2 : \mathbb{G}_1^T \to \{0,1\}^n$ — хеш-функции; n — длина открытого сообщения, подлежащего зашифрованию.

Извлечение частичного личного ключа пользователя A осуществляется согласно алгоритму BF-IBE Боне — Франклина: для идентификатора ID_A алгоритм возвращает значение $[s]H_1(\mathrm{ID}_A)$.

Пользователь вырабатывает секретное значение $x \in_R \mathbb{Z}_p^*$ и формирует действующий личный ключ $sk_A = [xs]H_1(\mathrm{ID}_A)$ и открытый ключ $pk_A = (X, Y) = ([x]P, [x]Q)$. Корректность открытого ключа может быть проверена равенством $e(P, Y) = e(Q, X)$.

Зашифрование n-битового сообщения $m \in \mathbb{Z}_p$ с помощью идентификатора ID_A и открытого ключа (X, Y) выполняется следующим образом: проверяется корректность открытого ключа, вырабатывается случайное значение $t \in_R \mathbb{Z}_p^*$ и формируется шифртекст

$$(U, v) = ([t]P,\ m \oplus h_2(e(h_1(\mathrm{ID}_A), Y)^t)).$$

Для расшифрования шифртекста (U, V) используется личный ключ sk_A:

$$m = v \oplus h_2(e(sk_A, U)).$$

3.2. IBE-системы с процедурой отзыва ключей

Наиболее удобным способом обновления ключей является принудительная периодическая замена ключей путём привязки их к определённому периоду времени, например к одной неделе.

Центр KGC, выступающий в данном случае в роли центра управления ключами, должен регулярно производить обновление ключей всех пользователей. При невозможности непосредственных контактов для этого необходимо организовать рассылку зашифрованных ключей. Однако такой способ оказывается неудобным для систем с большим числом пользователей, поскольку центру KGC в этом случае приходится регулярно выполнять массовую рассылку зашифрованной ключевой информации.

В работе [6] предложена оригинальная схема, позволяющая избежать массовой рассылки, в которой пользователи обращаются к общедоступному серверу центра KGC только в случае необходимости. Для этого использована конструкция, построенная на основе нечёткой системы шифрования Fuzzy IBE и структуры бинарного дерева [38].

Преимуществом систем шифрования на основе конструкции Fuzzy IBE является то, что шифртексты могут храниться на общедоступном открытом сервере. В конструкции Fuzzy IBE из [56], описанной в п. 3.1, пользовательские ключи и ключи, использованные для зашифрования текстов, связаны с наборами описательных атрибутов. Ключ пользователя может расшифровать тот или иной зашифрованный текст только в том случае, если у ключа зашифрования и ключа пользователя совпадает определённое количество атрибутов (так называемая «устойчивость к ошибкам»). Количество атрибутов, используемых для шифрования, и степень устойчивости к ошибкам определяются заранее.

Для защищённости от сговора требуется, чтобы разные пользователи, объединив свои атрибуты вместе, не смогли расшифровать зашифрованный текст, который ни

один из них не смог расшифровать по отдельности. Чтобы предотвратить сговоры, алгоритм генерации ключей Fuzzy IBE для каждого пользователя генерирует случайный полином, степень которого на единицу меньше, чем степень устойчивости к ошибкам, для каждого пользователя. Этот многочлен используется для вычисления ключей, соответствующих набору атрибутов. Поскольку все ключи вычисляются на разных полиномах, они не могут быть объединены каким-либо значимым образом.

В схеме IBE сообщения шифруются по двум атрибутам: идентификационной информации получателя и периоду времени. Ключ расшифрования также вычисляется для идентификационной информации атрибутов и времени с помощью полинома первой степени, что означает, что оба атрибута ключа расшифрования должны совпадать с атрибутами зашифрованного текста.

Ключ расшифрования каждого пользователя, соответствующий идентификатору и времени, разделён на два компонента: личный ключ и обновление ключа. Чтобы иметь возможность расшифровать зашифрованный текст, пользователю требуется как личный ключ, так и обновление ключа. Личный ключ выдаётся каждому пользователю центром KGC так же, как и обычные личные ключи в IBE. Обновление ключа публикуется центром KGC и является общедоступным для всех пользователей.

Таким образом, когда центру управления ключами KGC необходимо отозвать ключ пользователя, он может просто прекратить публикацию обновлений ключей для этого пользователя.

Чтобы избежать необходимости вычисления обновлений ключей для каждого пользователя отдельно, в [6] используется структура двоичного симметричного дерева высотой h, в котором каждому пользователю соответствует уникальный конечный узел — лист дерева. Каждому узлу дерева присвоен случайный многочлен.

Каждый пользователь получает ключи, соответствующие его идентификационной информации и вычисленные по полиномам всех узлов на пути от листа, соответствующего этому пользователю, к корню дерева. Чтобы иметь возможность расшифровать текст, зашифрованный в период времени t, пользователю достаточно получить обновления ключа, соответствующие t, для всех полиномов вершин на этом пути. Таким образом, когда ни один пользователь не отозван, центру ключей достаточно опубликовать обновление ключа, вычисленное на полиноме корня дерева. Когда отзывается подмножество пользователей, центр сначала находит минимальный набор вершин в дереве, который содержит предка (или сам узел) среди всех листьев, соответствующих неотозванным пользователям. Затем центр публикует обновления ключей по полиномам для вершин из этого набора.

Начальная установка.

Пусть $\mathbb{G}_1$ — группа простого порядка p с образующим P и операцией билинейного спаривания. Определим функцию

$$F_{P,J,H_1,\ldots,H_J}(x) = [x^2]P + \sum_{i=1}^{J}[\delta_{i,J}(x)]H_i,$$

где $\delta_{i,J}(x)$ определяется равенством (2).

Пусть имеется бинарное древо T. Каждой вершине соответствует некоторая строка, определяемая путём $\mathrm{Path}(v)$ от v к корню root дерева T. Если вершина v не является листом дерева, то обозначим через v_l и v_r соответственно её левого и правого сына.

Обозначим через $rl = \{(v_i, t_i)\}$ список вершин, ключи которых отозваны. В этом списке для каждой вершины v_i указывается время t_i, когда произведён отзыв ключа.

Требуется расшифровать текст в период времени t так, чтобы ключи расшифрования не зависели от вершин, ключи которых были отозваны до этого момента времени.

Определим функцию `KUNodes`, которая вычисляет минимальное множество вершин, для которого требуется обновить ключи так, чтобы пользователи, ключи которых не отозваны до момента времени t, могли расшифровать шифртекст. В качестве входов функции `KUNodes` выступают бинарное дерево T, время t и список rl вершин, ключи которых отозваны. В этом списке для каждой вершины указывается время, когда произошел отзыв ключа. Выходом является минимальное множество вершин дерева T, для которого ни одна из вершин из списка rl, у которой время отзыва не превосходит t, не имеет предшественников в этом множестве и все остальные листья имеют в этом множестве в точности одного предшественника.

Алгоритм вычисления функции `KUNodes` состоит в следующем. Сначала помечаются как отзываемые все предшественники отозванных до момента времени t вершин, а затем и все дети отзываемых вершин. На рис. 2 показан пример работы алгоритма вычисления функции `KUNodes` при отзыве ключей пользователя, соответствующего вершине u_3. Символом $\otimes$ помечены вершины, ключи которых отзываются, а символом $\checkmark$ — вершины, ключи которых подлежат обновлению.

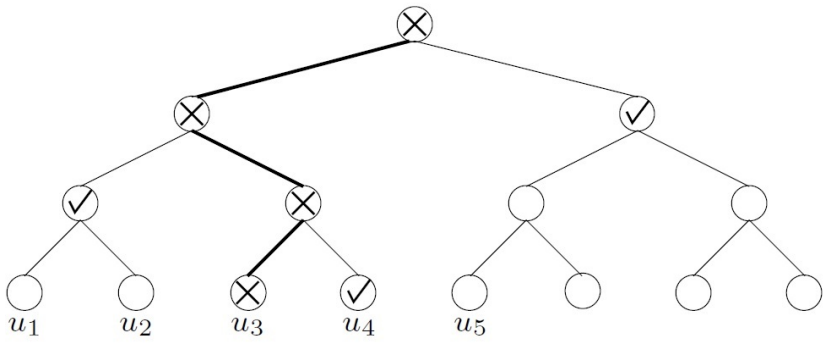

Рис. 2. Результат работы функции `KUNodes` при отзыве ключей пользователя u_3 [6]

Приведём описание алгоритма отзыва. Помимо дополнительных выходов rl и st, оно в точности повторяет процедуру из Fuzzy IBE.

Полагаем $rl = \varnothing$, и пусть T — бинарное дерево с n листьями. Пусть для примера $\mathbb{G}_1 = \langle P \rangle$, $|\mathbb{G}| = p$, $J \in \{1, 2, 3\}$, $a \in_R \mathbb{Z}_p$, $P_1 = [a]P$, $P_2, H_1, H_2, H_3 \in_R \mathbb{G}_1$.

Генерация закрытого ключа $\mathtt{SK}(pk, mk, \omega, st)$:

1) положить $pk = (P, P_1, P_2, H_1, H_2, H_3)$, $mk = a$, $st = T$;
2) взять непомеченную вершину-лист v из дерева T и приписать ω к этой вершине;
3) для всех $x \in Path(v)$:
   - если a_x не определено, то приписать вершине x значение $a_x \in_R \mathbb{Z}_p$;
   - $r_x \in_R \mathbb{Z}_p$;
   - $D_x = [a_x\omega + a]P_3 + [r_x]F_{P_2,J,H_1,H_2,H_3}(\omega)$;
   - $D'_x = [r_x]P$;
4) возвратить $sk_\omega = \{(x, D_x, D'_x)\}_{x \in Path(v)}$, st.

Заметим, что a_x фиксирует многочлен первой степени $q_x(y) = a_x y + a$ с условием $q_x(0) = a$, соответствующий вершине x. Алгоритм вычисляет ω-компоненты ключа расшифрования с помощью многочленов всех вершин на пути от листа, соответствующего ω, к корню дерева.

Генерация обновлённого ключа $\mathtt{KU}(pk, mk, t, rl, st)$:

1) положить $pk = (P, P_1, P_2, H_1, H_2, H_3)$, $mk = a$, $st = T$;
2) для всех $x \in \mathtt{KUNodes}(T, rl, t)$:
   - $r_x \in_R \mathbb{Z}_p$;
   - $E_x = [a_x t + a]P + [r_x]F_{P_2,J,H_1,H_2,H_3}(t)$;
   - $E'_x = [r_x]P$;
3) возвратить $ku_t = \{(x, E_x, E'_x) : x \in \mathtt{KUNodes}(T, rl, t)\}$.

Алгоритм сначала находит минимальное множество вершин, содержащее предшествующую вершину (или саму вершину) для всех неотзываемых вершин. Затем вычисляет t-компоненту ключа расшифрования с помощью многочленов всех вершин из этого множества.

Генерация ключа расшифрования (*Decryption Key Generation*) $DK(sk_\omega, ku_t)$:

1) положить $sk_\omega = \{(i, D_i, D'_i) : i \in I\}$, $ku_t = \{(j, E_j, E'_j) : j \in J\}$ для некоторых множеств вершин I, J;
2) для всех $(i, D_i, D'_i) \in sk_\omega, (j, E_j, E'_j) \in ku_t$:
   - если существуют $i \in I$, $j \in J$, такие, что $i = j$, то $dk_{\omega,t} = (D_i, E_j, D'_i, E'_j)$ (ключ создан);
   - в противном случае (т. е. sk_ω и ku_t не имеют ни одной общей вершины) $dk_{\omega,t} = \perp$ (символ $\perp$ указывает, что ключ отозван);
3) возвратить $dk_{\omega,t} = (D, E, D', E')$ (далее индексы i, j опускаем).

Алгоритм находит компоненты ключа sk_ω и ku_t, которые получены с использованием одинаковых многочленов.

Зашифрование $E(pk, \omega, t, m)$:

1) положить $pk = (P, P_1, P_2, H_1, H_2, H_3)$;
2) $z \in_R \mathbb{Z}_p$;
3) $c_1 = m \cdot e(P_1, P_2)^z$;
4) $C_2 = [z]P$;
5) $C_\omega = [z]F_{P_2,J,H_1,H_2,H_3}(\omega)$;
6) $C_t = [z]F_{P_2,J,H_1,H_2,H_3}(t)$;
7) возвратить $C = (\omega, t, C_\omega, C_t, c_1, C_2)$.

Алгоритм шифрования в точности повторяет IBE.

Расшифрование $D(dk_{\omega,t}, C)$:

1) положить $dk_{\omega,t} = (D, E, D', E')$, $C = (\omega, t, C_\omega, C_t, c_1, C_2)$;
2) $m = \left(\dfrac{e(D', C_\omega)}{e(D, C_2)}\right)^{t/(t-\omega)} \left(\dfrac{e(E', C_t)}{e(E, C_2)}\right)^{e(D,C_2)/(\omega-t)} \cdot c_1$;
3) возвратить m.

Алгоритм расшифрования такой же, как у Fuzzy IBE.

Отзыв (Revocation) $R(\omega, t, rl, st)$:

1) для любой вершины v, ассоциированной с идентификатором ω, добавить (v, t) к списку отозванных ключей rl;
2) возвратить rl.

Функция $\mathtt{KUNodes}$ должна вычисляться, только если изменяется содержание списка rl. Поэтому её значение должно быть сохранено и использоваться до следующего изменения списка rl.

Если число пользователей превышает ёмкость текущего дерева, то необходимо дополнить его непомеченным деревом такого же размера путём присоединения корневых

вершин обоих деревьев к новой корневой вершине. Тем самым число пользователей может быть увеличено в 2 раза. Каждый новый пользователь получает дополнительную компоненту личного ключа, определённую по старому или новому дереву, которая должна быть зашифрована с помощью соответствующей идентификационной информации с указанием времени и опубликована.

3.3. IDB-системы цифровой подписи

IDB-схема подписи Шамира на основе RSA

А. Шамир предложил схему цифровой IDB-подписи на основе RSA. Для подписи сообщения m пользователю A с открытым ключом $pk = h(\mathrm{ID}_A)$ и личным ключом $sk = h(\mathrm{ID}_A)^d \bmod n$ надо:

1) выбрать случайное число $r \in \mathbb{Z}_n$;
2) вычислить $t = r^e \bmod n$;
3) вычислить $f = h(t, m)$, где h — однонаправленная функция;
4) вычислить $s = sk \cdot r^f \bmod n$.

Значением подписи будет (s, t).

Для проверки подписи надо проверить выполнение равенства

$$s^e \stackrel{?}{=} pk \cdot t^{h(t,m)} \bmod n.$$

IDB-схема подписи GQ Гийу — Кискатера на основе RSA

L. C. Guillou и J.-J. Quisquater в 1999 г. [30] предложили модификацию схемы Шамира. Пусть центр KGC обладает закрытым мастер-ключом d и открытым ключом (n, e), $ed = 1 \,(\bmod n)$. Пользователь A получает в центре «тень» J_A своего идентификатора ID_A в качестве открытого ключа и RSA-подпись $S_A = J_A^{-d} \bmod n$ в качестве личного ключа. Ключ J_A формируется с учётом внесения избыточности в идентификационную информацию ID_A для того, чтобы не проходила (экзистенциональная) атака, позволяющая строить новые ключевые пары путём возведения в степень и умножения значений из известных противнику ключевых пар.

Алгоритм вычисления подписи $\sigma = (s, t)$ под сообщением m:

1) $r \in_R \mathbb{Z}_n$;
2) $u = r^e \bmod n$;
3) $t = J_A^m u^{e^k} \bmod n$, где k выбирается из условия $e^{k-1} \leqslant m \leqslant e^k$;
4) $s = rS_A^t$.

Для проверки подписи надо вычислить $u = J_A^t s^e$ и проверить выполнение равенства

$$t \stackrel{?}{=} J_A^m u^{e^k}.$$

Модифицированный вариант этой схемы [30] вошёл в международный стандарт ISO/IEC 14888-2: 1999.

IDB-схема подписи GQ1 на основе RSA из ISO/IEC 14888-2: 2008

Рассмотрим вариант GQ1 IDB-схемы цифровой подписи из последней версии этого стандарта. Пусть $n = p_1 \cdot \ldots \cdot p_f$ — произведение различных простых чисел; v — простое число (верификационная экспонента), $v < n$.

Формирование ключей владельца подписи.

Ключ проверки подписи формируется как элемент $G \in \mathbb{Z}_n$, каким-то образом сопоставленный идентификационной информации пользователя: $\mathrm{ID} \mapsto G$.

Ключ подписи $Q \in \mathbb{Z}_n$ можно вычислить двумя способами так, что полученные в результате числа G и Q удовлетворяют условию

$$G \cdot Q^v \bmod n = 1.$$

Способ 1 (с применением китайской теоремы об остатках (CRT, Chinese Remainder Theorem)):

1) для $i = 1, \ldots, f$:
   - найти число s_i как наименьшее положительное число, такое, что $v\,s_i - 1$ кратно $p_i - 1$;
   - вычислить $u_i = p_i - 1 - s_i$, $G_i = G \bmod p_i$, $Q_i = G_i^{u_i} \bmod p_i$;
2) найти $Q = \mathrm{CRT}(Q_1, \ldots, Q_f)$ с помощью китайской теоремы об остатках.

Способ 2 (без применения CRT):

1) найти число s как наименьшее положительное число, такое, что $v\,s - 1$ кратно наибольшему общему делителю $(p_1 - 1, \ldots, p_f - 1)$;
2) вычислить $u = (p_1 - 1, \ldots, p_f - 1) - s$;
3) вычислить $Q = G^u \bmod n$.

Алгоритм подписи:

1) выбрать случайные числа $r = (r_1, \ldots, r_t)$;
2) для $i = 1, \ldots, t$ вычислить $r_i^v \bmod n$ и сопоставить им битовую строку W;
3) вычислить $H = h(W||M)$, пусть R — начальный отрезок из $t(l(v) - 1)$ битов строки H, где $l(v)$ — битовая длина v;
4) для $i = 1, \ldots, t$ вычислить $r_i \cdot Q^{R_i} \bmod n$, сопоставить им битовую строку S.

Подписью является пара (R, S).

Алгоритм проверки подписи:

1) разделить битовые строки R и S на компоненты и выделить соответствующие числа r_i и s_i, $i = 1, \ldots, t$;
2) для $i = 1, \ldots, t$ вычислить $S_i^v \cdot G^{R_i} \bmod n$, сопоставить им битовую строку W^*;
3) вычислить $H^* = h(W^*||M)$, пусть R^* — начальный отрезок из $t(l(v) - 1)$ битов строки H^*;
4) проверить равенство $R^* \stackrel{?}{=} R$.

IBS-1-схема подписи на основе ECDLOG из ISO/IEC FDIS 14888-3: 2018

IDB-схема подписи IBS-1 строится на основе группы точек эллиптической кривой и операции билинейного спаривания. Этот механизм основан на алгоритме, разработанном в [35]. Пусть $\mathbb{G}_1 = \langle P \rangle$ — циклическая подгруппа порядка q группы точек эллиптической кривой E над полем $\mathrm{GF}(p^m)$.

Мастер-ключом центра KGC является ключевая пара (U, V), где открытый ключ U — случайное число в интервале $0 < U < q$, а закрытый ключ вычисляется по формуле $V = [U]P$.

Ключ формирования и проверки подписи пользователя — это ключевая пара (X, Y), где $Y = h_1(\mathrm{ID})$ — ключ проверки подписи, полученный из идентификационных данных владельца ID с помощью хеш-функции h_1, а X — личный ключ подписи, вычисляемый центром KGC: $X = [U]Y$.

Конкретные параметры схемы подписи — $\mathbb{G}_1, \mathbb{G}_2, P, q, e(,), h_1$ и h_2 — определены в [70]. Здесь $e(,)$ обозначает операцию билинейного спаривания $e : \mathbb{G}_1^2 \to \mathbb{G}_2$, которая предполагает наличие в мультипликативной подгруппе поля, являющегося некоторым расширением поля $\mathrm{GF}(p^m)$, циклической подгруппы $\mathbb{G}_2$ порядка q.

Алгоритм формирования подписи:

1) выбрать случайное целое k, $0 < k < q$, и сохранить его в секрете;
2) вычислить $\Pi = e(X, P)^k$.
   З а м е ч а н и е. Элемент Π принадлежит расширению поля $\mathrm{GF}(p^m)$ степени 4 для характеристики $p = 2$, степени 6 для характеристики $p = 3$ и степени 2 для характеристики $p > 3$;
3) при $p > 3$ вычислить
$$R = h_2(M \| \Pi_a \| \Pi_b) \bmod q,$$
   где $\Pi = (\Pi_a, \Pi_b) \in \mathrm{GF}(p^{2m})$. Если $R = 0$, то перейти к п. 1.
   Для полей более высокой степени расширения следует рассматривать больше компонент в строке Π. Например, для степени расширения 4: $\Pi = (\Pi_a, \Pi_b, \Pi_c, \Pi_d)$;
4) вычисление второй части подписи:
$$S = [k - R]X.$$

Подписью является $\Sigma = (R, S)$.

Алгоритм проверки подписи:

1) проверить $S \in \mathbb{G}_1$; если нет, то подпись неверна;
2) вычислить $\Pi' = e(S, P) * e(Y, V)^R$.
   З а м е ч а н и е . Значение спаривания $e(Y, V)$ может быть вычислено заранее;
3) вычислить
$$R' = h_2(M \| \Pi'_a \| \Pi'_b) \bmod q;$$
4) проверить свидетельство $R' \stackrel{?}{=} R$. Если да, то подпись верна, иначе неверна.

IDB-схема подписи BLMQ

Эта схема, основанная на операции билинейного спаривания и описанная в работе П. Баретто, Б. Либерта, Н. МакКулаха и Дж. Кискатера (P. Barreto, B. Libert, N. McCullagh и J. Quisquater) [5] в 2005 г., была опубликована, а затем предложена ими для включения в стандарт IEEE P1363.3 [4].

Стойкость схемы основана на трудности проблемы k-обращения Диффи — Хеллмана k-DHI (k-Diffie-Hellman Inversion) для групп $(\mathbb{G}_1, \mathbb{G}_2)$: для заданного $(k+2)$-набора $(P, Q, aQ, a^2Q, \ldots, a^kQ)$ найти $a^{-1}P$, где $P \in \mathbb{G}_1$; $Q \in \mathbb{G}_2$; $a \in_R \mathbb{Z}_q^*$; q — порядок этих групп.

Пусть $h_1, h_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ — хеш-функции, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ — операция билинейного спаривания. Открытый ключ центра: $P, Q, sQ, g = e(P, Q)$. Закрытый ключ центра: $s \in \mathbb{Z}_q$. Пользователь A получает в центре личный ключ

$$S_{\mathrm{ID}_A} = (h_1(\mathrm{ID}_A) + s)^{-1} P.$$

Подпись к сообщению m — это (H, S), где $H = h_2(m, r)$; $r = g^x$; $S = (x + H)S_{\mathrm{ID}_A}$; $x \in_R \mathbb{Z}_q^*$. Проверка подписи (H, S) заключается в проверке равенства

$$H \stackrel{?}{=} h_2(m, e(S, h_1(\mathrm{ID}_A)Q + sQ)g^{-h}).$$

IDB-схема подписи Уотерса

Схема IDB-шифрования, основанная на симметричном спаривании, предложенная в работе B. Waters в 2005 г. [66] и рассмотренная в п. 3.1, может быть преобразована в схему цифровой подписи следующим образом.

Пусть имеется операция билинейного спаривания $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, $\mathbb{G}_1 = \langle P_0 \rangle$. Закрытый ключ центра — случайный элемент $x \in_R \mathbb{Z}_p^*$.

Центр KGC вырабатывает случайные $P_2, U', U_1, \ldots, U_n \in_R \mathbb{G}_1$ и выдаёт пользователю личный ключ $[x]P_2$.

Для формирования подписи сообщение $m \in \{0,1\}^*$ сначала сжимается криптографической хеш-функцией $h\colon \{0,1\}^* \to \{0,1\}^n$: $h(m) = (m_1, \ldots, m_n)$, затем пользователь вырабатывает случайный элемент $r \in \mathbb{Z}_p$ и с помощью полученного в центре KGC личного ключа вычисляет

$$\mathrm{Sig}(m) = ([x]P_2 + [r]\left(U' + \sum_{i:m_i=1} U_i\right),\ [r]P_0).$$

Для проверки подписи $\mathrm{Sig}(m) = (\sigma_1, \sigma_2)$ к сообщению m с помощью открытого ключа $P_2, U', U_1, \ldots, U_n$ надо проверить равенство

$$e(\sigma_1, P_0)/e\left(\sigma_2, U' + \sum_{i:m_i=1} U_i\right) = e(P_0, P_2).$$

Описание других схем подписи на основе операции билинейного спаривания в группе точек эллиптической кривой, например предложенные Патерсоном (Paterson) и др., можно найти в обзоре [27].

IDB-система подписи с недоверенным центром KGC

В работе [18] Ч. Чен, Ф. Занг и К. Ким (X. Chen, F. Zhang и K. Kim) предложили вариант схемы подписи, позволяющий пользователю доказывать с нулевым разглашением арбитру, что центр KGC совершил атаку по (экзистенциональной) подмене подписи.

Пусть имеется операция билинейного спаривания $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q$, $\mathbb{G}_1 = \langle P \rangle$, две хеш-функции $h_1 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{G}_1$ и $h_2 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_q$.

Пользователь A выбирает $r \in_R \mathbb{Z}_q^*$ в качестве своего долговременного секрета и отправляет центру значение rP в качестве открытого ключа. Центр вычисляет $Q_A = h_1(\mathrm{ID}_A \| t, rP)$ и $S_A = sQ_A$, где t — срок действия, а затем передаёт их пользователю. Открытым ключом пользователя будет ID_A, а личным ключом — (S_A, s).

Алгоритм формирования подписи для сообщения m:

1) $a \in_R \mathbb{Z}_q$, $U = aQ_A$;
2) $V = rh_2(m, U)$;
3) $h = h_2(m, U + V)$;
4) $W = (a + h)S_A$.

Подписью является набор $\sigma = (U, V, W, t, rP) \in \mathbb{G}_1^3 \times \{0,1\}^* \times \mathbb{G}_1$.

Для проверки подписи надо вычислить $Q_A = h_1(\mathrm{ID}_A \| t, rP)$, $h_1(m, U)$ и $h = h_2(m, U + V)$, а затем проверить выполнимость равенств

$$e(W, P) = e(U + hQ_A, Q), \quad e(V, P) = e(h_1(m, U), rP).$$

Если центр выполнит атаку по подмене подписи для сообщения m следующим образом: вычислит $r' \in_R \mathbb{Z}_q$ и $Q'_A = h_2(\mathrm{ID}_A \| t, r'P)$, а затем сформирует подпись в соответствии с приведённым алгоритмом

$$\sigma' = (U', V', W', t, r'P),$$

то проверка подписи будет успешной, однако пользователь сможет доказать арбитру с нулевым разглашением на основе знания своего личного ключа, что это не его подпись.

IDB-системы одновременного вычисления цифровой подписи и шифрования (IBSigncryption)

Протокол Зенга. В 1997 г. Zheng в работе [68] предложил одну из первых схем одновременного вычисления цифровой подписи и шифрования, более быстрого, чем их последовательное вычисление.

Общими параметрами являются:

p — большое простое число;

q — делитель $(p-1)$;

$g \in_R \mathbb{Z}_p$, $g^q = 1 \bmod p$;

h — однонаправленная хеш-функция (с не менее чем 128-битовым выходом);

$\mathtt{KH}_k()$ — хеш-функция, зависящая от ключа;

(E, D) — алгоритмы симметричного зашифрования и расшифрования.

Пусть также:

— личным ключом A является $x_a \in_R \{1, \ldots, q-1\}$, открытым — $y_a = g^{x_a} \bmod p$;

— личным ключом B является $x_b \in_R \{1, \ldots, q-1\}$, открытым — $y_b = g^{x_b} \bmod p$.

Протокол передачи сообщения m имеет вид

$$A \to B: \ (c, r, s).$$

Алгоритм формирования подписи и зашифрования:

1) сторона A выбирает $x \in_R \{1, \ldots, q-1\}$, вычисляет $k = h(y_b^x \bmod p)$ и выделяет из k ключи $k = k_1 \| k_2$;
2) вычисляет $c = E_{k_1}(m)$, $r = \mathtt{KH}_{k_2}(m)$, $s = x/(r + x_a) \bmod q$;
3) отправляет стороне B подписанный шифртекст (c, r, s).

Алгоритм расшифрования и проверки подписи:

1) сторона B восстанавливает k из c, r, s, g, p, x_a и x_b:

$$k = h((y_a \cdot g^r)^{s x_b} \bmod p)$$

   и выделяет из k ключи k_1 и k_2;
2) вычисляет $m = D_{k_1}(c)$;
3) признаёт m подлинным сообщением от A в том и только в том случае, когда $r = \mathtt{KH}_{k_2}(m)$.

Этот протокол не защищён от чтения назад, так как если противник восстановит долговременный ключ x_a, то в силу равенства

$$h((y_a \cdot g^r)^{s x_b} \bmod p) = h((y_b^{x_a + r})^s \bmod p)$$

он сможет вычислить ключ $k = h((y_b^{x_a+r})^s \bmod p)$.

Протокол Нэйла — Ридди. На базе данной схемы D. Nalla и K. C. Reddy в работе [47] предложили IDB-схему одновременного вычисления цифровой подписи и шифрования, основанную на операции билинейного спаривания. Пусть $\mathbb{G}_1$ — подгруппа группы точек эллиптической кривой порядка q, для которой определена операция билинейного спаривания Вейля $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$; $\mathbb{G}_2$ — мультипликативная группа того же порядка q. Предположим также, что имеются хеш-функция $h'\colon \{0,1\}^* \to \mathbb{Z}_q^*$, функция вычисления ключа шифрования $h''\colon \mathbb{Z}_q^* \to \{0,1\}^*$ и псевдослучайная функция $h_1\colon \mathbb{G}_2 \to \{0,1\}^*$.

Стороны A и B получают в центре KGC ключевые пары (Q_A, S_A) и (Q_B, S_B), сформированные на основе своих идентификаторов, $S_A = [s]Q_A$, $S_B = [s]Q_B$, где $s \in_R \mathbb{Z}_q^*$ — закрытый мастер-ключ центра.

Для отправления стороне B подписанного сообщения $m \in \{0,1\}^*$ сторона A использует открытые ключи (Q_A, Q_B) и свой личный ключ S_A.

Алгоритм формирования подписи и зашифрования:

1) сторона A выбирает случайный элемент $a \in_R \mathbb{Z}_q^*$;
2) вычисляет

$$R = [a]S_A, \quad d = h'(R||h_1(\hat{e}(Q_B, S_A))||m), \quad S = [ad]Q_A,$$
$$k_A = h''(\hat{e}(Q_B, S_A)^{ad}), \quad c = k_A \oplus m;$$

3) отправляет (R, S, c) стороне B.

Алгоритм расшифрования и проверки подписи.

Для проверки полученного сообщения сторона B, используя полученные значения (R, S, c, Q_A, Q_B) и свой личный ключ S_B:

1) вычисляет $k_B = h''(\hat{e}(S_B, S))$ и $m = k_B \oplus c$;
2) вычисляет $d' = h'(R||h_1(\hat{e}(S_B, Q_A))||m)$ и принимает m, только если выполнено равенство $\hat{e}(S_B, S) = \hat{e}(Q_B, R)^{d'}$. В противном случае B прерывает протокол.

Данный протокол защищён от чтения назад и является вычислительно более эффективным.

IDB-системы одновременного вычисления цифровой подписи и шифрования с сокрытием идентификаторов

Рассмотрим протокол IBHigncryption (от Id-Based Higncryption). Термин «higncryption» означает identity-hiding signcryption, т. е. одновременное вычисление цифровой подписи и шифрования с сокрытием идентификаторов.

Пусть имеется операция симметричного билинейного спаривания $e\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ для группы $\mathbb{G}_1 = \langle P \rangle$ порядка q и $h : \{0,1\}^* \to \mathbb{G}_1$ — криптографическая хеш-функция. Центр выбирает закрытый ключ $msk = s \in_R \mathbb{Z}_q^*$. Открытыми параметрами являются $(q, \mathbb{G}_1, \mathbb{G}_T, e, P, h)$.

Пользователь A применяет в качестве открытого ключа идентификатор $pk_A = \mathrm{ID}_A$ и получает в центре $sk_A = [s]h(\mathrm{ID}_A)$ в качестве личного ключа.

Пусть (k, E, D) — некоторая схема аутентифицированного шифрования с ассоциированными данными. Ключ k вырабатывается с использованием функции выработки производного ключа $\mathtt{KDF}\colon \mathbb{G}_T \times \{0,1\}^* \to K$.

Для зашифрования и подписи сообщения $n \in \{0,1\}^*$ с сокрытием идентификаторов отправитель A:

1) выбирает случайный $x \in_R \mathbb{Z}_q^*$ и вычисляет $X = [x]h(\mathrm{ID}_A) \in \mathbb{G}_1$;

2) вычисляет предварительный секрет $PS = e(sk_A, h(\mathrm{ID}_B))^x \in \mathbb{G}_T$;
3) вычисляет ключ для AE шифрования $k = \mathtt{KDF}(PS, X \| \mathrm{ID}_A)$;
4) вычисляет $c_{AE} = E_k(H, \mathrm{ID}_A \| n \| x)$ с ассоциированными данными $H \in \{0,1\}^*$;
5) отправляет получателю B сообщение $C = (H, X, c_{AE})$.

Для расшифрования полученного сообщения $C = (H, X, c_{AE})$ и проверки подписи получатель B:

1) вычисляет $PS = e(X, sk_B) \in \mathbb{G}_T$ и ключ $k = \mathtt{KDF}(PS, X \| \mathrm{ID}_B)$;
2) расшифровывает $D_k(H, c_{AE})$ с проверкой целостности;
3) получает ID_A, n, x и проверяет равенство $X = [x]h(\mathrm{ID}_A)$. Если всё правильно, то принимает сообщение m, в противном случае прерывает протокол.

Подписи на основе сертификатов

Цифровые подписи на основе сертификатов (Certificate-Based Signature, CBS) строятся аналогично CBE-системам. В них сертификат, построенный на основе открытого ключа и идентификационной информации пользователя, используется как составная часть ключа подписи, составленного непосредственно из сгенерированного пользователем личного ключа и полученного сертификата.

Впервые такая система предложена в [37], однако позднее в работе [41] найдена атака и построен исправленный вариант. В [3] предложена схема анонимной циклической подписи на основе сертификатов. Все указанные схемы построены с использованием операции билинейного спаривания.

Рассмотрим предложенную в работе [43] CBC-схему цифровой подписи, не использующую операцию билинейного спаривания. Пусть $\mathbb{G}$ — мультипликативная группа порядка q. Центр KGC выбирает случайный образующий элемент $g \in_R \mathbb{G}$ и случайное число $x \in_R \mathbb{Z}_q^*$, являющееся его мастер-ключом. Пусть $X = g^x$ и $h : \{0,1\}^* \to \mathbb{Z}_q^*$ — криптографическая хеш-функция. Параметрами схемы являются $(\mathbb{G}, q, g, X, h)$.

Пользователь A выбирает $u \in_R \mathbb{Z}_q^*$ в качестве своего личного ключа sk_A и вычисляет открытый ключ $pk_A = (g^u, X^u, \pi_u)$, где π_u — неинтерактивное доказательство знания значения дискретного логарифма $u = \log_g U_1 = \log_X U_2$ для элементов $U_1 = g^u$ и $U_2 = X^u$ (например, на основе протокола аутентификации Шнорра).

Идентификационной информацией пользователя служит значение $\tilde{H} = h(pk_A, \mathrm{ID}_A)$, где pk_A и ID_A — открытый ключ и идентификатор пользователя A. Сертификационный центр CA выбирает случайно $r \in_R \mathbb{Z}_q^*$ и вычисляет сертификат (R, s) пользователя A (фактически сертификат (R, s) является цифровой подписью Эль-Гамаля под сообщением $\tilde{H}$, связывающим значения pk_A и ID_A):

$$R = g^r, \qquad s = r^{-1}(\tilde{H} - xR) \bmod q.$$

Корректность сертификата проверяется равенством

$$R^s X^R = g^{\tilde{H}}.$$

Для подписания сообщения $m \in \{0,1\}^*$ пользователь использует сертификат (R, s) и личный ключ u, а также случайный элемент $y \in_R \mathbb{Z}_q^*$ и вычисляет значение подписи $\sigma = (Y, H, z)$:

$$Y = R^{-y}, \quad H = h(Y, R, m), \quad z = (y + Hsu) \bmod q.$$

Проверка подписи осуществляется следующим образом. Сначала проверяется, что π_u действительно является неинтерактивным доказательством знания пользователем с идентификатором ID_A и открытым ключом pk_A значения ключа u. Если оно

прошло проверку, то вычисляются $H = h(Y, R, m)$ и $\tilde{H} = h(pk_A, ID_A)$, а затем проверяется равенство

$$(g^u)^{\tilde{H}} \stackrel{?}{=} R^z Y (X^u)^{HR}.$$

Если оно выполнено, то подпись признаётся истинной.

3.4. IDB-системы аутентификации сторон

Протоколы аутентификации сторон обычно строятся на основе протоколов цифровой подписи, использующих технику доказательства знания без разглашения секретов. Рассмотрим пример такого протокола.

IDB-система аутентификации сторон GQ

В 1988 г. Л. Гийу и Ж. Кискатер [29] предложили протокол аутентификации сторон на основе схемы RSA, вариант GQ1 которого [31] позже вошёл в международный стандарт ISO/IEC 9798-5:2009, а соответствующий ему IDB-протокол цифровой подписи — в рассмотренный выше стандарт ISO/IEC 14888-2:2008.

Пусть $n = pq$ и числа e, d удовлетворяют условию $ed = 1 \pmod{\varphi(n)}$. Число d известно только центру, выдающему ключи. Пусть h — хеш-функция, известная всем участникам. Каждый участник A получает в центре личный ключ $u = h(ID_A)^{-d} \in \mathbb{Z}_n$ и открытый ключ $v = h(\mathrm{ID}_A)$. Тем самым выполняется равенство $v = (u^{-1})^e \bmod n$, и мы можем использовать ранее рассмотренный протокол:

$$\begin{aligned} A \to B: &\quad A, \gamma = r^e \bmod n, \\ A \leftarrow B: &\quad x, \\ A \to B: &\quad y = ru^x \bmod n. \end{aligned}$$

Для проверки правильности B использует равенство

$$\gamma = v^x y^b \bmod n.$$

3.5. IDB-протоколы выработки общего ключа

IBKE-протокол Окамото на основе RSA

Первый IDB-протокол выработки общего ключа (Identity-Based Key Exchange) на основе RSA предложен Е. Окамото (E. Okamoto) в 1987 г. [48]. KGC имеет открытый ключ (e, n), а число d, такое, что $ed = 1 \pmod{\varphi(n)}$, KGC хранит в секрете. Пусть g_0 — образующий элемент мультипликативной группы $\mathbb{Z}_n^*$.

Пользователь $X \in \{A, B\}$ имеет открытый ключ ID_X и получает в центре KGC личный ключ s_X, удовлетворяющий условию $(s_X)^e \mathrm{ID}_X = 1 \bmod n$. Значение s_X центр KGC вычисляет по формуле $s_X = (1/\mathrm{ID}_X)^d \bmod n$, где ID_X — идентификационные данные пользователя X.

Протокол:

$$\begin{aligned} A: &\quad r_A \in_R \mathbb{Z}_n^*, \\ A \to B: &\quad t_A = s_A g^{r_A} \bmod n, \\ B: &\quad r_B \in_R \mathbb{Z}_n^*, \\ A \leftarrow B: &\quad t_B = s_B g^{r_B} \bmod n. \end{aligned}$$

Теперь A и B вычисляют общий ключ $k = g^{er_A r_B}$ соответственно по формулам

$$k_A = (t_B^e \mathrm{ID}_B)^{r_A}, \qquad k_B = (t_A^e \mathrm{ID}_A)^{r_B}.$$

IBAKE-протокол с обеспечением частичной аутентификации ключа

Протокол, предложенный Е. Окамото и К. Танака (E. Okamoto, K. Tanaka) в 1989 г. [50], является вариантом предыдущего. В нём используется хеш-функция $h : \{0,1\}^* \to \mathbb{Z}_{n-1}^*$ для обеспечения частичной аутентификации ключа. Обозначим через T_A, T_B метки времени.

Протокол:

$$\begin{array}{rl}
A: & r_A \in_R \mathbb{Z}_n^*, u_A = g^{er_A} \bmod n,\\
A: & c_A = h(u_A, \mathrm{ID}_A, \mathrm{ID}_B, T_A),\\
A \to B: & u_A, v_A = s_A g^{c_A r_A} \bmod n,\\
B: & c_A = h(u_A, \mathrm{ID}_A, \mathrm{ID}_B, T_A),\\
B: & \text{проверяет } \mathrm{ID}_A \stackrel{?}{=} u_A^{c_A}/v_A^e,\\
B: & r_B \in_R \mathbb{Z}_n^*, u_B = h^{er_B} \bmod n,\\
B: & c_B = h(u_B, \mathrm{ID}_B, \mathrm{ID}_A, T_B),\\
A \leftarrow B: & u_B, v_B = s_B g^{c_B r_B} \bmod n,\\
A: & c_B = h(u_B, \mathrm{ID}_B, \mathrm{ID}_A, T_B),\\
A: & \text{проверяет } \mathrm{ID}_B \stackrel{?}{=} u_B^{c_B}/v_B^e.
\end{array}$$

Теперь A и B вычисляют общий ключ $k = g^{er_A r_B}$ соответственно по формулам

$$k_A = u_B^{r_A}, \qquad k_B = u_A^{r_B}.$$

Два предыдущих протокола не защищены от нечестного центра KGC, так как он имеет возможность восстановить личные ключи всех пользователей, а также не обеспечивают защиты от чтения назад при компрометации мастер-ключа центра KGC. Поэтому необходимо обеспечить защиту пользователя от центра, предоставив ему возможность самому формировать личные ключи.

IBKE-протокол с защитой личного ключа пользователя от центра KGC

М. Гиро и Дж. Пале (M. Girault, J. Paillès) в [25] предложили модификацию протокола Окамото, в которой предусмотрена защита личного ключа пользователя от центра KGC. Пользователь A передаёт в центру KGC значение $g^{x_A} \bmod n$, где $sk_A = x_A \in \mathbb{Z}_n^*$ — выбранный им самостоятельно и сохраняемый в секрете личный ключ, и получает открытый ключ $pk_A = y_A = \mathrm{ID}_A^{-d} g^{-x_A} \bmod n$, корректность которого он может проверить из условия $(y_A)^e \mathrm{ID}_A = g^{-ex_A} \bmod n$ (заметим, что A может вычислить y_A самостоятельно по значению s_A из протокола Окамото).

Протокол:

$$\begin{array}{rl}
A: & r_A \in_R \mathbb{Z}_n^*,\\
A \to B: & t_A = y_A g^{x_A - r_A} \bmod n,\\
B: & r_B \in_R \mathbb{Z}_n^*,\\
A \leftarrow B: & t_B = y_B g^{x_B - r_B} \bmod n.
\end{array}$$

Теперь A и B вычисляют общий ключ $k = g^{-er_A r_B}$ аналогично протоколу Окамото по формулам

$$k_A = (t_B^e \mathrm{ID}_B)^{r_A}, \qquad k_B = (t_A^e \mathrm{ID}_A)^{r_B}.$$

В результате выполнения протокола получается ключ, отличающийся знаком от того, который вычисляется в протоколе Окамото. Однако так как ключ не зависит от личных ключей участников, данный протокол не имеет преимуществ по сравнению с протоколом Окамото.

IBKE-протокол с самосертифицируемыми открытыми ключами

В 1991 г. М. Гиролт (M. Girault) предложил следующий протокол [26]. Пользователь A передаёт центру KGC значение $g^{x_A} \bmod n$, где $sk_A = x_A \in \mathbb{Z}_n^*$ — выбранный им самостоятельно и сохраняемый в секрете личный ключ, и получает открытый ключ $pk_A = y_A$, удовлетворяющий условию $y_A = (g^{x_A} - \mathrm{ID}_A)^d \bmod n$.

Центр KGC также не может узнать значение x_A, но теперь с помощью проверки равенства $g^{x_A} = y_A^e + \mathrm{ID}_A \pmod n$ каждый может убедиться в правильности открытого ключа.

Протокол:

$$\begin{array}{rl} A: & r_A \in_R \mathbb{Z}_n^*, \\ A \to B: & t_A = g^{r_A} \bmod n, \\ B: & r_B \in_R \mathbb{Z}_n^*, \\ A \leftarrow B: & t_B = g^{r_B} \bmod n. \end{array}$$

Теперь A и B вычисляют общий ключ $k = g^{r_A x_B + r_B x_B}$ аналогично протоколу MTI/A0 [44] по формулам

$$k_A = t_B^{x_A}(y_B^e + \mathrm{ID}_B)^{r_A}, \qquad k_B = t_A^e(y_A^e + \mathrm{ID}_A)^{r_B}.$$

IBKE-протокол выработки общего ключа на основе цифровой подписи

К. Гюнтер (C. G. Günther) в [32] предложил протокол выработки общего ключа на основе цифровой подписи, сформированной центром KGC. Закрытым и открытым ключами центра являются элементы $x_T \in \{1, \ldots, p-1\}$ и $y_T = g^{x_T} \in \mathbb{Z}_p^*$, где p — большое простое число; g — образующий элемент группы $\mathbb{Z}_p^*$.

Пользователь A получает в KGC цифровую подпись (u_A, v_A) по схеме Эль-Гамаля для своего идентификатора $\mathrm{ID}_A \in \mathbb{Z}_p^*$:

$$u_A = g^{k_A}, \quad v_A = (\mathrm{ID}_A - x_T u_A) k_A^{-1} \bmod (p-1),$$

где $k_A \in_R \mathbb{Z}_p^*$; $(k_A, p-1) = 1$. Проверка подписи проводится с помощью уравнения

$$u_A^{v_A} = g^{\mathrm{ID}_A} y_T^{-u_A}.$$

В роли открытого ключа пользователя A выступает его идентификатор ID_A и первая половина подписи u_A, а в роли личного ключа — вторая половина подписи $v_A = \log_{u_A}(g^{\mathrm{ID}_A} y_t^{-u_A})$. Для выработки общего ключа стороны A и B выбирают случайные элементы $r_A, r_B \in_R \mathbb{Z}_p^*$ и выполняют следующий протокол:

$$\begin{array}{rl} A \to B: & \mathrm{ID}_A, u_A, \\ A \leftarrow B: & \mathrm{ID}_B, u_B, \\ A \to B: & w_A = u_B^{r_A}, \\ A \leftarrow B: & w_B = u_A^{r_B}. \end{array}$$

Теперь A и B могут вычислить общий ключ соответственно по формулам

$$k_A = w_B^{v_A}(g^{\mathrm{ID}_B} y_T^{-u_B})^{r_A}, \quad k_B = w_A^{v_B}(g^{\mathrm{ID}_A} y_T^{-u_A})^{r_B}.$$

В результате формируется общий ключ $k = w_B^{v_A} w_A^{v_A} = g^{k_B r_A v_B + k_A v_A r_B}$.

Данный протокол также нестоек к атаке чтения назад, так как при компрометации долговременных ключей v_A, v_B имеется возможность определения разовых общих ключей по передаваемым сообщениям w_B, w_A:

$$k = w_B^{v_A} w_A^{v_B}.$$

В качестве улучшения этого протокола с одновременным сокращением числа передаваемых сообщений С. Саедниа (S. Saeednia) в 2000 г. [52] предложил следующую модификацию:

$$\begin{aligned} A \to B: &\quad \mathrm{ID}_A, u_A, t_A = g^{r_A}, \\ A \leftarrow B: &\quad \mathrm{ID}_B, u_B, t_B = g^{r_B}. \end{aligned}$$

Улучшение достигнуто за счёт изменения формулы для вычисления второй части подписи Эль-Гамаля

$$v_A = \mathrm{ID}_A k_A - x_T u_A \bmod (p-1)$$

и проверочного соотношения

$$g^{v_A} = u_A^{\mathrm{ID}_A} y_T^{u_A}.$$

Стороны A и B могут вычислить общий ключ соответственно по формулам

$$k_A = t_B^{v_B} (u_B^{\mathrm{ID}_B} y_S^{u_B})^{r_A}, \quad k_B = t_A^{v_B} (u_A^{\mathrm{ID}_A} y_S^{u_A})^{r_B}.$$

В результате получается $k = g^{v_B r_A + v_A r_B}$.

Протоколы выработки общего ключа на основе операции билинейного спаривания

IBKE-протокол Сакаи — Огиши — Казахара

Протокол типа SK предложен R. Sakai, K. Ohgishi и M. Kasahara в 2001 г. [54]. В нём используется операция билинейного спаривания с *разными* группами $\mathbb{G}_1$ и $\mathbb{G}_2$, для которых имеются хеш-функции $h_1\colon \{0,1\}^* \to \mathbb{G}_1$ и $h_2\colon \{0,1\}^* \to \mathbb{G}_2$.

Центр KGC обладает ключевой парой (s, Q_{pub}), $Q_{\mathrm{pub}} = [s]P$, $P \in \mathbb{G}_2$.

Протокол является неинтерактивным и не предполагает обмена сообщениями, аналогично статичному протоколу Диффи — Хеллмана.

Первый вариант протокола использует только одну хеш-функцию h_2 и предполагает наличие гомоморфизма $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$. Сторона A получает открытый ключ $Q_A = h_2(\mathrm{ID}_A)$ и личный ключ $S_A = [s]Q_A$.

Общий ключ $k = e(\psi(Q_A), Q_B)^s$ вычисляется сторонами по формулам

$$k = e(\psi(S_A), Q_B) = e(\psi(S_B), Q_A).$$

Второй вариант протокола не использует гомоморфизм. Сторона A получает открытый ключ $(Q_A, Q'_A) = (h_1(\mathrm{ID}_A), h_2(\mathrm{ID}_A))$ и личный ключ $(S_A, S'_A) = ([s]Q_A, [s]Q'_A)$.

Общий ключ $k = e(Q_A, Q'_B)^s \cdot e(Q_B, Q'_A)^s$ вычисляется сторонами по формулам

$$k_A = e(S_A, Q'_B) \cdot e(Q_B, S'_A), \qquad k_B = e(S_B, Q'_A) \cdot e(Q_A, S'_B).$$

IBKE-протокол Смарта

В 2002 г. Н. Смарт (N. P. Smart) [61] предложил первый интерактивный протокол выработки общего ключа на основе операции билинейного спаривания, аналогичный протоколу MTI/A0 [44]. Это протокол типа SOK, в котором центр обладает ключевой парой (s, Q_{pub}), $Q_{\text{pub}} = [s]P$, $s \in_R \mathbb{Z}_q^*$, $\langle P \rangle = \mathbb{G}$, $|\mathbb{G}| = q$, а участник A получает ключевую пару $(Q_A = h_1(\text{ID}_A), S_A = [s]Q_A)$.

Для выработки общего ключа стороны выбирают случайно $r_A, r_B \in_R \mathbb{Z}_q^*$ и выполняют следующий протокол:

$$\begin{array}{ll} A \to B: & T_A = [r_A]P, \\ A \leftarrow B: & T_B = [r_B]P. \end{array} \tag{3}$$

Теперь A и B могут вычислить общий ключ соответственно по формулам

$$k_A = e([r_A]Q_B, Q_{\text{pub}})\, e(S_A, T_B), \quad k_B = e([r_B]Q_A, Q_{\text{pub}})\, e(S_B, T_A).$$

В результате получается ключ

$$\begin{gathered} k = e([r_A]Q_B, Q_{\text{pub}})\, e([s]Q_A, [r_B]P) = e([r_A]Q_B, Q_{\text{pub}})\, e([r_B]Q_A, Q_{\text{pub}}) = \\ = e([r_A]Q_B + [r_B]Q_A, Q_{\text{pub}}). \end{gathered}$$

Данный протокол, подобно протоколу MTI, защищён от атаки «противник-в-середине», но он не стоек к атаке чтения назад, так как при компрометации долговременных ключей S_A и S_B пользователей противник имеет возможность вычислять действующие ключи для любого сеанса

$$k = e(S_A, T_B)e(S_B, T_A).$$

IBKE-протокол Скотта

В 2002 г. М. Скотт (M. Scott) в [59] предложил другой способ обмена сообщениями, зависящими от идентификаторов сторон, аналогичный протоколу MTI/C1:

$$\begin{array}{ll} A \to B: & p_A = e(S_A, Q_B)^{r_A}, \\ A \leftarrow B: & p_B = e(S_B, Q_A)^{r_B}. \end{array}$$

Теперь A и B могут вычислить общий ключ $k = e(Q_A, Q_B)^{sr_Ar_B}$ соответственно по формулам

$$k_A = p_B^{r_A}, \quad k_B = p_A^{r_B}.$$

IBKE-протокол Шима

В 2003 г. К. Шим (K. Shim) [62] предложил защищённый от чтения назад вариант протокола. Он также отличается только способом вычисления общего ключа $k = e(T_A + Q_A, T_B + Q_B)^s$ по формулам

$$k_A = e([r_A]Q_{\text{pub}} + S_A, T_B + Q_B), \quad k_B = e([r_B]Q_{\text{pub}} + S_B, T_A + Q_A).$$

Однако в [63] найдена атака «противник-в-середине» на этот протокол. Противник C выбирает случайные числа a' и b' и подменяет сообщения в протоколе (3):

$$\begin{array}{lll} A \to & C(B): & T_A = [r_A]P, \\ & C(A) \to B: & T'_A = [a']P - Q_A, \\ & C(A) \leftarrow B: & T_B = [r_B]P, \\ A \leftarrow & C(B): & T'_B = [b']P - Q_B. \end{array}$$

Теперь A и B вычисляют различные ключи по формулам

$$k_A = e([r_A]Q_{pub} + S_A, T'_B + Q_B) = e([r_A]Q_{\text{pub}} + S_A, [a']P) = e(P,P)^{r_A s b'} e(Q_A, P)^{s b'},$$
$$k_B = e([r_B]Q_{\text{pub}} + S_B, T'_A + Q_A) = e([r_B]Q_{\text{pub}} + S_B, [b']P) = e(P,P)^{a' s r_B} e(Q_A, P)^{s a'}.$$

При этом противник C может вычислить эти значения по формулам

$$k'_A = e(T_A, b'Q_{\text{pub}}) e(Q_A, b'Q_{\text{pub}}) = e(P,P)^{r_A s b'} e(Q_A, P)^{s b'} = k_A,$$
$$k'_B = e(T_B, a'Q_{\text{pub}}) e(Q_B, a'Q_{\text{pub}}) = e(P,P)^{a' s r_B} e(Q_A, P)^{s a'} = k_B.$$

IBKE-протокол Рюи — Юн — Юу

Защищённый от этих атак вариант протокола Смарта типа SOK предложили в 2004 г. Е. Рюи, Е. Юн и К. Юу (E. K. Ryu, E. J. Yoon и K. Y. Yoo) [51]. Он отличается только способом вычисления общего ключа $k = ([r_A r_B]P, e(Q_A, Q_B)^s)$ по формулам

$$k_A = ([r_B]T_A, e(S_A, Q_B)), \quad k_B = ([r_A]T_B, e(S_B, Q_A)).$$

В 2009 г. С. Ванг и др. [64] предложили новый способ вычисления общего ключа, основанный на применении хеш-функции

$$k = h(\text{ID}_A, \text{ID}_B, [r_A r_B]P, e(Q_A, Q_B)^s, T_A, T_B).$$

IBKE-протокол Ванга

В 2013 г. Ю. Ванг (Y. Wang) [65] предложил новый аутентифицированный протокол вида

$$\begin{array}{ll} A \to B: & T_A = [r_A]Q_A, \\ A \leftarrow B: & T_B = [r_B]Q_B \end{array}$$

типа SOK, но отличающийся способом выработки общего ключа

$$k = e(Q_A, Q_B)^{s(t_A + s_A)(t_B + s_B)},$$

где $s_A = h(T_A, T_B)$; $s_B = h(T_B, T_A)$; $h : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Пользователи вычисляют общий ключ по формулам

$$K_A = e((t_A + s_A)S_A, s_B Q_B + T_B)), \quad K_B = e((t_B + s_B)S_B, s_A Q_A + T_A)).$$

IBKE-протокол МакКалаха — Барето

В 2005 г. Н. МакКалах и П. Барето (N. McCullagh и P. S. L. M. Barreto) [45] предложили протокол выработки общего ключа на основе идентификаторов, имеющий аналогии с протоколом цифровой подписи BLMQ. Пусть P — точка эллиптической кривой над полем $\mathbb{Z}_p$, $\langle P\rangle = \mathbb{G}$, $h : \{0,1\}^* \to \mathbb{Z}_p^*$ — хеш-функция. Центр обладает закрытым ключом s и открытым ключом $[s]P$. Пользователь A получает в удостоверяющем центре открытый ключ $Q_A = [h(\mathrm{ID}_A)]P + [s]P = [h(\mathrm{ID}_A) + s]P$ и личный ключ $S_A = [(h(\mathrm{ID}_A) + s)^{-1}]P$. Пользователи A и B выбирают случайные элементы поля r_A и r_B соответственно и выполняют протокол

$$\begin{aligned} A \to B: &\quad N_A = [r_A]Q_B, \\ A \leftarrow B: &\quad N_B = [r_B]Q_A. \end{aligned}$$

Теперь A и B могут вычислить общий ключ $k = e(P,P)^{s(r_A+r_B)}$ по формулам

$$k_A = e(S_A, N_B)^{r_A}, \qquad k_B = e(S_B, N_A)^{r_B}.$$

IBKE-протоколы выработки общего ключа на основе эллиптической кривой без операции билинейного спаривания

IBKE-протокол Као — Коу — Ду

Протокол предложен X. Cao, W. Kou, X. Du. в 2010 г. [11]. Центр KGC генерирует для пользователя с идентификатором $\mathtt{ID}_A$ случайное число s_A, выступающее в роли случайного параметра для текущей ключевой пары, и вычисляет открытый ключ $Q_A = s_A P$ и личный ключ $\sigma_A = s_A + h_A s \bmod q$, где $h_A = h_1(\mathtt{ID}_A \| s_A)$, $h_1 : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_q$.

Для выработки общего ключа пользователи A и B выбирают случайные числа r_A и r_B и выполняют протокол

$$\begin{aligned} A \to B: &\quad T_A = [r_A]P, Q_A, \\ A \leftarrow B: &\quad T_B = [r_B]P, Q_B. \end{aligned}$$

Теперь A и B могут вычислить общий ключ $K = [r_A r_B]P$ по формулам

$$K_A = Q_A + [r_A](T_B + [h_B]Q_{\mathrm{pub}}), \qquad K_B = Q_B + [r_B](T_A + [h_A]Q_{\mathrm{pub}}).$$

IBKE-протокол Ислама — Бисваса

Протокол предложили H. Islam и G. P. Biswas в 2010 г. [36]. Он отличается от предыдущего видом пересылаемых сообщений и формулой для общего ключа:

$$\begin{aligned} A \to B: &\quad T_A = [r_A](Q_A + [h_A]Q_{\mathrm{pub}}), Q_A, \\ A \leftarrow B: &\quad T_B = [r_B](Q_B + [h_B]Q_{\mathrm{pub}}), Q_B. \end{aligned}$$

Теперь A и B могут вычислить общий ключ $K = [(r_A + r_B)\sigma_A\sigma_B]P$ по формулам

$$K_A = [\sigma_A](T_B + [r_A](Q_B + [h_B]Q_{\mathrm{pub}})), \qquad K_B = [\sigma_B](T_A + [r_B](Q_A + [h_B]Q_{\mathrm{pub}})).$$

IBKE-протокол Горейши и др.

Протокол предложен в 2015 г. [24]. Пусть $h_1 : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_q$, $h_2 : \{0,1\}^* \to \mathbb{Z}_q^*$. Помимо ключевой пары (Q_A, σ_A), пользователь A выбирает случайное число $x_A \in_R \mathbb{Z}_q^*$ и вычисляет $X_A = x_A P$, $y_A = \sigma_A + h_2(\mathtt{ID}_A)x_A \bmod q$, $Y_A = [y_A]P$. Протокол предполагает выполнение предварительного этапа обмена значениями (Q_A, P_A) и (Q_B, P_B):

$$\begin{aligned} A \to B: &\quad Q_A, X_A, \\ A \leftarrow B: &\quad Q_B, X_B, \\ A \to B: &\quad T_A = [r_A \sigma_A y_A] Y_B, \\ A \leftarrow B: &\quad T_B = [r_B \sigma_B y_B] Y_A. \end{aligned}$$

Пользователи A и B вычисляют по формулам $K_A = [r_A \sigma_A] T_B$ и $K_B = [r_B \sigma_B] T_A$ общий ключ

$$K = [r_A r_B \sigma_A \sigma_B y_A y_B] P.$$

Данный протокол более быстрый, так как в нём требуется только два раза вычислять кратные точки вместо трёх.

Выводы

В работе проанализированы основные положительные и отрицательные свойства криптографических систем с открытыми ключами, вычисляемыми на основе идентификационной информации. Несмотря на очевидные достоинства, связанные с упрощенной процедурой распределения ключей, такие системы обладают целым рядом ограничений, вытекающих из способа их построения. К их числу относятся: трудность масштабирования на распределённые системы с большим числом пользователей, необходимость наличия защищённого канала для получения ключей пользователями, высокая степень доверия к центру генерации ключей, имеющему возможность в любой момент восстанавливать все ранее им выданные личные ключи пользователей, сложность процедур отзыва и обновления ключей и др. Описаны способы защиты от возможных уязвимостей и перечислены применяемые при этом математические конструкции.

ЛИТЕРАТУРА

1. *Abdalla M., Bellare M., Catalano D., et al.* Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions // LNCS. 2005. V. 3621. P. 205–222.
2. *Al-Riyami S. S. and Paterson K. G.* Certificateless public key cryptography // LNCS. 2003. V. 2894. P. 452–473.
3. *Au M., Liu J., Susilo W., and Yuen T.* Certificate based (linkable) ring signature // LNCS. 2007. V. 4464. P. 79–92.
4. *Barreto P. S. L. M., Libert B., McCullagh N., and Quisquater J-J.* Efficient and Secure Identity-Based Signatures and Signcryption from Bilinear Maps. `https://www.slideserve.com/connie/efficient-and-secure-identity-based-signatures-and-signcryption-from-bilinear-maps`.
5. *Barreto P. S. L. M., Libert B., McCullagh N., and Quisquater J-J.* Efficient and provably-secure identity-based signatures and signcryption from bilinear maps // LNCS. 2005. V. 3788. P. 515–532.
6. *Boldyreva A., Goyal V., and Kumar V.* Identity-based encryption with efficient revocation // Proc. CCS'08. N.Y.: ACM, 2008. P. 417–426.

7. *Baek J., Newmarch J., Safavi-Naini R., and Susilo W.* A survey of identity-based cryptography // Proc. Australian Unix Users Group Annual Conf. 2004. P. 95–102.
8. *Boneh D and Boyen X.* Efficient selective-ID secure Identity-Based Encryption without random oracles // LNCS. 2004. V. 3027. P. 223–238.
9. *Boneh D. and Franklin M.* Identity based encryption from the Weil pairing // LNCS. 2001. V. 2139. P. 213–229; SIAM J. Comput. 2003. V. 32. No. 3. P. 586–615.
10. *Boyd C., Mathura A., and Stebila D.* Protocols for Authentication and Key Establishment. 2nd ed. Berlin; Heidelberg: Springer, 2020. 521 p.
11. *Cao X., Kou W., and Du X.* A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges // Inform. Sci. 2010. V. 180. Iss. 15. P. 2895–2903.
12. *Cha J. C. and Cheon J. H.* An identity-based signature from gap Diffie — Hellman groups // LNCS. 2003. V. 2567. P. 18–30.
13. *Chatterjee S. and Sarkar P.* Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model // LNCS. 2005. V. 3935. P. 424–440.
14. *Chen L. and Cheng Z.* Security proof of Sakai — Kasahar's identity-based encryption scheme // LNCS. 2005. V. 3796. P. 442–459.
15. *Chen L.* Identity-based Cryptography. Hewlett-Packard Laboratories. September 2006. `http://www.sti.uniurb.it/events/fosad06/papers/Chen-fosad06.pdf`.
16. *Chatterjee S. and Sarkar P.* Identity-Based Encryption. N.Y.: Springer, 2011. 180 p.
17. *Chen L., Cheng Z., and Smart N. P.* Identity-based Key Agreement Protocols from Pairings. Cryptology ePrint Archive. Report 2006/199. 2006. `https://eprint.iacr.org/2006/199.pdf`.
18. *Chen X., Zhang F., and Kim K.* A new ID-based group signature scheme from bilinear pairings // LNCS. 2003. V. 2908. P. 585–592.
19. *Cocks C.* An identity-based encryption scheme based on quadratic residues // LNCS. 2001. V. 2260. P. 360–363.
20. *Gagné M.* Identity-Based Encryption: a Survey // RSA Laboratories Cryptobytes. 2003. V. 6. No. 1. P. 10–19.
21. *Galindo D.* Boneh-Franklin identity based encryption revisited // Proc. ICALP 2005. Lisbon, Portugal, 2005. P. 791–802
22. *Gentry C.* Certificate-based encryption and the certificate revocation problem // LNCS. 2003. V. 2656. P. 272–293.
23. *Gentry C. and Silverberg A.* Hierarchical ID-based cryptography // LNCS. 2002. V. 2501. P. 548–566.
24. *Ghoreishi S.-M., Isnin I. F., Razak S. A., and Chizari H.* Secure and authenticated key agreement protocol with minimal complexity of operations in the context of identity-based cryptosystems // Proc. I4CT. Kuching, Malaysia, 2015. P. 299–303.
25. *Girault M. and Paillès J. C.* An identity-based scheme providing zero-knowledge authentication and authenticated key exchange // Proc. ESORICS. AFCET, Toulouse, 1990. P. 173–184.
26. *Girault M.* Self-certified public keys // LNCS. 1991. V. 547. P. 490–497.
27. *Gorantla M. C., Gangishetti R., and Saxena A.* A Survey on ID-Based Cryptographic Primitives. `http://eprint.iacr.org/2005/094`.
28. *Green M. and Hohenberger S.* Blind identity-based encryption and simulatable oblivious transfer // LNCS. 2007. V. 4833. P. 265–282.
29. *Guillou L. and Quisquater J.-J.* A practical zero knowledge protocol fitted to security microprocessor minimizing both transmission and memory // LNCS. 1988. V. 330. P. 123–128.

30. *Guillou L. C. and Quisquater J.-J.* A "paradoxical" identity-based signature scheme resulting from zero-knowledge // LNCS. 1990. V. 403. P. 216–231.
31. *Guillou L. C., Ugon M., and Quisquater J.-J.* Cryptographic authentication protocols for smart cards // Computer Networks Magazine. 2002. V. 36. P. 437–451.
32. *Günther C. G.* An identity-based key-exchange protocol // LNCS. 1990. V. 434. P. 29–37.
33. *Grumăzescu C. and Patriciu V-V.* A comprehensive survey on ID-based cryptography for wireless sensor networks // J. Military Technology. 2018. V. 1. No. 1. P. 57–70.
34. *Horwitz J. and Lynn B.* Toward hierarchical identity-based encryption // LNCS. 2002. V. 2332. P. 466–481.
35. *Hess F.* Efficient identity based signature schemes based on pairings // LNCS. 2003. V. 2595. P. 310–324.
36. *Islam H. and Biswas G. P.* An improved pairing-free identity-based authenticated key agreement protocol based on ECC // Procedia Engineering. 2012. V. 30. P. 499–507.
37. *Kang B. G., Park J. H., and Hahn S. G.* A certificate-based signature scheme // LNCS. 2004. V. 2964. P. 99–111.
38. *Katz J.* Binary tree encryption: Constructions and applications // LNCS. 2004. V. 2971. P. 1–11.
39. *Lee K., Lee D. H., and Park J. H.* Efficient revocable identity-based encryption via subset difference methods // Des. Codes Cryptogr. 2017. V. 85. P. 39–76.
40. *Tseng Y. and Tsai T.* Efficient revocable ID-based encryption with a public channel // Computer J. 2012. V. 55. No. 4. P. 475–486.
41. *Li J., Huang X., Mu Y., et al.* Certificate-based signature: Security model and efficient construction // LNCS. 2007. V. 4582. P. 110–125.
42. *Libert B. and Vergnaud D.* Adaptive-ID Secure revocable identity-based encryption // LNCS. 2009. V. 5473. P. 1–15.
43. *Liu J. K., Baek J., Susilo W., and Zhou J.* Certificate-based signature schemes without pairings or random oracles // LNCS. 2008. V. 5222. P. 285–297.
44. *Matsumoto T., Takashima Y., and Imai H.* On seeking smart public-key distribution systems // Trans. IECE. Japan. Sec. E. 1986. V. 69. Iss. 2. P. 99–106.
45. *McCullagh N. and Barreto P. S. L. M.* A new two-party identity-based authenticated key agreement // LNCS. 2005. V. 3376. P. 262–274.
46. *Naccache D.* Secure and Practical Identity-Based Encryption. Cryptology ePrint Archive. Report 2005/369. 2005. `https://eprint.iacr.org/2005/369`.
47. *Nalla D. and Reddy K. C.* Signcryption Scheme for Identity-based Cryptosystems. `https://eprint.iacr.org/2003/066.pdf`.
48. *Okamoto E.* Key distribution systems based on identification information // LNCS. 1987. V. 293. P. 194–202.
49. *Okamoto T.* Efficient blind and partially blind signatures without random oracles // LNCS. 2006. V. 3876. P. 80–99.
50. *Okamoto E. and Tanaka K.* Key distribution system based on identification information // IEEE J. Selected Areas Communications. 1989. V. 7. No. 4. P. 481–485.
51. *Ryu E. K., Yoon E. J., and Yoo K. Y.* An efficient ID-based authenticated key agreement protocol from pairings // LNCS. 2004. V. 3042. P. 1464–1469.
52. *Saeednia S.* Improvement of Gunther's identity-based key exchange protocol // Electronics Lett. 2000. V. 36. No. 18. P. 1535–1536.
53. *Sakai R., Ohgishi K., and Kasahara M.* Cryptosystems based on pairing // Proc. SCIS'00. Okinawa, Japan, 2000. P. 26–28.

54. *Sakai R., Ohgishi K., and Kasahara M.* Cryptosystems based on pairing over elliptic curve // Proc. Symp. on Cryptography and Information Security. Oiso, Japan, January 2001. (in Japanese)
55. *Sakai R. and Kasahara M.* ID Based Cryptosystems with Pairing on Elliptic Curve. Cryptology ePrint Archive. Report 2003/054. `https://eprint.iacr.org/2003/054.pdf`. 2003.
56. *Sahai A. and Waters B.* Fuzzy identity-based encryption // LNCS. 2005. V. 3494. P. 457–473.
57. *Sayid J., Sayid I., and Kar J.* Certificateless public key cryptography: A research survey // Intern. J. Security Appl. 2016. V. 10. No. 7. P. 103–118.
58. *Seo J. H. and Emura K.* Revocable hierarchical identity-based encryption // Theor. Comput. Sci. 2014. V. 542. P. 44–62.
59. *Scott M.* Authenticated ID-Based Key Exchange and Remote Log-in with Simple Token and PIN Number. Cryptology ePrint Archive. 2002. Report 2002/164. `https://eprint.iacr.org/2002/164`.
60. *Shamir A.* Identity-based cryptosystems and signature schemes // LNCS. 1984. V. 196. P. 47–53.
61. *Smart N. P.* An identity based authenticated key agreement protocol based on the Weil pairing // Electronics Lett. 2002. V. 38. No. 13. P. 630–632.
62. *Shim K.* Efficient ID-based authenticated key agreement protocol based on Weil pairing // Electronics Lett. 2003. V. 39. No. 8. P. 653–654.
63. *Sun H.-M. and Hsieh B.-T.* Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive. 2003. Report 2003/113. `http://eprint.iacr.org/2003/113`.
64. *Wang S., Cao Z., Choo K. K. R., and Wang L.* An improved identity-based key agreement protocol and its security proof // Inf. Sci. 2009. V. 179. No. 3. P. 307–318.
65. *Wang Y.* Efficient identity-based and authenticated key agreement protocols // LNCS. 2013. V. 7420. P. 172–197.
66. *Waters B.* Efficient identity-based encryption without random oracles // Proc. EUROCRYPT'05. Aarhus, Denmark, 2005. P. 114–127.
67. *Yao D., Fazio N., Dodis Y., and Lysyanskaya A.* Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption // Proc. CCS'04. Washington: ACM, 2004. P. 354–363.
68. *Zheng Y.* Digital signcryption or how to achieve cost(signature&encryption) $\ll$ cost(signature) + cost(encryption) // LNCS. 1997. V. 1294. P. 165–179.
69. ISO/IEC 14888-2. Information Technology — Security Techniques — Digital Signatures with Appendix — P. 2: Integer Factorization Based Mechanisms. ISO/IEC, 1999.
70. ISO/IEC 14888-3. Information Technology — Security Techniques — Digital Signatures with Appendix — P. 3: Discrete Logarithm Based Mechanisms. ISO/IEC, 1998.
71. ISO/IEC 11770-3. Information Technology — Security Techniques — Key Management — P. 3: Mechanisms Using Asymmetric Techniques. ISO/IEC, 1999.
72. IEEE P1363.3. Identity-Based Public Key Cryptography Using Pairings. `https://standards.ieee.org/ieee/1363.3/3822/`. 2013.
73. GM/T 0044.2-2016. Identity-Based Cryptographic Algorithm using Bilinear Pairings — P. 2: Digital Signature Algorithm. 2016. (in Chinese).

REFERENCES

1. *Abdalla M., Bellare M., Catalano D., et al.* Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. LNCS, 2005, vol. 3621, pp. 205–222.

2. *Al-Riyami S. S. and Paterson K. G.* Certificateless public key cryptography. LNCS, 2003, vol. 2894, pp. 452–473.
3. *Au M., Liu J., Susilo W., and Yuen T.* Certificate based (linkable) ring signature. LNCS, 2007, vol. 4464, pp. 79–92.
4. *Barreto P. S. L. M., Libert B., McCullagh N., and Quisquater J-J.* Efficient and Secure Identity-Based Signatures and Signcryption from Bilinear Maps. `https://www.slideserve.com/connie/efficient-and-secure-identity-based-signatures-and-signcryption-from-bilinear-maps`.
5. *Barreto P. S. L. M., Libert B., McCullagh N., and Quisquater J-J.* Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. LNCS, 2005, vol. 3788, pp. 515–532.
6. *Boldyreva A., Goyal V., and Kumar V.* Identity-based encryption with efficient revocation. Proc. CCS'08, N.Y., ACM, 2008, pp. 417–426.
7. *Baek J., Newmarch J., Safavi-Naini R., and Susilo W.* A survey of identity-based cryptography. Proc. Australian Unix Users Group Annual Conf., 2004, pp. 95–102.
8. *Boneh D and Boyen X.* Efficient selective-ID secure Identity-Based Encryption without random oracles. LNCS, 2004, vol. 3027, pp. 223–238.
9. *Boneh D. and Franklin M.* Identity based encryption from the Weil pairing. LNCS, 2001, vol. 2139, pp. 213–229; SIAM J. Comput., 2003, vol. 32, no. 3, pp. 586–615.
10. *Boyd C., Mathura A., and Stebila D.* Protocols for Authentication and Key Establishment. 2nd ed. Berlin; Heidelberg, Springer, 2020. 521 p.
11. *Cao X., Kou W., and Du X.* A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. Inform. Sci., 2010, vol. 180, iss. 15, pp. 2895–2903.
12. *Cha J. C. and Cheon J. H.* An identity-based signature from gap Diffie — Hellman groups. LNCS, 2003, vol. 2567, pp. 18–30.
13. *Chatterjee S. and Sarkar P.* Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. LNCS, 2005, vol. 3935, pp. 424–440.
14. *Chen L. and Cheng Z.* Security proof of Sakai — Kasahar's identity-based encryption scheme. LNCS, 2005, vol. 3796, pp. 442–459.
15. *Chen L.* Identity-based Cryptography. Hewlett-Packard Laboratories. September 2006. `http://www.sti.uniurb.it/events/fosad06/papers/Chen-fosad06.pdf`.
16. *Chatterjee S. and Sarkar P.* Identity-Based Encryption. N.Y., Springer, 2011. 180 p.
17. *Chen L., Cheng Z., and Smart N. P.* Identity-based Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2006/199, 2006. `https://eprint.iacr.org/2006/199.pdf`.
18. *Chen X., Zhang F., and Kim K.* A new ID-based group signature scheme from bilinear pairings. LNCS, 2003, vol. 2908, pp. 585–592.
19. *Cocks C.* An identity-based encryption scheme based on quadratic residues. LNCS, 2001, vol. 2260, pp. 360–363.
20. *Gagné M.* Identity-Based Encryption: a Survey. RSA Laboratories Cryptobytes, 2003, vol. 6, no. 1, pp. 10–19.
21. *Galindo D.* Boneh-Franklin identity based encryption revisited. Proc. ICALP 2005, Lisbon, Portugal, 2005, pp. 791–802
22. *Gentry C.* Certificate-based encryption and the certificate revocation problem. LNCS, 2003, vol. 2656, pp. 272–293.
23. *Gentry C. and Silverberg A.* Hierarchical ID-based cryptography. LNCS, 2002, vol. 2501, pp. 548–566.

24. *Ghoreishi S.-M., Isnin I. F., Razak S. A., and Chizari H.* Secure and authenticated key agreement protocol with minimal complexity of operations in the context of identity-based cryptosystems. Proc. I4CT. Kuching, Malaysia, 2015. pp. 299–303.
25. *Girault M. and Paillès J. C.* An identity-based scheme providing zero-knowledge authentication and authenticated key exchange. Proc. ESORICS, AFCET, Toulouse, 1990, pp. 173–184.
26. *Girault M.* Self-certified public keys. LNCS, 1991, vol. 547, pp. 490–497.
27. *Gorantla M. C., Gangishetti R., and Saxena A.* A Survey on ID-Based Cryptographic Primitives. `http://eprint.iacr.org/2005/094`.
28. *Green M. and Hohenberger S.* Blind identity-based encryption and simulatable oblivious transfer. LNCS, 2007, vol. 4833, pp. 265–282.
29. *Guillou L. and Quisquater J.-J.* A practical zero knowledge protocol fitted to security microprocessor minimizing both transmission and memory. LNCS, 1988, vol. 330, pp. 123–128.
30. *Guillou L. C. and Quisquater J.-J.* A "paradoxical" identity-based signature scheme resulting from zero-knowledge. LNCS, 1990, vol. 403, pp. 216–231.
31. *Guillou L. C., Ugon M., and Quisquater J.-J.* Cryptographic authentication protocols for smart cards. Computer Networks Magazine, 2002, vol. 36, pp. 437–451.
32. *Günther C. G.* An identity-based key-exchange protocol. LNCS, 1990, vol. 434, pp. 29–37.
33. *Grumăzescu C. and Patriciu V-V.* A comprehensive survey on ID-based cryptography for wireless sensor networks. J. Military Technology, 2018, vol. 1, no. 1, pp. 57–70.
34. *Horwitz J. and Lynn B.* Toward hierarchical identity-based encryption. LNCS, 2002, vol. 2332, pp. 466–481.
35. *Hess F.* Efficient identity based signature schemes based on pairings. LNCS, 2003, vol. 2595, pp. 310–324.
36. *Islam H. and Biswas G. P.* An improved pairing-free identity-based authenticated key agreement protocol based on ECC. Procedia Engineering, 2012, vol. 30, pp. 499–507.
37. *Kang B. G., Park J. H., and Hahn S. G.* A certificate-based signature scheme. LNCS, 2004. vol. 2964. pp. 99–111.
38. *Katz J.* Binary tree encryption: Constructions and applications. LNCS, 2004, vol. 2971, pp. 1–11.
39. *Lee K., Lee D. H., and Park J. H.* Efficient revocable identity-based encryption via subset difference methods. Des. Codes Cryptogr., 2017, vol. 85, pp. 39–76.
40. *Tseng Y. and Tsai T.* Efficient revocable ID-based encryption with a public channel. Computer J., 2012, vol. 55, no. 4, pp. 475–486.
41. *Li J., Huang X., Mu Y., et al.* Certificate-based signature: Security model and efficient construction. LNCS, 2007, vol. 4582, pp. 110–125.
42. *Libert B. and Vergnaud D.* Adaptive-ID Secure revocable identity-based encryption. LNCS, 2009, vol. 5473, pp. 1–15.
43. *Liu J. K., Baek J., Susilo W., and Zhou J.* Certificate-based signature schemes without pairings or random oracles. LNCS, 2008, vol. 5222, pp. 285–297.
44. *Matsumoto T., Takashima Y., and Imai H.* On seeking smart public-key distribution systems. Trans. IECE, Japan, Sec. E, 1986, vol. 69, iss. 2, pp. 99–106.
45. *McCullagh N. and Barreto P. S. L. M.* A new two-party identity-based authenticated key agreement. LNCS, 2005, vol. 3376, pp. 262–274.
46. *Naccache D.* Secure and Practical Identity-Based Encryption. Cryptology ePrint Archive. Report 2005/369. 2005. `https://eprint.iacr.org/2005/369`.

47. *Nalla D. and Reddy K. C.* Signcryption Scheme for Identity-based Cryptosystems. https://eprint.iacr.org/2003/066.pdf.
48. *Okamoto E.* Key distribution systems based on identification information. LNCS, 1987, vol. 293, pp. 194–202.
49. *Okamoto T.* Efficient blind and partially blind signatures without random oracles. LNCS, 2006, vol. 3876, pp. 80–99.
50. *Okamoto E. and Tanaka K.* Key distribution system based on identification information. IEEE J. Selected Areas Communications. 1989, vol. 7, no. 4, pp. 481–485.
51. *Ryu E. K., Yoon E. J., and Yoo K. Y.* An efficient ID-based authenticated key agreement protocol from pairings. LNCS, 2004, vol. 3042, pp. 1464–1469.
52. *Saeednia S.* Improvement of Gunther's identity-based key exchange protocol. Electronics Lett., 2000, vol. 36, no. 18, pp. 1535–1536.
53. *Sakai R., Ohgishi K., and Kasahara M.* Cryptosystems based on pairing. Proc. SCIS'00. Okinawa, Japan, 2000, pp. 26–28.
54. *Sakai R., Ohgishi K., and Kasahara M.* Cryptosystems based on pairing over elliptic curve. Proc. Symp. on Cryptography and Information Security, Oiso, Japan, January 2001. (in Japanese)
55. *Sakai R. and Kasahara M.* ID Based Cryptosystems with Pairing on Elliptic Curve. Cryptology ePrint Archive, Report 2003/054. https://eprint.iacr.org/2003/054.pdf. 2003.
56. *Sahai A. and Waters B.* Fuzzy identity-based encryption. LNCS, 2005, vol. 3494, pp. 457–473.
57. *Sayid J., Sayid I., and Kar J.* Certificateless public key cryptography: A research survey. Intern. J. Security Appl., 2016. vol. 10, no. 7, pp. 103–118.
58. *Seo J. H. and Emura K.* Revocable hierarchical identity-based encryption. Theor. Comput. Sci., 2014. vol. 542, pp. 44–62.
59. *Scott M.* Authenticated ID-Based Key Exchange and Remote Log-in with Simple Token and PIN Number. Cryptology ePrint Archive, 2002, Report 2002/164. https://eprint.iacr.org/2002/164.
60. *Shamir A.* Identity-based cryptosystems and signature schemes. LNCS, 1984, vol. 196, pp. 47–53.
61. *Smart N. P.* An identity based authenticated key agreement protocol based on the Weil pairing. Electronics Lett., 2002, vol. 38, no. 13, pp. 630–632.
62. *Shim K.* Efficient ID-based authenticated key agreement protocol based on Weil pairing. Electronics Lett., 2003, vol. 39, no. 8, pp. 653–654.
63. *Sun H.-M. and Hsieh B.-T.* Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, 2003, Report 2003/113. http://eprint.iacr.org/2003/113.
64. *Wang S., Cao Z., Choo K. K. R., and Wang L.* An improved identity-based key agreement protocol and its security proof. Inf. Sci., 2009, vol. 179, no. 3, pp. 307–318.
65. *Wang Y.* Efficient identity-based and authenticated key agreement protocols. LNCS, 2013, vol. 7420, pp. 172–197.
66. *Waters B.* Efficient identity-based encryption without random oracles. Proc. EUROCRYPT'05, Aarhus, Denmark, 2005, pp. 114–127.
67. *Yao D., Fazio N., Dodis Y., and Lysyanskaya A.* Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. Proc. CCS'04, Washington, ACM, 2004, pp. 354–363.
68. *Zheng Y.* Digital signcryption or how to achieve cost(signature&encryption) $\ll$ cost(signature) + cost(encryption). LNCS, 1997, vol. 1294, pp. 165–179.

69. ISO/IEC 14888-2. Information Technology — Security Techniques — Digital Signatures with Appendix — P. 2: Integer Factorization Based Mechanisms. ISO/IEC, 1999.
70. ISO/IEC 14888-3. Information Technology — Security Techniques — Digital Signatures with Appendix — P. 3: Discrete Logarithm Based Mechanisms. ISO/IEC, 1998.
71. ISO/IEC 11770-3. Information Technology — Security Techniques — Key Management — P. 3: Mechanisms Using Asymmetric Techniques. ISO/IEC, 1999.
72. IEEE P1363.3. Identity-Based Public Key Cryptography Using Pairings. `https://standards.ieee.org/ieee/1363.3/3822/`. 2013.
73. GM/T 0044.2-2016. Identity-Based Cryptographic Algorithm using Bilinear Pairings — P. 2: Digital Signature Algorithm. 2016. (in Chinese).

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 004.94 DOI 10.17223/20710410/61/5

МОДЕЛЬ И МЕТРИКИ ОСВЕДОМЛЕННОСТИ В КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ. ЧАСТЬ 1. ПОТЕНЦИАЛЬНАЯ ОСВЕДОМЛЕННОСТЬ

Н. А. Гайдамакин

Уральский федеральный университет имени первого Президента России Б. Н. Ельцина, г. Екатеринбург, Россия

E-mail: n.a.gaidamakin@urfu.ru

В рамках субъектно-объектной формализации компьютерных систем введены понятия потенциальной и фактической осведомлённости пользователей в конфиденциальной информации. Потенциальная осведомлённость рассматривается как величина, определяющаяся имеющимися у пользователя правами доступа к объектам, содержащим конфиденциальную информацию, и объёмом конфиденциальной информации соответствующих объектов. Объём конфиденциальной информации объекта предложено определять на основе количества слов, содержащихся в тексте объекта, и величины информативности объекта, которая устанавливается внешним фактором, например автором и/или выделенным пользователем (аналитиком). Для основных моделей управления доступом (дискреционной, мандатной, тематико-иерархической и ролевой) представлены аналитические соотношения, определяющие в количественной шкале диапазона $[0, 1]$ величину потенциальной осведомлённости пользователей в конфиденциальной информации, содержащейся (обрабатывающейся) в компьютерной системе. Доказано удовлетворение соответствующих величин требованиям метрики.

Ключевые слова: *конфиденциальная информация, осведомлённость, потенциальная осведомлённость, модель осведомлённости, метрики осведомлённости, управление доступом, права доступа, субъекты доступа, объекты доступа.*

THE MODEL AND METRICS OF AWARENESS IN CONFIDENTIAL INFORMATION. PART 1. POTENTIAL AWARENESS

N. A. Gaydamakin

Ural Federal University named after the first President of Russia B. N. Yeltsin, Ekaterinburg, Russia

As part of the subject-object formalization of computer systems, the concepts of potential and actual user awareness of confidential information are introduced. Potential awareness is considered as a value determined by the user's access rights to objects containing confidential information and the volume of confidential information of the corresponding objects. The volume of confidential information of the object is proposed to be determined on the basis of the number of words contained in the text of the

object and the amount of information content of the object, which is determined by an external factor, for example, the author and/or a dedicated user (analyst). For the main access control models (discretionary, mandatory, thematic-hierarchical and role-based), analytical relations are presented that determine, on a quantitative scale of the range $[0, 1]$, the amount of potential awareness of users in confidential information contained (processed) in a computer system. The satisfaction of the corresponding values to the metric requirements is proved.

Keywords: *confidential information, awareness, potential awareness, awareness model, awareness metrics, access control, access rights, access subject, access objects.*

Введение

Анализ осведомлённости в конфиденциальной информации является важной составляющей мониторинга процессов обеспечения информационной безопасности.

В толковом словаре Ушакова [1] «*осведомлённость*» определяется как *знание, наличие сведений о чем-либо*. В практической и нормативной сфере употребляется смежный, во многих случаях синонимичный, термин «*владение информацией*». Иначе говоря, быть осведомлённым означает владеть определённой информацией, т. е. знать соответствующие сведения, сообщения, данные, составляющие (выражающие) информацию.

В качестве субъекта владения рассматривается человек, являющийся в контексте компьютерной сферы пользователем, осуществляющим доступ к информации, точнее, к объектам, содержащим конфиденциальную информацию.

Модели управления доступом пользователей к информации в компьютерных системах [2, 3] регламентируют правила санкционирования доступов в соответствии с правами, задаваемыми явно (дискреционная модель — DAC, Discretionary Access Control) или посредством соотношения меток безопасности (мандатная модель — MAC, Mandatory Access Control). В рамках имеющихся прав пользователи осуществляют доступы к объектам, в результате которых формируется их осведомлённость в конфиденциальной информации, содержащейся (обрабатывающейся) в компьютерной системе. Соответственно можно выделить «потенциальную» и «фактическую» осведомлённость.

Потенциальная осведомлённость (Potential Awareness) пользователя определяется имеющимися у него правами доступа, реализуя которые, пользователь может стать осведомлённым в конфиденциальной информации.

Фактическая осведомлённость (Actual Awareness) пользователя является результатом осуществления доступов к конфиденциальной информации.

Рассмотрение формализованной и процедурно-аналитической стороны потенциальной осведомлённости является предметом данной работы.

1. Исходные положения

Будем использовать распространённую парадигму в сфере компьютерной безопасности, в рамках которой компьютерная система рассматривается как совокупность субъектов и объектов доступа. Под *субъектами*, именуемыми активными сущностями, понимаются выполняющиеся по командам пользователей компьютерные программы. Под *объектами*, именуемыми пассивными сущностями, понимаются элементарные информационные структуры (файлы, таблицы баз данных, их строки, поля, записи) или составные (каталоги, базы данных), к которым пользователями осуществляется доступ на чтение или запись (изменение).

В рамках субъектно-объектного подхода сделаем несколько исходных предположений и определений.

Положение 1. Компьютерная система представляется множеством объектов доступа $o \in O$ и множеством субъектов доступа $s \in S$, которые управляются пользователями $u \in U$.

В дальнейшем в процессах анализа доступов к объектам будем отождествлять понятие субъекта и пользователя, оговаривая особенности такого допущения в необходимых случаях.

Положение 2. В компьютерной системе действует дискретное время, в каждый момент t_k которого пользователи $u \in U$ посредством субъектов $s \in S$ осуществляют доступы к объектам $o \in O$.

Определение 1. Под *доступом* будем понимать имеющий временны́е рамки процесс воздействия субъекта $s \in S$ на объект $o \in O$, в результате которого формируется поток информации — однонаправленный, т. е. от объекта к субъекту или от субъекта (через субъект) к объекту, либо двунаправленный, т. е. одновременно от субъекта к объекту и от объекта к субъекту.

Однонаправленный поток от объекта к субъекту реализуется в рамках доступа вида «Чтение» (Read), от субъекта к объекту — вида «запись» (Write). Двунаправленный поток реализуется в рамках доступов вида «Чтение» и «Запись», одновременно осуществляемых субъектом к соответствующему объекту. Далее в контексте анализа осведомлённости ограничимся рассмотрением только доступов вида «Чтение» к объектам, содержащим текстовую информацию.

Существуют различные подходы к понятию *информационного потока*. В частности, в «детерминистской» трактовке информационный поток рассматривается как процесс изменения слова, характеризующего (описывающего, составляющего) объект-приёмник, в который поступает информация в виде слова, характеризующего объект-источник информационного потока [4]. В теоретико-информационном смысле объект доступа рассматривается как слово некоторого языка в определённом алфавите. В рамках отмеченных ограничений (рассмотрение только доступов вида «Read» к объектам, содержащим текстовую информацию) соответствующим языком будем считать естественный язык письменной речи. Под «словом» понимается в том числе и совокупность слов, характеризующих, выражающих информацию или часть информации объекта. В теоретико-вероятностной трактовке информационный поток рассматривается как процесс изменения неопределённости состояния объекта (изменения множества его возможных состояний) [5].

Отметим, что при «детерминистском» подходе в случае чтения объекта изменяется слово, характеризующее состояние субъекта, точнее, домена, выделенного субъекту [4], т. е. областей (буферов) оперативной памяти компьютерного устройства, в которых размещаются исполняемый код и данные соответствующего вычислительного процесса и визуальное отображение которых непосредственно воспринимается пользователем. Иначе говоря, в упрощённой трактовке объектом-приёмником при чтении объекта можно считать области оперативной видеопамяти, выделенной вычислительному процессу субъекта.

Определение 2. Под *объёмом* (*количеством*) $V(o_n, t_k)$ *конфиденциальной информации* объекта o_n понимается величина, пропорциональная количеству слов $Q(o_n, t_k)$, содержащихся в момент времени t_k в тексте объекта o_n:

$$V(o_n, t_k) = Q(o_n, t_k)\theta(o_n, t_k),$$

где $\theta(o_n, t_k)$ — изменяющийся в диапазоне $[0, 1]$ коэффициент информативности объекта o_n в момент времени t_k (1 — максимальная информативность).

Введение коэффициента информативности $\theta(o_n, t_k)$ обусловлено тем, что языком письменной речи одну и ту же информацию можно выразить по-разному, с разной ясностью, чёткостью, полнотой, «понятностью» и, следовательно, с различным словесным объёмом. Очевидно, что информативность является скорее качественным понятием, но будем считать, что существует вещественнозначная функция, выражающая данное свойство объектов в числовом диапазоне $[0, 1]$.

Фундаментальным в сфере компьютерной безопасности является понятие конфиденциальности информации. В специальной литературе и нормативных документах приводятся различные определения понятия конфиденциальности информации [6–8], отталкиваясь от которых дадим следующее определение.

Определение 3. Под *конфиденциальностью* информации будем понимать такое свойство информации, устанавливаемое федеральным законом либо обладателем информации, когда может быть причинён ущерб гражданам, организациям, обществу, государству либо обладателю информации при свободном обороте таковой информации (свободном доступе к ней и соответственно осведомлённости в ней неопределённого круга лиц), при условии того, что информация известна только уполномоченным лицам и обладателем информации принимаются и реализуются меры по ограничению доступа к этой информации неуполномоченных лиц.

Таким образом, конфиденциальность является свойством определённой информации. Иначе говоря, не все объекты доступа содержат конфиденциальную информацию или не вся информация объекта является конфиденциальной.

Конфиденциальность как свойство информации является качественным понятием и в дискреционной модели управления доступом рассматривается как дихотомическая характеристика (информация «конфиденциальна/неконфиденциальна»), в мандатной модели — как характеристика в порядково-вербальной шкале (как правило, в трёх градациях — «высокая конфиденциальность», «средняя», «низкая»). В практических приложениях могут использоваться расширенные системы классификации конфиденциальности информации с большим количеством градаций с квалиметрическими характеристиками каждого уровня [9, Приложение А].

Из определения 3 следует, что конфиденциальность информации может изменяться с течением времени.

Положение 3. Существует вещественнозначная функция $f_{\text{conf}}(o_n, t_k)$, которая каждому объекту компьютерной системы $o_n \in O$ в каждый момент времени t_k ставит в соответствие некоторую величину (значение) конфиденциальности $\mathcal{K} = f_{\text{conf}}(o_n, t_k)$.

Функцию конфиденциальности $f_{\text{conf}}(o_n, t_m)$ в системах дискреционного и мандатного управления доступом можно рассматривать как кусочно-постоянную функцию, значения которой в интересах нормирования устанавливаются в диапазоне $[0, 1]$ (рис. 1 и 2). В общем случае вид и параметры функции $f_{\text{conf}}(o_n, t_k)$ определяются особенностями предметной области и обладателем информации (собственником компьютерной системы).

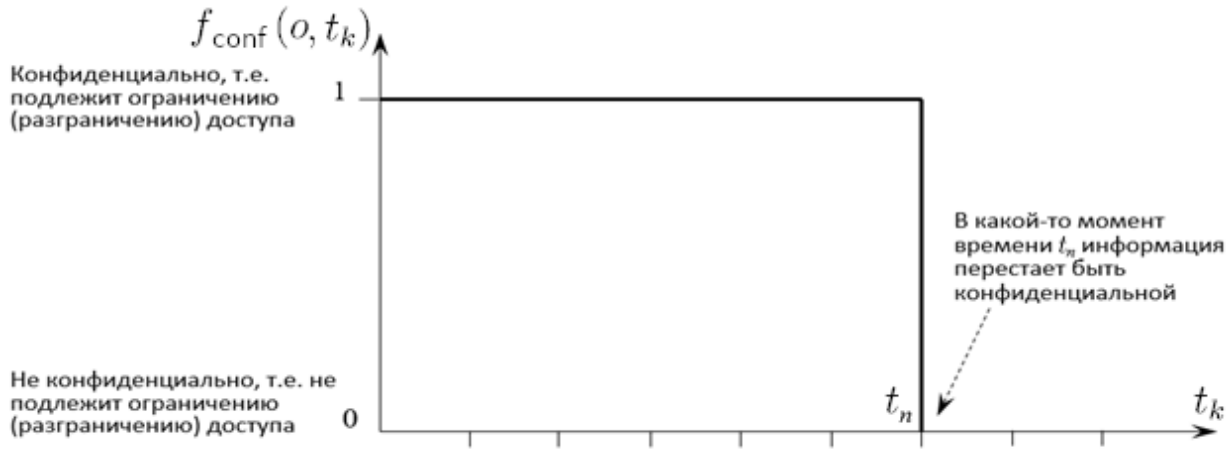

Рис. 1. Пример функции конфиденциальности объектов $f_{\text{conf}}(o, t_k)$ при дискреционном управлении доступом

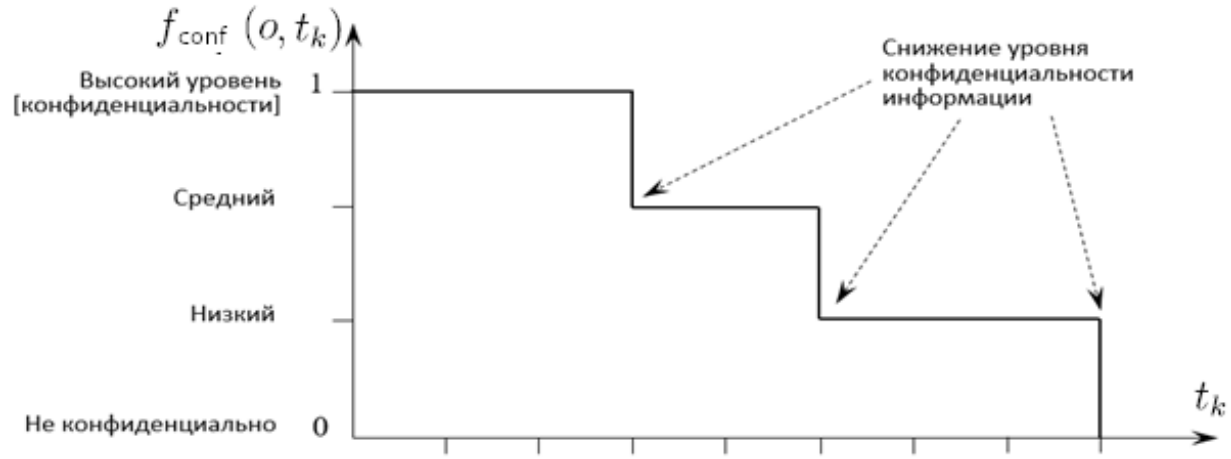

Рис. 2. Пример функции конфиденциальности объектов $f_{\text{conf}}(o, t_k)$ при мандатном управлении доступом

2. Потенциальная осведомлённость пользователей в конфиденциальной информации в системах дискреционного управления доступом

В дискреционной модели управления доступом (DAC) права пользователей на доступ к объектам устанавливаются явно в виде троек «пользователь — разрешённая операция доступа — объект» и фиксируются в тех или иных информационных структурах (ACL, Access Control List — списки управления доступом к файлам в операционных системах; системные таблицы прав доступа баз данных в СУБД).

Математическим образом соответствующих структур, в частности совокупности ACL файлов в операционных системах является матрица доступа $\mathbf{A}$, строки которой соответствуют пользователям $u_l \in U$, столбцы — объектам доступа $o_n \in O$, в ячейках

записываются идентификаторы разрешённых процедур доступа, например:

$$\mathbf{A} = \begin{array}{c} \\ u_1 \\ u_2 \\ \dots \\ u_L \end{array} \begin{array}{c} \begin{array}{cccc} o_1 & o_2 & \dots & o_N \end{array} \\ \begin{pmatrix} \text{Read} & - & \dots & - \\ - & \text{Read, Write} & \dots & \text{Read} \\ \dots & \dots & \dots & \dots \\ \text{Write} & - & \dots & \text{Read, Write} \end{pmatrix} \end{array}. \tag{1}$$

Каждый ACL рассматривается как столбец матрицы доступа, из которого удалены все пустые ячейки. Поскольку в процессе анализа осведомлённости мы ограничились только доступами «Чтение» (Read), то элементы матрицы доступа a_{ln} будем представлять двоичными числами:

$$a_{ln} = \begin{cases} 1, & \text{если Read} \in (\mathbf{A})_{ln}; \\ 0, & \text{если Read} \notin (\mathbf{A})_{ln}. \end{cases}$$

В такой кодировке матрица доступа (1) выглядит следующим образом:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Как в теоретических моделях управления доступом, так и при их практической реализации в компьютерных системах права доступа пользователей к объектам могут изменяться. Отсюда с учётом положения 2 матрица доступа является динамическим объектом, т. е. её структура (размеры) и значения элементов могут быть различными в разные моменты времени. Данное обстоятельство будем отображать временной зависимостью матрицы доступа и её элементов — $\mathbf{A}(t_k), a_{ln}(t_k)$.

Сделаем следующее очевидное предположение.

Положение 4. *Потенциальная осведомлённость* (*Potential Awareness*) пользователя в конфиденциальной информации тем больше, чем больше у него прав доступа к объектам компьютерной системы и чем больше конфиденциальной информации имеется в соответствующих объектах.

Очевидно, ситуацию, в которой пользователь имеет права доступа на чтение ко всем объектам компьютерной системы, можно охарактеризовать как максимальную (100 %) потенциальную осведомлённость.

Тогда, основываясь на положении 4, потенциальную осведомлённость $\mathcal{A}_{\text{DAC\_Pot}}(u_l, t_k)$ пользователя u_l в конфиденциальной информации компьютерной системы в момент времени t_k при дискреционном управлении доступом можно определять как

$$\mathcal{A}_{\text{DAC\_Pot}}(u_l, t_k) = \frac{1}{V(O, t_k)} \sum_{n=1}^{N_{t_k}} a_{ln}(t_k) V(o_n, t_k),$$

где $V(O, t_k)$ — объём конфиденциальной информации, содержащейся во всех объектах компьютерной системы в момент времени t_k:

$$V(O, t_k) = \sum_{n=1}^{N_{t_k}} V(o_n, t_k); \tag{2}$$

N_{t_k} — количество объектов с конфиденциальной информацией в момент времени t_k. Нетрудно видеть, что если пользователь имеет право доступа на чтение всех объектов компьютерной системы, то его потенциальная осведомлённость $\mathcal{A}_{\text{DAC\_Pot}}(u_l, t_k) = 1$, т. е. 100 %. При полном отсутствии прав на доступ к объектам компьютерной системы (все элементы соответствующей строки матрицы доступа $a_{ln} = 0$) $\mathcal{A}_{\text{DAC\_Pot}}(u_l, t_k) = 0$.

3. Потенциальная осведомлённость пользователей в конфиденциальной информации в системах мандатного управления доступом

В мандатной модели управления доступом (MAC), как уже отмечалось, права доступа пользователей к объектам определяются по соотношению меток безопасности субъектов и объектов доступа. Метки безопасности conf_i, $i = 1, 2, \ldots, I$ (I — количество уровней конфиденциальности), являются элементами порядковой шкалы абстрактных уровней безопасности. В классической модели мандатного управления доступом множество уровней безопасности рассматривается как линейно упорядоченное множество (линейная решётка). Метки (уровни) безопасности $\text{conf}(u)$, характеризующие пользователей $u \in U$ и соответственно их субъектов $s \in S$, именуются *уровнями доверия*. Метки (уровни) безопасности $\text{conf}(o)$, характеризующие объекты доступа $o \in O$, именуются *уровнями конфиденциальности*.

Права доступа на чтение определяются правилом NRU (No Read Up, «нет чтения вверх»), т. е. пользователь u_l имеет право чтения объекта o_n, если его метка безопасности (уровень доверия) равна или выше метки безопасности объекта (грифа конфиденциальности):

$$\text{conf}(u_l) \geqslant \text{conf}(o_n).$$

Данный порядок определения прав доступа основан на идеологии градуированного доверия пользователям и совершенно не учитывает реальные потребности пользователей в доступе к объектам, исходя из их функциональных обязанностей. Следствием этого является в большинстве случаев избыточность прав доступа, т. е. необусловленность имеющихся прав доступа к объектам с соответствующими грифами конфиденциальности фактическими потребностями конкретных пользователей.

Для устранения избыточности прав доступа, устанавливаемых правилом NRU, в мандатной модели управления доступом дополнительно вводят дискреционную матрицу доступа $\mathbf{A}(t_k)$. Значения матрицы доступа $a_{ln}(t_k)$ определяются, во-первых, правилом NRU, что является необходимым условием:

$$\forall l, n \left(a_{ln}(t_k) \neq \varnothing \Rightarrow \text{conf}(u_l) \geqslant \text{conf}(o_n)\right),$$

а во-вторых, как и в дискреционной модели, фактическими потребностями пользователя u_l в доступе к объекту o_n. В результате при мандатном управлении доступом фактические права доступа пользователей $u \in U$ к объектам $o \in O$ «прописаны» в ячейках той же матрицы доступа $\mathbf{A}(t_k)$.

При этом конфиденциальность объектов является не двоичной величиной (конфиденциально/неконфиденциально, $f_{\text{conf}}(o_n, t_k) \in \{0, 1\}$), а порядковой (порядково-вербальной — $f_{\text{conf}}(o_n, t_k) \in [0, 1]$.

Исходя из этого, потенциальную осведомлённость $\mathcal{A}_{\text{MAC\_Pot}}(u_l, t_k)$ пользователей при мандатном управлении доступом можно определять на основе следующего соотношения:

$$\mathcal{A}_{\text{MAC\_Pot}}(u_l, t_k) = \frac{1}{V(O, t_k)} \sum_{n=1}^{N_{t_k}} a_{ln}(t_k) V(o_n, t_k) f_{\text{conf}}(o_n, t_k),$$

где

$$V(O,t_k)=\sum_{n=1}^{N_{t_k}}V(o_n,t_k)f_{\text{conf}}(o_n,t_k). \tag{3}$$

Отметим, что функция конфиденциальности в выражении (3) отражает не долю конфиденциальной информации в соответствующем объекте, а именно уровень конфиденциальности.

Таким образом, при мандатном управлении доступом уровень потенциальной осведомлённости пользователей в конфиденциальной информации выше не только тогда, когда больше объём доступной пользователю конфиденциальной информации, но и когда выше уровень её конфиденциальности. Иначе говоря, пользователь, который потенциально может владеть большим объёмом конфиденциальной информации невысокого уровня, может оказаться менее осведомлённым (в секретах), чем пользователь, который может потенциально владеть пусть и меньшим объёмом, но существенно более конфиденциальной информацией.

В рамках анализа осведомлённости пользователей в конфиденциальной информации рассмотрим величину $\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},u_{l_2},t_k)$:

$$\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},u_{l_2},t_k)=|\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},t_k)-\mathcal{A}_{\text{MAC\_Pot}}(u_{l_2},t_k)|. \tag{4}$$

Лемма 1. Величина (4) неотрицательна, удовлетворяет свойствам симметричности и неравенства треугольника, является частным случаем расстояния Хемминга.

Доказательство. Неотрицательность ($\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},u_{l_2},t_k)\geqslant 0$), симметричность ($\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},u_{l_2},t_k)=\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_2},u_{l_1},t_k)$) и выполнение неравенства треугольника

$$\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},u_{l_2},t_k)\leqslant\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},u_{l_3},t_k)+\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_3},u_{l_2},t_k)$$

очевидны по свойствам операции $|a-b|$. Подставляя в соотношение (4) выражения для величин $\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},t_k)$, $\mathcal{A}_{\text{MAC\_Pot}}(u_{l_2},t_k)$ и производя несложные преобразования, получаем

$$\Delta\mathcal{A}_{\text{MAC\_Pot}}(u_{l_1},u_{l_2},t_k)=\frac{1}{V(O,t_k)}\sum_{n=1}^{N_{t_k}}V(o_n,t_k)f_{\text{conf}}(o_n,t_k)|a_{l_1n}(t_k)-a_{l_2n}(t_k)|.$$

Нетрудно видеть, что сумма в этой формуле является известным расстоянием Хемминга [10] между двоичными векторами прав доступа пользователей u_{l_1} и u_{l_2}, образуемыми соответствующими строками матрицы доступа $\mathbf{A}(t_k)$, координаты которых «взвешены» произведениями параметров $V(o_n,t_k)$ и $f_{\text{conf}}(o_n,t_k)$ объектов доступа. ■

Отметим, что аналогичными свойствами метрики обладает величина $\mathcal{A}_{\text{DAC\_Pot}}(u_l,t_k)$.

Особенности метрики $\mathcal{A}_{\text{MAC\_Pot}}(u_l,t_k)$ иллюстрируются следующим примером.

Пример 1. На рис. 3 приведены расчёты $\mathcal{A}_{\text{MAC\_Pot}}(u_l,t_k)$ для шести объектов доступа с различными параметрами объёма и конфиденциальности информации и информативности объектов и девяти пользователей с различными уровнями доверия в системе с мандатным управлением доступом. Для перевода порядково-вербальных характеристик конфиденциальности в числовые использована следующая эвристика: «высокая степень конфиденциальности» — $f_{\text{conf}}(o_n,t_k)=1$, «средняя» — $f_{\text{conf}}(o_n,t_k)=0{,}809$, «низкая» — $f_{\text{conf}}(o_n,t_k)=0{,}5$.

Рис. 3. Пример потенциальной осведомлённости пользователей при мандатном управлении доступом

Как видно из рис. 3, при наличии прав доступа ко всем объектам, содержащим конфиденциальную информацию, пользователь (на рис. 3 пользователь u_1) характеризуется наивысшей степенью потенциальной осведомлённости 100 %. Заметим, что по правилам мандатного доступа для 100 %-й потенциальной осведомлённости пользователь должен иметь наивысший уровень доверия (на рис. 3 — «высокий»). При этом если с учётом функциональных обязанностей или других соображений пользователь с наивысшим уровнем доверия имеет по матрице доступа права доступа только к некоторым объектам, скажем, только к объектам со степенью конфиденциальности «высокая», то потенциальная осведомлённость такого пользователя может характеризоваться невысокой величиной из-за невысокого объёма информации или информативности соответствующих объектов. Например, пользователь u_2 также с наивысшим уровнем доверия, обладая по матрице доступа правом доступа только к объекту с самым высоким уровнем конфиденциальности (o_3), характеризуется всего лишь 20 %-й величиной потенциальной осведомлённости ввиду незначительного объёма информации соответствующего объекта. Следует, однако, заметить, что данный уровень потенциальной осведомлённости в 20 % является достаточно существенным относительно величины потенциальной осведомлённости пользователей, обладающих правами доступа к другим объектам с существенно большими объёмами информации, но с меньшими уровнями конфиденциальности и информативности.

Чувствительность метрики потенциальной осведомлённости к объёму и уровню конфиденциальности объектов доступа также иллюстрируется показателями $\mathcal{A}_{\text{MAC\_Pot}}(u_l, t_k)$ для пользователей с одинаковым уровнем доверия, но с различными фактическими правами доступа к объектам (по матрице доступа) — пользователи u_3 и u_4, пользователи u_5, u_6, u_7, u_8, u_9.

Анализ поведения метрики $\mathcal{A}_{\text{MAC\_Pot}}(u_l, t_k)$ по приведённым на рис. 3 расчётам позволяет сделать вывод о её соответствии интуитивным представлениям о потенциальной осведомлённости пользователей и, следовательно, о возможности её применения в прикладных системах.

4. Потенциальная осведомлённость пользователей в конфиденциальной информации в системах тематико-иерархического управления доступом

Тематико-иерархическое управление доступом [3, 11, 12] осуществляется на основе сравнения тематических меток (индексов) объектов доступа и тематических меток (полномочий) субъектов доступа (пользователей). В качестве тематических меток объектов и субъектов доступа используются тематические мультирубрики $\mathcal{T}_i^{(m)} = \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_I}\}$, которые представляют собой наборы рубрик иерархического тематического классификатора $T_{TH} = \{r, \tau_1, \tau_2, \ldots, \tau_K\}$ (иерархического тематического рубрикатора), представляющего собой множество рубрик-тематик $\tau_{i_m} \in T_{TH}$, на котором задано отношение частичного порядка, выражаемое графом вида «корневое дерево» (r — корень дерева) [13]. Самым известным примером такого иерархического тематического рубрикатора является универсальная десятичная классификация (УДК), применяемая в библиотечной сфере для систематизации произведений науки, литературы и искусства, периодической печати, различных видов документов и организации картотек. Мультирубрики $\mathcal{T}_i^{(m)} = \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_I}\}$ включают наборы элементов τ_{i_m}, которые не находятся между собой в подчинённости по иерархическому рубрикатору, и никакие совокупности (подмножества) элементов мультирубрики не содержат полные совокупности элементов-сыновей каких-либо элементов-отцов иерархического рубрикатора.

Пример 2. На рис. 4 приведён пример корневого дерева иерархического тематического рубрикатора и наборы его рубрик, составляющих мультирубрики. Заметим, что полустепень исхода нелистовых вершин графа не может быть меньше двух, поскольку делить рубрику-тематику на подрубрики-подтематики имеет смысл только тогда, когда число подрубрик не меньше двух.

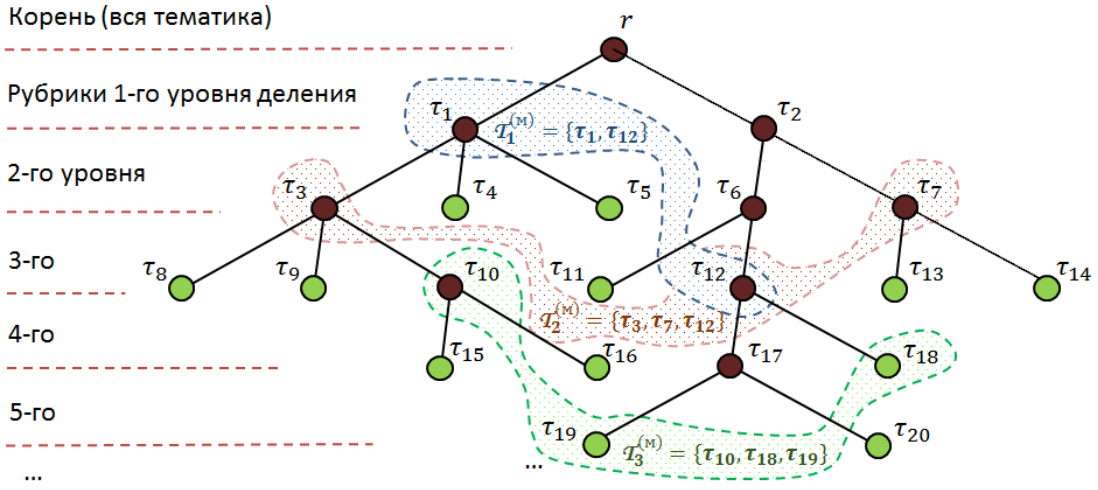

Рис. 4. Пример иерархического тематического рубрикатора и мультирубрик на нём, являющихся тематическими метками объектов доступа и тематическими полномочиями субъектов доступа (пользователей)

Множество мультирубрик $\mathcal{T}_i^{(m)} \in T^{(m)}$ является решёточно-упорядоченным [3, 11, 13] относительно специальной операции доминирования мультирубрик (тематика одной мультирубрики шире тематики другой, т. е. с учётом подчинённости полностью охватывает тематику другой мультирубрики, или мультирубрики несравнимы по отношению «шире — уже», в том числе когда их рубрики-тематики частично пересека-

ются). Если мультирубрика $\mathcal{T}_i^{(m)} = \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_I}\}$ доминирует над мультирубрикой $\mathcal{T}_j^{(m)} = \{\tau_{j_1}, \tau_{j_2}, \ldots, \tau_{j_J}\}$, то для любой рубрики $\tau_{j_m} \in \mathcal{T}_j^{(m)}$ найдётся рубрика $\tau_{i_n} \in \mathcal{T}_i^{(m)}$, которая совпадает с ней или является старшей по корневому дереву иерархического рубрикатора:

$$\mathcal{T}_i^{(m)} \geqslant T_j^{(m)} \Rightarrow \forall \tau_{j_m} \in \mathcal{T}_j^{(m)} \exists \tau_{i_n} \in \mathcal{T}_i^{(m)} \left(\tau_{i_n} \geqslant_m \tau_{j_m}\right),$$

где $\geqslant_m$ — знак тематического доминирования мультирубрик (тематика шире, т. е. полностью охватывает тематику другой мультирубрики); $\geqslant$ — знак подчинённости или эквивалентности (совпадения) рубрик по корневому дереву иерархического рубрикатора. Таким образом, отношение доминирования мультирубрик как подмножеств рубрик не является отношением теоретико-множественного включения $\subseteq$.

На рис. 4 мультирубрики $\mathcal{T}_1^{(m)} = \{\tau_1, \tau_{12}\}$ и $\mathcal{T}_2^{(m)} = \{\tau_3, \tau_7, \tau_{12}\}$ доминируют над мультирубрикой $\mathcal{T}_3^{(m)} = \{\tau_{10}, \tau_{18}, \tau_{19}\}$: $\mathcal{T}_1^{(m)} \geqslant_m \mathcal{T}_3^{(m)}$, $\mathcal{T}_2^{(m)} \geqslant_m \mathcal{T}_3^{(m)}$, но между собой несравнимы, несмотря на существенное пересечение тематик: $\mathcal{T}_1^{(m)} \geqslant\leqslant_m \mathcal{T}_2^{(m)}$, где знак $\geqslant\leqslant_m$ означает несравнимость мультирубрик.

Как и в мандатных системах, пользователь имеет право чтения объекта по правилу, аналогичному правилу NRU, т. е. когда мультирубрика пользователя (набор разрешённых ему рубрик) доминирует над мультирубрикой объекта доступа.

Аспект возможного изменения состояния системы тематико-иерархического управления доступом будем отображать временно́й зависимостью мультирубрик субъектов и объектов доступа: $\mathcal{T}^{(m)}(u_l, t_k)$, $\mathcal{T}^{(m)}(o_n, t_k)$.

В результате, исходя из положения 4, потенциальная осведомлённость пользователя u_l в момент времени t_k в конфиденциальной информации при тематико-иерархическом управлении доступом $\mathcal{A}_{\text{THA\_Pot}}(u_l, t_k)$ определяется на основе соотношения

$$\mathcal{A}_{\text{THA\_Pot}}(u_l, t_k) = \frac{1}{V(O, t_k)} \sum_{n=1}^{N_{t_k}} V(o_n, t_k)\, \delta(\mathcal{T}^{(m)}(u_l, t_k), \mathcal{T}^{(m)}(o_n, t_k)),$$

где

$$\delta(\mathcal{T}^{(m)}(u_l, t_k), \mathcal{T}^{(m)}(o_n, t_k)) = \begin{cases} 1, & \text{если } \mathcal{T}^{(m)}(u_l, t_k) \geqslant_m \mathcal{T}^{(m)}(o_n, t_k); \\ 0, & \text{если } \left(\mathcal{T}^{(m)}(u_l, t_k) <_m \mathcal{T}^{(m)}(o_n, t_k)\right) \vee \\ & \vee \left(\mathcal{T}^{(m)}(u_l, t_k) \geqslant\leqslant_m \mathcal{T}^{(m)}(o_n, t_k)\right). \end{cases}$$

Нетрудно видеть, что величина $\mathcal{A}_{\text{THA\_Pot}}(u_l, t_k)$ обладает свойствами метрики.

Тематико-иерархическое управление доступом предоставляет дополнительные возможности по анализу потенциальной осведомлённости пользователей в конфиденциальной информации. Поскольку конфиденциальная информация объектов доступа проиндексирована (размечена) тематическими мультирубриками, то появляется возможность анализа осведомлённости пользователей не просто в целом по конфиденциальной информации, обрабатываемой в компьютерной системе, но и по отдельным её тематикам, что может быть важным в тех или иных аспектах информационной безопасности. В частности, по любой простой тематике (одна рубрика-тематика τ_i является частным случаем мультирубрики: $\mathcal{T}_i^{(m)} = \{\tau_i\}$) или по комплексу рубрик-тематик (тематическая мультирубрика $\mathcal{T}_i^{(m)} = \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_I}\}$) можно определять величину потенциальной осведомлённости $\mathcal{A}_{\text{THA\_Pot}}(u_l, \mathcal{T}_i^{(m)}, t_k)$ пользователя u_l в момент времени t_k:

$$\mathcal{A}_{\text{THA\_Pot}}(u_l, \mathcal{T}_i^{(m)}, t_k) =$$
$$= \frac{1}{V(O, \mathcal{T}_i^{(m)}, t_k)} \sum_{n=1}^{N_{t_k}} V(o_n, t_k)\, \delta\left(\mathcal{T}^{(m)}(u_l, t_k), \mathcal{T}^{(m)}(o_n, t_k)\right) \delta\left(\mathcal{T}^{(m)}(o_n, t_k), \mathcal{T}_i^{(m)}\right),$$

где $V(O, \mathcal{T}_i^{(m)}, t_k)$ — объём всей конфиденциальной информации, тематика которой характеризуется в момент времени t_k мультирубрикой $\mathcal{T}_i^{(m)} = \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_I}\}$:

$$V(O, \mathcal{T}_i^{(m)}, t_k) = \sum_{n=1}^{N_{t_k}} V(o_n, t_k)\, \delta\left(\mathcal{T}^{(m)}(o_n, t_k), \mathcal{T}_i^{(m)}\right).$$

5. Потенциальная осведомлённость пользователей в конфиденциальной информации в системах ролевого управления доступом

Ролевое управление доступом [14, 15] основывается на понятии роли (точнее — ролевого субъекта доступа) $\rho \in R$ (R — множество ролей), аналогом которой в некомпьютерной сфере является понятие «должность» в организационно-штатной структуре предприятий (организаций) [3]. Пользователи получают права доступа к объектам не напрямую, а посредством разрешения им работы в одной или нескольких ролях. Роли наделяются функционально обоснованной (для соответствующей должности) совокупностью прав доступа к объектам. Функционирование компьютерной системы разбивается на сеансы, работу в которых пользователи начинают с авторизации в одной или нескольких из разрешённых им в системе ролях и осуществляют доступы к объектам по правам соответствующих ролей.

Ролевое управление доступом в системах с большим количеством пользователей и объектов доступа позволяет существенно сократить количество назначений доступа для наделения пользователей необходимыми им для работы правами и в настоящее время широко применяется в корпоративных сетях.

Базовая модель ролевого управления доступом (RBAC, Role Based Access Control), помимо множеств пользователей $u \in U$, объектов доступа $o \in O$ и ролей $\rho \in R$, включает отображение множества U на множество R, наиболее простым и естественным способом задания которого является двоичная $(L \times M)$-матрица W «Пользователи — Роли» (L — количество пользователей, M — количество ролей):

$$W = \begin{array}{c} \\ u_1 \\ u_2 \\ \ldots \\ u_L \end{array} \begin{array}{c} \begin{array}{cccc} \rho_1 & \rho_2 & \ldots & \rho_M \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & 0 & \ldots & 0 \end{pmatrix} \end{array}. \tag{5}$$

Ненулевые элементы w_{lm} матрицы W означают разрешение работы l-го пользователя в m-й роли.

Права ролевых субъектов доступа часто именуются *полномочиями*, поскольку в расширениях ролевой модели и её практических приложениях включают права запуска (исполнения) определённых функционально-технологических процедур. Упрощённым прототипом таких процедур являются элементарные виды доступа к объектам — «Чтение» (Read), «Запись» (Write), «Выполнение» (Exec).

Наделение ролей полномочиями (в упрощённой трактовке — правами доступа к объектам) может осуществляться на основе одного из двух исходных принципов управления доступом — дискреционного или мандатного (мандатно-ролевые модели) [2, 3], поэтому ролевая модель управления доступом является некой надстройкой над ними. При этом, как отмечалось в п. 3, и при мандатном принципе фактические (итоговые) права конкретных субъектов устанавливаются матрицей доступа, только в данном случае для ролевых субъектов доступа в форме матрицы A_R «Роли — Объекты».

Ввиду рассмотрения доступов только вида «Чтение», матрицу A_R можно считать двоичной, ненулевые элементы $a_{R_{mn}}$ которой отражают разрешение чтения ролевым субъектом ρ_m объекта o_n:

$$A_R = \begin{array}{c} \\ \rho_1 \\ \rho_2 \\ \dots \\ \rho_M \end{array} \begin{array}{c} \begin{array}{cccc} o_1 & o_2 & \dots & o_N \end{array} \\ \begin{pmatrix} 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & 1 \end{pmatrix} \end{array}.$$

С учётом иерархичности отношений в системах организационно-штатной структуры предприятий в наиболее распространённых на практике расширениях базовой ролевой модели на множестве ролей устанавливаются отношения частичного порядка, отображаемые графом типа «корневое дерево».

Пример 3. На рис. 5 приведён пример иерархической организации системы ролей, структура которой представляет граф вида «корневое дерево», полустепень исхода нелистовых вершин в котором может быть равна 1, т. е. старшая роль (должность) может иметь в подчинении всего одну роль (должность).

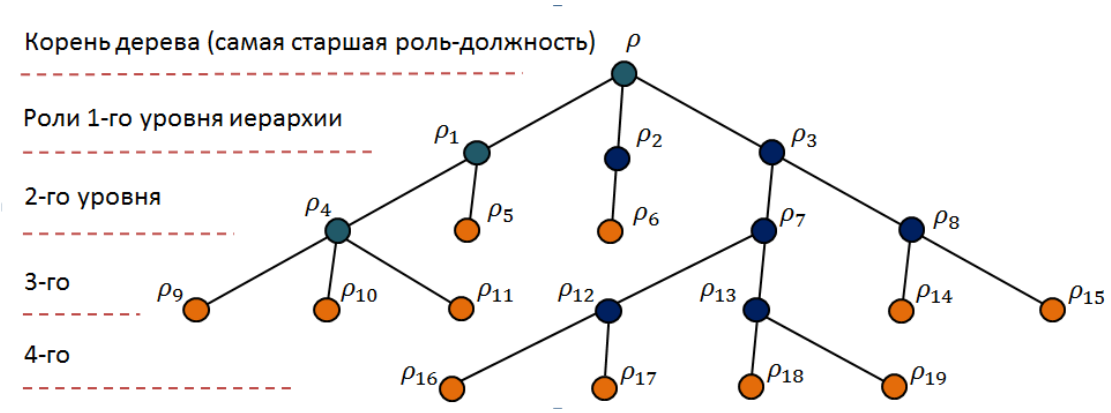

Рис. 5. Пример иерархической системы ролей

Права доступа к объектам и процедурам старших ролей включают права доступа всех подчинённых ролей. Это позволяет ещё больше снизить количество назначений доступа, поскольку полномочия старших ролей могут автоматически складываться только из полномочий подчинённых («листовой» принцип [3]) или из полномочий подчинённых ролей с добавлением их старшим ролям дополнительных («номенклатурных») полномочий (не строго «листовой» иерархически охватный принцип [3]).

Таким образом, в системах с иерархической организацией системы ролей если пользователь в каком-либо сеансе авторизован в определённой роли ρ, то совокупность его

прав доступа охватывает права доступа всех ролей $\rho_i < \rho$, подчинённых по иерархии роли ρ (знак $<$ означает подчинённость по ветви корневого дерева). Определение итоговых прав доступа пользователя в этом случае можно осуществлять на основе матрицы доступа A_R «Роли — Объекты» и матрицы связности (достижимости) $H^{(s)}$ корневого дерева иерархической системы ролей:

$$H^{(s)} = H + H^2 + H^3 + \ldots + H^d,$$

где H — матрица смежности корневого дерева иерархической системы ролей; d — высота дерева, т. е. максимальная длина пути от листовой вершины до корня.

Как и ранее, временно́й аспект функционирования ролевой системы управления доступом будем отображать зависимостью от времени матриц $W(t_k)$, $A_R(t_k)$, $H(t_k)$ и $H^{(s)}(t_k)$.

Ненулевые элементы $h_{ij}^{(s)}(t_k)$ означают подчинённость j-й вершины по иерархии корневого дерева i-й вершине в момент времени t_k — непосредственной (длина пути между вершинами равна 1) либо подчинённости по ветви дерева (длина пути больше 1). Диагональные элементы матриц смежности и связности равны нулю: $h_{ii}(t_k) = 0$, $h_{ii}^{(s)}(t_k) = 0$.

Тогда итоговые права доступа ролей (по непосредственным назначениям и правам доступа подчинённых ролей) задаются элементами матрицы $A_R^{(s)}(t_k)$, которая является произведением матрицы $A_R(t_k)$ на матрицу связности ролей $H^{(s)}(t_k)$, дополненную единичными значениями по диагональным элементам:

$$A_R^{(s)}(t_k) = \left(\left((A_R(t_k))^{\mathrm{T}} \left(H^{(s)}(t_k) + I \right) \right)^{\mathrm{T}} \right)_{|1},$$

где I — единичная матрица; $(x)_{|1} = \begin{cases} 1, & \text{если } x \geqslant 1; \\ 0, & \text{если } x < 1. \end{cases}$

Исходя из этого, итоговые права доступа пользователей в виде матрицы доступа «Пользователи — Объекты» $A_{RW}^{(s)}(t_k)$ в иерархической системе ролей можно определить на основе произведения матрицы $W(t_k)$ на матрицу $A_R^{(s)}(t_k)$:

$$A_{RW}^{(s)}(t_k) = \left(W(t_k)\, A_R^{(s)}(t_k) \right)_{|1}.$$

В результате, исходя из положения 4, потенциальная осведомлённость пользователя u_l в момент времени t_k в конфиденциальной информации при ролевом управлении доступом с иерархически организованной системой ролей $\mathcal{A}_{\mathrm{RBAC\_H\_Pot}}(u_l, t_k)$ определяется на основе следующего соотношения:

$$\mathcal{A}_{\mathrm{RBAC\_H\_Pot}}(u_l, t_k) = \frac{1}{V(O, t_k)} \sum_{n=1}^{N_{t_k}} V(o_n, t_k)\, a_{RW}^{(s)}(t_k).$$

Нетрудно показать, что величина $A_{\mathrm{RBAC\_H\_Pot}}(u_l, t_k)$, как и аналогичные величины при дискреционном, мандатном и тематико-иерархическом управлении доступом, удовлетворяет требованиям метрики.

Заключение

Использование метрик $\mathcal{A}_{\text{DAC\_Pot}}(u_l, t_k)$, $\mathcal{A}_{\text{MAC\_Pot}}(u_l, t_k)$, $\mathcal{A}_{\text{THA\_Pot}}(u_l, t_k)$ и $\mathcal{A}_{\text{RBAC\_H\_Pot}}(u_l, t_k)$ может являться основой специального программного инструментария не только для анализа состояния существующей (функционирующей) системы прав доступа пользователей, но и средством проектирования системы управления доступом пользователей к конфиденциальной информации. Администраторы компьютерных систем, выстраивая систему прав доступа или анализируя её текущее состояние, могут видеть значения и тенденции в потенциальной осведомлённости тех или иных пользователей и принимать на этой основе решения в контексте обеспечения информационной безопасности.

Параметры объёма информации объектов доступа $Q(o_n, t_k)$ в словах, как правило, автоматически определяются и включаются в метаданные текстовых файлов в современных офисных системах работы с документами и системах электронного документооборота.

Порядково-вербальные характеристики конфиденциальности информации $f_{\text{conf}}(o_n, t_k)$ в мандатных системах управления доступом устанавливаются на основе процедур, закрепляемых нормативными регламентациями делопроизводства, и также, как правило, включаются в метаданные файлов офисных систем и систем электронного документооборота. Подобные же по смыслу и содержанию процедуры могут быть регламентированы для определения характеристик информативности объектов доступа $\theta(o_n, t_k)$. К примеру, уровень информативности документа устанавливает его исполнитель и/или выделенный сотрудник (аналитик).

Отдельно следует отметить проблемы построения (обоснования) эвристик перевода (отображения) порядково-вербальных характеристик конфиденциальности в числовую шкалу $[0, 1]$ в системах с мандатным управлением доступа.

Насколько известно, строгих и корректных процедур такого рода преобразований не существует. Объективной причиной этого является различная информативность порядковых и количественных шкал. Показания количественной шкалы $[0, 1]$ помимо порядка отображают (воспроизводят) расстояния между оцениваемыми объектами по соответствующим характеристикам. Таким образом, переводя оценки из порядковой шкалы в количественную, необходимо сформировать дополнительную информацию о расстояниях по соответствующему свойству/показателю между оцениваемыми объектами.

Обратные преобразования (из числовой шкалы $[0, 1]$ в порядковую) осуществляют на основе интервального принципа, разбивая диапазон $[0, 1]$ на равные или неравные интервалы, все числовые элементы из которых объединяются в соответствующие отсчёты (показатели) порядковой шкалы. При этом происходит потеря информации о расстояниях и применяются опять-таки определённые эвристики. Наиболее часто такие эвристики основываются на гипотезе о равных интервалах, согласно которой диапазон шкалы $[0, 1]$ разбивается на одинаковые по величине интервалы, каждый из которых соответствует своему отсчёту на шкале порядка.

Другим принципом разбиения шкалы $[0, 1]$ на порядковые интервалы может быть принцип «золотого сечения» («золотой пропорции»), согласно которому меньшая часть целого (следующий, более высокий отсчёт порядка и соответствующий интервал) относится к большей части целого (первый отсчёт порядка) так же, как большая часть относится к целому. Тогда двум отсчётам в порядковой шкале «низкий — высокий» соответствуют два интервала на шкале $[0, 1]$ — (0, 0,618) и (0,618, 1). Одним из вариантов разбиения шкалы $[0, 1]$ на три порядковых интервала по принципу «золотой

пропорции» может быть разбиение «верхнего» интервала (0,618, 1) также по принципу «золотой пропорции». В результате диапазон шкалы $[0, 1]$ разбивается на три интервала — низкий (0, 0,618), средний (0,618, 0,854) и высокий (0,854, 1). Другим вариантом разбиения диапазона шкалы $[0, 1]$ на три порядковых интервала может быть добавление к отрезку $[0, 1]$ третьего отрезка, который соотносится с меньшим интервалом двухуровневого разбиения, т. е. с интервалом (0,618, 1), по принципу «золотой пропорции». Длина такого отрезка равна 0,235999937. Нормируя увеличенный диапазон шкалы (0, 1,235999937) на единичную длину, получаем следующие три интервала — низкий (0, 0,499994), средний (0,499994, 0,809), высокий (0,809, 1).

Исходя из интервального принципа, эвристики преобразования показателей из порядково-вербальных шкал в шкалу $[0, 1]$ заключаются в присвоении всем объектам оценки, характеризующимся определённым показателем шкалы порядка, такого числового значения на шкале $[0, 1]$, которое соответствует определённой характеристике соответствующего интервала (обычно левой/правой границе или середине интервала). Соответственно различия в конфиденциальности в шкале $[0, 1]$ тех или иных объектов доступа внутри интервалов нивелируются.

Таким образом, программно-техническая составляющая систем анализа потенциальной осведомлённости пользователей на основе представленных метрик с учётом отмеченных эвристик может быть реализована в современных офисных системах работы с документами и системах электронного документооборота.

ЛИТЕРАТУРА

1. *Ушаков Д. Н.* Большой толковый словарь русского языка. М.: Дом Славянской кн., 2008. 959 с.
2. *Девянин П. Д.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками. М.: Горячая линия-Телеком, 2020. 352 с.
3. *Гайдамакин Н. А.* Разграничение доступа к информации в компьютерных системах. Екатеринбург: Изд-во Урал. ун-та, 2003. 328 с.
4. *Грушо А. А., Применко Е. А., Тимонина Е. Е.* Теоретические основы компьютерной безопасности. М.: Издательский центр «Академия», 2009. 272 с.
5. *Shannon C. E.* A mathematical theory of communication // Bell System Technical J. 1948. V. 27. P. 379–423.
6. Р 50.1.053-2005. Информационные технологии. Основные термины и определения в области технической защиты информации. М.: Стандартинформ, 2005. 11 с.
7. ГОСТ Р ИСО/МЭК 27000-2021. Информационные технологии. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология. М.: Стандартинформ, 2021. 21 с.
8. Федеральный закон «Об информации, информационных технологиях и о защите информации» от 27.07.2006 № 149-ФЗ (с изм.). Собрание законодательства Российской Федерации. 2006. № 31. Ст. 3448.
9. ГОСТ Р 58545-2019. Менеджмент знаний. Руководящие указания по сбору, классификации, маркировке и обработке информации. М.: Стандартинформ, 2019. 34 с.
10. *Деза Е. И., Деза М. М.* Энциклопедический словарь расстояний. М.: Наука, 2008. 446 с.
11. *Гайдамакин Н. А.* Модель тематического разграничения доступа к информации при иерархической структуре классификатора в автоматизированных системах управления // Автоматика и телемеханика. 2003. № 3. С. 177–189.
12. *Гайдамакин Н. А.* Многоуровневое тематико-иерархическое управление доступом (MLTHS-система) // Прикладная дискретная математика. 2018. № 39. С. 42–57.

13. *Гайдамакин Н. А., Баранский В. А.* Алгебра мультирубрик на корневых деревьях иерархических тематических классификаторов // Сиб. электрон. матем. изв. 2017. Т. 14. С. 1030–1040.
14. *Ferrariolo D. F. and Kuhn D. R.* Role Based Access Control // 15th National Computer Secure Conf. Baltimore, 1992. P. 554–563.
15. *Sundhu R., Coyne E. J., Feinstein H. L., and Youman C. E.* Role-Based Access Control models // IEEE Computer. 1996. V. 29. No. 2. P. 38–47.

REFERENCES

1. *Ushakov D. N.* Bol'shoy tolkovyy slovar' russkogo yazyka [Great Dictionary of Russian Language]. Moscow, Dom Slavyanskoy kn., 2008. 959 p. (in Russian)
2. *Devyanin P. D.* Modeli bezopasnosti komp'yuternykh sistem. Upravlenie dostupom i informatsionnymi potokami [Security Models of Computer Systems. Access and Information Flow Management]. Moscow, Goryachaya liniya-Telekom, 2020. 352 p. (in Russian)
3. *Gaydamakin N. A.* Razgranichenie dostupa k informatsii v komp'yuternykh sistemakh [Differentiation of Access to Information in Computer Systems]. Ekaterinburg, UrFU Publ., 2003. 328 p. (in Russian)
4. *Grusho A. A., Primenko E. A., and Timonina E. E.* Teoreticheskie osnovy komp'yuternoy bezopasnosti. [Theoretical Foundations of Computer Security]. Moscow, Publishing Center "Akademiya", 2009. 272 p. (in Russian)
5. *Shannon C. E.* A mathematical theory of communication. Bell System Technical J., 1948, vol. 27, pp. 379–423.
6. R 50.1.053-2005. Informatsionnye tekhnologii. Osnovnye terminy i opredeleniya v oblasti tekhnicheskoy zashchity informatsii [R 50.1.053-2005. Information Technology. Basic Terms and Definitions in the Field of Technical Information Protection]. Moscow, Standartinform Publ., 2005. 11 p. (in Russian)
7. GOST R ISO/MEK 27000-2021. Informatsionnye tekhnologii. Metody i sredstva obespecheniya bezopasnosti. Sistemy menedzhmenta informatsionnoy bezopasnosti. Obshchiy obzor i terminologiya [GOST R ISO/IEC 27000-2021. Information Technology. Methods and Means of Ensuring Security. Information Security Management Systems. General Overview and Terminology]. Moscow, Standartinform Publ., 2021. 24 p. (in Russian)
8. Federal'nyy zakon "Ob informatsii, informatsionnykh tekhnologiyakh i o zashchite informatsii" [Federal Law "On information, information technologies and on information protection"]. 27.07.2006, no. 149-FZ. Collection of Legislation of the Russian Federation, Moscow, 2006, no. 31, st. 3448. (in Russian)
9. GOST R 58545-2019. Menedzhment znaniy. Rukovodyashchie ukazaniya po sboru, klassifikatsii, markirovke i obrabotke informatsii [GOST R 58545-2019. Knowledge management. Guidelines for the collection, classification, labeling and processing of information]. Moscow, Standartinform Publ., 2019. 34 p. (in Russian)
10. *Deza E. I. and Deza M. M.* Entsiklopedicheskiy slovar' rasstoyaniy [Encyclopedic Dictionary of Distances]. Moscow, Nauka, 2008. 446 p. (in Russian)
11. *Gaydamakin N. A.* A model of thematic differentiation of access to information for the hierarchical classifier in automatic control systems. Autom. Remote Control, 2003, vol. 64, no. 3, pp. 505–516.
12. *Gaydamakin N. A.* Mnogourovnevoe tematiko-ierarkhicheskoe upravlenie dostupom (MLTHS-sistema) [Multilevel thematic-hierarchical access control (MLTHS-system)]. Prikladnaya Diskretnaya Matematika, 2018, no. 39, pp. 42–57. (in Russian)
13. *Gaydamakin N. A., Baranskiy V. A.* Algebra multirubrik na kornevykh derevyakh iyerarkhicheskikh tematicheskikh klassifikatorov [Algebra of multirubric on root trees

of hierarchical thematic classifiers]. Sib. Èlektron. Mat. Izv., 2017, vol. 14, pp. 1030–1040. (in Russian)

14. *Ferrariolo D. F. and Kuhn D. R.* Role Based Access Control. 15th National Computer Secure Conf., Baltimore, 1992, pp. 554–563.
15. *Sundhu R., Coyne E. J., Feinstein H. L., and Youman C. E.* Role-Based Access Control models. IEEE Computer, 1996, vol. 29, no. 2, pp. 38–47.

УДК 004.056 DOI 10.17223/20710410/61/6

USING X86 MODE SWITCHING FOR PROGRAM CODE PROTECTION

R. K. Lebedev

Novosibirsk State University, Novosibirsk, Russia

E-mail: n0n3m4@gmail.com

A novel program code obfuscation approach involving the x86 mode switching is proposed in the paper. The details and existing applications of x86 mode switching are reviewed, as well as the possible consequences of using this switching to the reverse engineering tools. Based on this approach, a few specific methods are proposed and evaluated against the most popular reverse engineering tools of various purposes, including disassemblers, decompilers, binary instrumentation and symbolic execution tools. A method of seamless integration of these machine code level obfuscations to the C, C++ and possibly other compilers is also proposed.

Keywords: *code protection, reverse engineering, obfuscation, x86 mode switching, disassembly, decompilation, symbolic execution.*

ИСПОЛЬЗОВАНИЕ ПЕРЕКЛЮЧЕНИЯ РЕЖИМОВ X86 ДЛЯ ЗАЩИТЫ ПРОГРАММНОГО КОДА

Р. К. Лебедев

Новосибирский государственный университет, г. Новосибирск, Россия

Предложен новый подход к обфускации программного кода с использованием переключения режимов x86. Рассмотрены детали и существующие применения переключения режимов x86, а также возможные последствия его использования для инструментов реверс-инжиниринга. На основе этого подхода предложено несколько методов, проверенных против различных наиболее популярных инструментов реверс-инжиниринга, включая дизассемблеры, декомпиляторы, инструменты бинарной инструментации и символьного исполнения. Предложен метод бесшовной интеграции обфускаций на уровне машинного кода в компиляторы C, C++ и, возможно, других языков.

Ключевые слова: *защита кода, реверс-инжиниринг, обфускация, переключение режимов x86, дизассемблирование, декомпиляция, символьное исполнение.*

1. Introduction

Program code protection is an important problem to solve currently due to the increasing spread of information technology in our life. Programs can be considered as important as hardware devices, so it is desirable for their authors to be able to leave some of the implementation details unknown to prevent intellectual property theft and software patent violations.

One of the commonly used approaches to the code protection is software obfuscation — the process of program code replacement with some equivalent, but less readable one. It's applicable both on source code and machine code levels. Although absolute code protection

(i.e., turning a program into a black box) is theoretically impossible in general [1], in practice, there are various methods that can greatly increase the time and skill required for reverse engineering, making it impractical.

Machine code level obfuscation is more practical for protecting the commercial software, because it usually comes in a compiled executable form (unless it's written in a language that cannot be compiled). Furthermore, compilation itself can be considered as a kind of code protection, for example IL2CPP AOT compiler of Unity engine makes games significantly harder to decompile compared to the original .NET bytecode, so there is even a project dedicated solely to revert this process [2].

Obfuscation methods can be divided into two types: generic and architecture-dependent.

Generic methods are those that can be performed at any level, including source code, and can be applied to some extent to any platform on which the program can run. This makes these methods particularly useful for cross-platform software development because there are at least two popular processor architectures on the market: ARM and x86. The aforementioned advantage is also significant for web applications written in JavaScript and WebAssembly.

Here are some examples of these generic obfuscation methods:

— Control flow graph flattening is an approach that aims at the branches and loops of the program, making the graph representation of the program "flat" and making it much more difficult to study. One of the most common ways to implement it is to split the program in basic blocks, which are branch or loop destinations, and place them as the clauses of the switch-like structure controlled by some state in the loop. Branching, looping and any other control flow altering is done by changing this state, effectively turning the program into a finite-state machine [3].
— Opaque predicates — insertion of some conditional branches whose condition is always constant, but is unclear during static analysis, thus being "opaque". A branch that is never taken can contain a fake code that may mislead the analyst: incorrect implementation of the block, unrelated function calls or just random bytes. This technique also makes control flow graph of the program more complex, independent of the fake code used [4].
— Constant obfuscation is the replacement of numeric constants and literals with some runtime-computed value, for example, a mathematical formula or string decryption routine. Prevents some simple static analysis methods like strings extraction [5].
— Instructions Substitution is the replacement of standard operations such as arithmetic or Boolean with some other instructions that produce the same result, but are written in a less readable form. A simple example of such transforms for some Boolean operations is the use of De Morgan's laws, and some compilers may actually use similar transforms for optimization [6].

While generic obfuscation methods are quite popular in the existing obfuscators, such as Obfuscator-LLVM and Tigress [7, 8], they have a significant downside: machine code remains completely usual because it is generated by the compiler from source or intermediate representation like LLVM IR. This means that any of the existing reverse engineering tools are still applicable: disassemblers, debuggers and symbolic execution tools are available to the attacker. The code may still be resilient to attacks on its own, but neutralising these tools at the lower level may assist in protecting the program even more.

Here architecture-dependent obfuscation comes useful: it is possible to generate some machine code that generic reverse engineering tools will process improperly or won't process

at all. This is possible because some processor instruction sets are extremely complex, and reverse engineering tools may focus only on the machine code constructions that compilers commonly use. As a variable-length CISC (complex instruction set computer) instruction set, x86 is an architecture, where these methods can be particularly efficient. Furthermore, x86-64 processors retain backward compatibility with 32-bit and 16-bit modes, which complicates things more.

Here are some examples of architecture-dependent obfuscation methods for x86:

— Self-modifying code is a code that modifies itself at runtime. This technique makes it possible to hide entire program logic from static analysis, since the executing code may differ tremendously from the code available in the program image. Packers and protectors, quite popular types of tools to do software protection, also use this approach [9]. This method is only applicable to the von Neumann architecture, and some operating systems may add additional data-code separation, making it impossible to be used.

— Instruction overlapping is a method that uses jumps to the middle of specially prepared program instructions. As x86 uses variable length instructions encoding and doesn't require program counter alignment, it is possible to effectively hide one instruction inside another: for example, in the case of a long (e.g. 64-bit) constant load, constant itself may contain some meaningful machine code. So jumping in the middle of instruction will execute this hidden instruction while it remains invisible to the disassembler, unless it is manually disassembled at the right start position [10]. Furthermore, it is possible to craft machine code that is impossible to disassemble correctly at all. This occurs because one byte is shared between two instructions, both of which are actually executed in the program [11].

— Rare instructions use is a method that involves replacing common instructions with their less common alternatives, e.g. instructions from x86 extensions which could be unsupported in disassemblers and decompilers. For example, constant loading may be implemented via AES-NI instructions instead of plain MOV call, that effectively prevents decompiler attempts to simplify the code and breaks symbolic execution in some tools [12].

The obfuscation method proposed in this paper is also architecture-dependent.

2. Basics of x86 mode switching

Modern x86 processors are able to switch between 64-bit and 32-bit execution modes during program runtime. This feature is particularly useful for running 32-bit programs on 64-bit operating systems, because kernel is always running in the native mode. A transition should happen at system calls to execute 64-bit kernel code in the kernel mode, while the reverse transition should happen to continue executing the program in the 32-bit mode. One of the famous uses of this approach is the WOW64 (Windows 32-bit on Windows 64-bit) subsystem of Microsoft Windows, which allows 32-bit applications to run on 64-bit operating system [13].

This switch is implemented by choosing a proper descriptor in x86 GDT (Global Descriptor Table), which is initialized and set by the operating system, its scheme is available in the Fig. 1.

The interesting bits of the GDT descriptor are L and D bits: when L bit is set, the code is considered to be 64-bit, and the D bit should be unset. D bit set means that the running code is 32-bit, while if both bits are unset, the code is considered to be running in a legacy 16-bit mode. In both Windows and Linux there are corresponding 32-bit and 64-bit user-mode descriptors that have indexes 4 and 6 in GDT, respectively [14, 15].

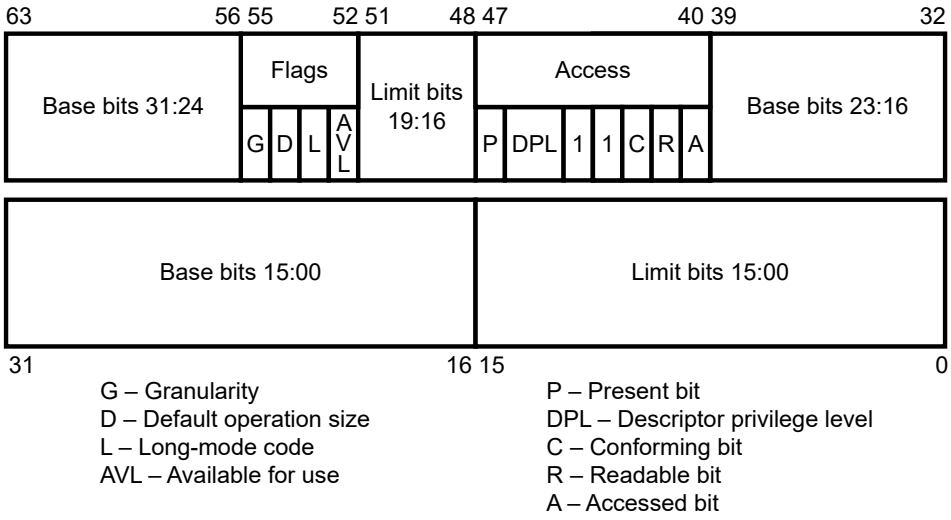

Fig. 1. Code-Segment Descriptor

Modern x86 operating systems are always running in the protected and long modes only. In these modes, a GDT descriptor can be chosen by loading a new segment selector value, that is stored in a Code Segment register (CS), to a proper value. A scheme of this selector is available in the Fig. 2.

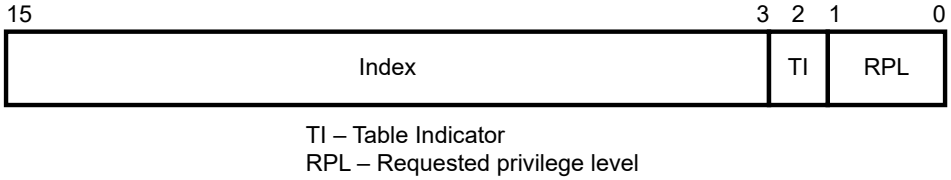

Fig. 2. Segment Selector

To choose another descriptor, one can just change an Index part of the segment selector leaving TI and RPL the same. While the values of TI and RPL can be retrieved from the running OS, according to the documentation for regular user-mode programs they should be TI = 0 (as GDT descriptor is selected, not LDT) and RPL = 3 (because user-mode programs are running in the "ring 3", the least privileged mode).

Therefore, to select a GDT descriptor by index, we should set CS according to the following simple formula:

$$\mathrm{CS} = \mathrm{Index} \cdot 2^3 + 3.$$

So we can compute the CS values for 32-bit and 64-bit execution modes on Windows and Linux using the respective GDT descriptor indexes 4 and 6:

$$\mathrm{CS}_{\text{32-bit}} = 0\text{x}23, \quad \mathrm{CS}_{\text{64-bit}} = 0\text{x}33.$$

The last thing required to switch between 32-bit and 64-bit modes on x86 is the ability to set the CS register. According to the x86 platform documentation, in user-mode this can be done by the following instructions only [16, ch. 3.4.3, vol. 3A]:

1) Far Jump and Far Call: These instructions act like regular JMP and CALL instructions, with the only difference: they not only change the instruction pointer, but also change the code segment. They are available in both 32-bit and 64-bit mode, but immediate values for the address are allowed in 32-bit mode only (EA cp and

9A cp opcodes for JMP and CALL respectively), in 64-bit mode it is required to store an address in memory first. Far Call also pushes the code segment on the stack in addition to the return address.

2) Far Return: This instruction is a variant of a regular RET instruction that can also change the code segment being complimentary to the Far Call instruction. It takes both the target address and the code segment from the stack upon return.
3) Interrupt Return: While this instruction is designed for exception and interrupt handling, in fact it is quite similar to the Far Return: it does the same thing, but also restores EFLAGS register from the stack upon return.

3. Applications of 32-bit x86 mode for obfuscation

3.1. General lack of support in the reverse engineering tools

Mode switching is not common in the regular programs, so there is little demand for supporting it in the reverse engineering tools.

For example, none of the two most popular disassemblers, IDA and Ghidra, can automatically detect x86 mode switching, so the user has to specify 32-bit regions manually. Furthermore, Ghidra is completely lacking mixed mode support [17], so a binary with such switches will have to be divided in parts to be analyzed, that is rather time consuming and inconvenient. IDA does have mixed mode support, which can be used by creating new segments in the program, but it is impossible to use the decompiler with these segments, because IDA requires a correct disassembler version (32-bit or 64-bit) to be launched to decompile respective binary: if the binary uses both modes, this is clearly impossible.

Therefore, just adding mode switching along with some important 32-bit code to the app alone may make it much harder or even impossible to analyze with the reverse engineering tools even if the attacker is aware of it being used in the program.

3.2. Machine code mismatch

Assuming that reverse engineering tools are complicated to use with the mixed mode binaries, one may try to disassemble any machine code as if it was 64-bit, this is also what IDA and Ghidra do by default. This usually works, because most encodings of the instructions remain unchanged in the 64-bit instruction set. Still, there are some exceptions that may be made dangerous to the reverse engineering tools, because these tools will not just fail to disassemble the code, but instead silently misunderstand it, possibly leaving the attacker with incorrect decompiled code.

One of these exceptions is the family of one byte 4* (hex) machine codes [16, ch. B.1.2, vol. 2D]:

— On 32-bit these machine codes are used for encoding the INC and DEC instructions, one variant per register (16 total).
— On 64-bit they have been changed to encode the REX prefix. This is not an instruction on its own, instead, it modifies the behavior of the instruction following it. Lower nibble of its encoding is used as a bitmask with the following bits:
    1) W — operand size override, makes operand size 64-bit;
    2) R — extension of the ModR/M reg field, allows instruction use additional registers introduced in 64-bit mode (R8-R15);
    3) X — extension of the SIB index field;
    4) B — extension of the ModR/M rm field or SIB base field, depending on the instruction.

If an instruction doesn't require a prefix, for example, it is a no-op, the prefix is silently ignored independent of the flags specified [16, ch. 2.2.1, vol. 2A].

This ambiguity may be utilized to create machine code that executes different depending on the mode it is executed in:

1) First, zero out some register, e.g. EAX by using XOR EAX, EAX. This instruction has the same machine code in both modes.
2) Place a 40 (hex) machine code. It will decode as INC EAX in 32-bit mode and as REX prefix otherwise. Here the difference appears: in 32-bit mode EAX will be set to one, while in 64-bit mode it still remains zero.
3) Place a conditional jump depending on the zero flag, e.g. JZ label. Both XOR and INC instructions affect zero flag, so it will be ZF=1 in 32-bit mode and ZF=0 otherwise, so the execution mode will define whether the jump is taken. REX prefix will not affect this instruction in 64-bit mode.

An exact machine code illustration is available in the Table 1.

Table 1

Disassembly ambiguity

32-bit disassembly	Machine code, hex	64-bit disassembly
XOR EAX, EAX	31 C0	XOR EAX, EAX
INC EAX	40	REX JZ 0x1337
JZ 0x1337	0F 84 37 13 00 00	

This jump may be utilized to mislead the reverse engineer by forwarding the reverse engineering tools to the wrong code, creating an enhanced machine code-level version of the opaque predicates protection method.

3.3. Deprecated instructions

Some of the 32-bit x86 instructions are interesting for obfuscation on their own, but were removed from the 64-bit instruction set. The ability to switch to 32-bit x86 mode provides a possibility to use these instructions in the modern 64-bit programs.

One of the notable removed instruction sets is the Intel BCD instruction set, which contains six instructions, including AAM and AAD. These instructions are used to correct the result of binary-coded decimals multiplication and division, respectively. As noted in [18], although they are designed for binary-coded decimals, any other non-10 base can also be provided in the machine code form of these instructions, including zero. Due to assembly source form only supporting decimal base for AAM and AAD instructions, reverse engineering tools may be expected to miss other bases support.

According to the documentation, these instructions have the microcode implementations shown in the Listings 1 and 2 [16, ch. 3.3, vol. 2A].

```
1 IF 64-Bit Mode
2     THEN
3         #UD;
4     ELSE
5         tempAL := AL;
6         AH := tempAL / imm8;
7         /* imm8 is the second byte of machine code 0xA by
      default for AAM mnemonic */
8         AL := tempAL MOD imm8;
9 FI;
```

Listing 1. AAM implementation

```
IF 64-Bit Mode
    THEN
        #UD;
    ELSE
        tempAL := AL;
        tempAH := AH;
        AL := (tempAL + (tempAH * imm8)) AND FFH;
        /* imm8 is the second byte of machine code 0xA by
    default for AAD mnemonic */
        AH := 0;
FI;
```

Listing 2. AAD implementation

Using this information, one may easily implement some basic transforms of AX register. One of the common register operations is zeroing, usually implemented as XOR EAX, EAX for EAX register. Here we propose another implementation, done with AAM, AAD and BSWAP use, it consists of the following steps:

1) AAM with base 1, which will set AL to zero (as anything MOD 1 is zero) and AH will get the original AL value;
2) AAD with base 0, which will set AH to zero and AL will be increased by AH multipled by zero, so it will keep its value;
3) BSWAP EAX, this is required because AAM and AAD instructions can only operate on the lowest 16-bit half of the EAX register, so we swap these halves and continue with another register half;
4) repeat steps 1-2 for another register half.

This implementation can also be used to zero other registers if needed, with an additional pair of XCHG instructions required to swap target register with EAX and back. An example implementation of this approach is available in the Listing 3.

```
/* Any register may be used here, or this part may be
   omitted if EAX is to be zeroed */
XCHG EBX, EAX
/* Notation AAM (AAD) N is not official, but is supported
   in some assemblers, N is used as imm8 value */
AAM 1
AAD 0
BSWAP EAX
AAM 1
AAD 0
XCHG EBX, EAX
```

Listing 3. Register zeroing using AAM and AAD

4. Implementation of the proposed obfuscation methods

4.1. Mode switch scope and limitations

As with regular pure 32-bit programs, a processor running a part of a program in the 32-bit mode is limited to only have 32-bit memory accesses and 32-bit registers, this puts significant limitations on the method applications, which we will review:

1) First of all, the code running should reside in the lower 32 bits of virtual memory. If executable is not built as a PIE binary (for Linux) or has no dynamicbase option set (for Windows), it is possible to set an executable base in the appropriate area. Furthermore, for Linux it is being done automatically, because the default base address for most Linux compilers on x86-64 is 0x400000, for Windows, though, it is 0x140000000, which is beyond 32-bit range and should be changed. For PIE executables this method can be still used as long as there is no need to protect the whole executable and application can map appropriate memory pages using mmap or VirtualAlloc calls.
2) Protected part of the code shouldn't do any memory accesses to the heap or stack. Both heap and stack are usually allocated above 4GB range, so they cannot be accessed with 32-bit pointers. However, this limitation may not apply to the static memory if it is allocated near executable code, e.g. in non-PIE Linux binaries, where data usually starts at 0x600000.
3) Protected part of the code shouldn't do 64-bit arithmetics or use newer instructions introduced in the 64-bit ISA. However, operations that are known to zero out the upper half of 64-bit registers are allowed because x86-64 has automatic zero expansion for operations involving 32-bit registers [16, ch. 3.4.1.1, vol. 1]. One of the notable exceptions is register zeroing, which is commonly implemented as XOR Exy, Exy instead of XOR Rxy, Rxy.
4) Protected part of the code shouldn't call any 64-bit functions and is recommended to not use system calls. While it is possible to do syscalls from 32-bit mode sometimes, for example, in Linux via x32 ABI, this may require changes in arguments and system call numbers.

While these limitations seem rather strict, they are not very different from the limitations put by virtualization — a method commonly used for protecting the most sensitive code parts [19]. Therefore, methods are still acceptable for math and cryptographic applications, like license checking.

4.2. Mode switch implementation

The most straightforward way to change execution mode is using Far Jump instruction considered in the Section 2. In 64-bit mode, it has to be used as indirect jump with address stored in memory and pointed by register, while in 32-bit mode it is also available as an immediate jump. During experiments it turned out that the stack pointer register (RSP) gets truncated to 32 bits regardless of stack usage in the code, so it can be safely used as a scratch register for jumping from 64-bit mode, as it gets corrupted anyway. Its value can be saved in the RBP register, which doesn't get truncated and is closely related to the stack, which shouldn't be used in 32-bit mode anyway to comply with the mode switch restrictions. Original RBP value can be stored on the stack, while 64-bit indirect jump location and code segment value may be placed anywhere in the executable file. An implementation of this method is available in the Listing 4.

```
1 PUSH RBP
2 MOV RBP, RSP
3 MOV RSP, offset bit32_data
4 JMP fword ptr [RSP]
5 bit32:
6 /* The following code is executed in 32-bit mode */
7 .code32
```

```
 8 XOR EAX, EAX
 9 /* Return to the 64-bit mode */
10 JMP 0x33:offset bit64
11 bit64:
12 .code64
13 MOV RSP, RBP
14 POP RBP
15 /* Code execution continues as regular */
16
17 /* Location and CS value used for indirect far jump */
18 .section .rodata
19 bit32_data:
20 .int offset bit32
21 .short 0x23
```

Listing 4. Mode switch via Far Jump

4.3. Stealth mode switch

During the evaluation of the Far Jump method, it was found that some reverse engineering tools are unable to process this instruction at all. While this still renders them useless, it doesn't allow one to apply the more advanced protection — fake disassembly and decompilation, as proposed in the Section 3.2. While other methods noted in the Section 2 produce nearly the same results, there is one extra method that can be used — exception handlers. Both Linux and Windows expose the processor context to the exception (signal) handlers, so registers can be modified directly, and this also includes the CS register. This eliminates the problems that can appear with rare instruction forms like Far Jump or Far Return, because they are not used anymore.

It should still be noted that exception (signal) handler should preserve the stack pointer address, and the stack used while calling the handler should differ from the current program one, because the stack pointer is invalid due to truncation when the program is in the 32-bit mode. In Linux, this can be done by using the sigaltstack system call that allows one to specify the separate stack for signal handling.

However, causing exceptions in the code may look suspicious to the analyst and cause problems in some reverse engineering tools, as they may seem to be unhandled. For example, using the traditional UD2 instruction to throw an exception results in IDA setting function end as soon as this instruction is encountered.

In this paper, we propose a stealth method to cause exceptions that doesn't affect program listings, that is based on the optimizers abuse. Usually, reverse engineering tools simplify the code during processing to remove the junk code. While this is efficient against constant obfuscation, this gives us a valuable opportunity: we can add an operation that looks like a no-op to the optimizer, but still causes exception and so triggers our exception handler.

The easiest way to achieve this is using some meaningless arithmetic operation, for example, XOR some memory with zero. It is well known that applying XOR with zero to the value doesn't alter it, so the optimizer may remove this instruction if it doesn't care about possible side effects. The real processor, on the other side, will still try to read the value from memory, and trigger an exception if the address is invalid.

An example implementation of this method is available in the Listings 5, 6 and 7.

```
1  static void sighandler(int sno, siginfo_t *si, void *data)
2  {
3      static uintptr_t stack;
4      ucontext_t *uc = (ucontext_t *)data;
5      // Adjust RIP to skip the length
6      // of xor byte ptr [0x1337], 0
7      // It's 8 in 64-bit mode
8      if (uc->uc_mcontext.gregs[REG_CSGSFS] & 0x10)
9          uc->uc_mcontext.gregs[REG_RIP] += 8;
10     // It's 7 in 32-bit mode
11     else
12         uc->uc_mcontext.gregs[REG_RIP] += 7;
13     // We're switching from 64-bit, save stack ptr
14     if (uc->uc_mcontext.gregs[REG_CSGSFS] & 0x10)
15         stack = uc->uc_mcontext.gregs[REG_RSP];
16     // Restore otherwise
17     else
18         uc->uc_mcontext.gregs[REG_RSP] = stack;
19     // Switch 0x23 <-> 0x33
20     uc->uc_mcontext.gregs[REG_CSGSFS] ^= 0x10;
21 }
```

Listing 5. Signal handler implementation

```
1  static void prepare_sighandler()
2  {
3      // Setup an alternative stack for signal handler
4      // It doesn't matter if it is accessible from 32-bit
       address space
5      char * stack = malloc(SIGSTKSZ);
6      stack_t ss = {
7          .ss_size = SIGSTKSZ,
8          .ss_sp = stack,
9      };
10     sigaltstack(&ss, 0);
11
12     // Specify signal handler
13     struct sigaction action_handler;
14
15     action_handler.sa_sigaction = sighandler;
16     action_handler.sa_flags = SA_ONSTACK | SA_SIGINFO;
17
18     sigaction(SIGSEGV, &action_handler, NULL);
19 }
```

Listing 6. Signal handler setup

```
1       ...
2       XOR byte ptr [0x1337], 0
3       /* The following code is executed in 32-bit mode */
4       .code32
5       ...
6       /* Return to the 64-bit mode */
7       XOR byte ptr [0x1337], 0
8       .code64
9       ...
```

Listing 7. Stealth mode switch invocation

5. Software implementation

Being architecture-dependent, the proposed methods are difficult to be implemented at the source code or intermediate representation level. Therefore, machine code and assembly transforms are the only desirable solutions available. While machine code transformation is the most direct way to do machine-level obfuscations, it comes with major tradeoffs:

1) When working with machine code in the object file form, software should be able to distinguish between code and data reliably, and find instruction boundaries for variable length instruction encodings. This is a complicated task, especially provided that for machines with variable length instructions, it may be impossible to disassemble machine code correctly at all, as noted in the Introduction section.
2) Adding any extra code, or replacing existing code with a code of a different length, is also extremely complex. While removing code (and replacing with a smaller one) may be safely implemented by adding a corresponding number of one-byte NOP instructions, adding extra bytes to the code requires recomputing all relative offsets in the code, which include not only instructions but also data, which may have addresses stored in an unpredictable format.

Considering these problems, assembly transform was chosen as an implementation approach. It has other possible issues to consider, but they turned out to be manageable as long as GNU Assembler is used:

1) Assembler may be unable to produce mixed mode machine code, supporting only one variant of x86 at a time. While this may also be managed by carefully choosing instructions, so they match in both architectures, GNU Assembler provides a easier solution by supporting .code32 and .code64 directives. These directives allow to switch between 32-bit and 64-bit variant of x86, respectively.
2) Some machine code has no assembly representation, or there are multiple machine codes equivalent to the same instruction. This may be solved by manually writing machine code byte by byte using .byte directive.

For the easier integration with existing compilers and build systems, it was chosen to implement obfuscation as a GNU Assembler proxy, which transforms the input and then passes it, along with command-line flags, to the original GNU Assembler, a scheme of this process is available in the Fig. 3. This allows the tool to be used with any compiler that supports the use of an external assembly, including the most popular open source compilers GCC and Clang.

The proxy was implemented in Python programming language, processing the assembly code as text, with no additional assembly parsers involved. This decision was made considering the relative simplicity of the assembly language and the transformations applied.

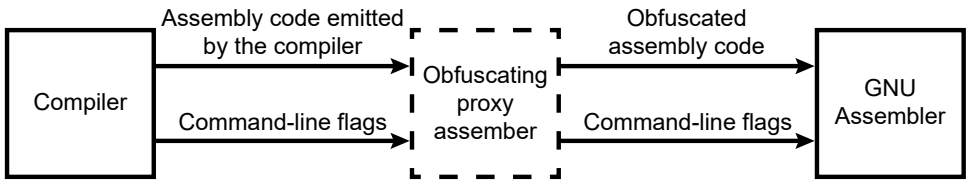

Fig. 3. Obfuscating proxy diagram

It was decided to support the Intel x86 syntax only (not AT&T which some of the compilers emit by default), but this is actually a limitation of the obfuscator prototype used in this work, there is little problem in implementing support for other syntaxes if needed.

Another desirable property of the obfuscator is the ability to fine-tune it right from the source code of the program, specifying which obfuscations should be applied to a particular function or a place in the program. Some of the obfuscators, e.g. Tigress, are using command-line flags for this kind of control, so the user has to specify all the functions he would like to have obfuscated in these flags. This approach was found to be impractical for the two main reasons. First of all, it is complicated to pass command-line flags to the external assembler: while compilers may have options to pass flags to assembler, they don't really have to (there's still an option to use environment variables, but this is harder to debug due to their automatic inheritance). And secondly, this is boring if the project is large enough.

So another approach was proposed: introduce special macros that user can utilize right in the source of the program to configure how it will be obfuscated. While the obfuscator doesn't have direct access to the source code, because it works on the lower level, we may define these macros to emit some inline assembly our tool may process. To our luck, inline assembly (at least in C/C++) gets emitted right in the place where it was defined, so this may even be a middle of the function, giving the most precise control possible.

To not introduce unnecessary compatibility breakage, it was decided to make these macros emit assembly comments keeping backward compatibility with the original assembler if the user would like to build an unobfuscated copy without having to bother about any compilation defines. Example of macros implementing this approach is available in the Listing 8, example of the usage is available in the Listings 9 and 10.

```
1 #define OBF_DEF_LABEL(name) \
2     asm ( "#Obf_label " #name ":" )
3 #define OBF_FAKE_FUN(funname) \
4     asm ( "#Obf_fake_fun " #funname )
5 #define OBF_FAKE_JUMP(funname, retlabel) \
6     asm ( "#Obf_fake_jumpback " #funname " " #retlabel)
```

Listing 8. Macros emitting assembly comments

```
1     printf("This code is entirely visible\n");
2     OBF_FAKE_JUMP(fakefun, retlabel);
3     printf("This is the code to be hidden\n");
4     OBF_DEF_LABEL(retlabel);
5     printf("Code continues\n");
```

Listing 9. A code fragment using proposed macros

```
1       lea rdi, .LC0[rip]
2       call    puts@PLT
3       #Obf_fake_jumpback fakefun retlabel
4       lea rdi, .LC1[rip]
5       call    puts@PLT
6       #Obf_label retlabel:
7       lea rdi, .LC2[rip]
8       call    puts@PLT
```

Listing 10. An assembly code generated by the compiler

6. Results

6.1. Testing methodology

To check the impact of the proposed methods, the following test cases were chosen:

1) A 64-bit program that implements the method described in the Section 3.2 using a Far Jump instruction described in the Section 4.2. Tools that do not support mode switching at all will also automatically fail this test.
2) A 64-bit program that implements the method described in the Section 3.2 using a stealth mode switch proposed in the Section 4.3.
3) A 32-bit program that implements the method described in the Section 3.3. The program is chosen to be built as 32-bit instead of 64-bit so that the tools that fail the previous tests may still pass it independently.

The following reverse engineering tools were chosen for testing:

1) IDA Pro — a popular commercial interactive disassembler with a decompiler [20]. Testing is done both for a disassembler and a decompiler.
2) Ghidra — a popular open source disassembler with a decompiler [21]. Testing is done both for a disassembler and a decompiler.
3) Valgrind — a dynamic binary instrumentation framework [22]. During the test, it is required to execute a program correctly.
4) Angr — an open source binary analysis platform for Python commonly used as a symbolic execution tool [23]. During the test, it is required to reach a particular path in the program. It was tested with two different binary lifters, default PyVEX (based on Valgrind) and P-code (based on Ghidra).

The program chosen for the testing is available in the Listing 11. It imitates a very simple license check with the use of obfuscation macros implementing the method used in tests 1 and 2, OBF_FAKE_JUMP macro inserts a jump intended to be seen by a decompiler only, while OBF_DEF_LABEL defines a label where execution should return to. Due to the availability of two different code paths, this code is also applicable for using with Angr, and an explicit register zeroing also makes it possible to test the method from the test 3, so this single code covers all scenarios required. The environment used for all tests was Linux with the GCC compiler, and all 64-bit tests were compiled with "-no-pie" flag specified.

```
1 #include <stdio.h>
2 #include <obf_defines.h>
3
4 static int user_input;
5 static int check_result;
6 int main()
```

```
 7 {
 8     puts("Please enter license code:");
 9     scanf("%d", &user_input);
10     OBF_FAKE_JUMP(fakefun, retlabel);
11     // This is the real code supposed to be hidden
12
13     // We make sure that zeroing is done with a register
14     // so we can also apply a method described in 2.3
15     register int valid_value = 0;
16     for (int i = 0; i < 10; i++)
17         valid_value += i * i;
18     check_result = valid_value == user_input;
19     OBF_DEF_LABEL(retlabel);
20     if (check_result)
21         puts("License check passed");
22     else
23         puts("License check failed");
24     return 0;
25 }
26
27 void fakefun()
28 {
29     // Fake code supposed to be seen by the decompiler
30     int valid_value = 0;
31     for (int i = 0; i < 1337; i++)
32         valid_value += i * i;
33     check_result = valid_value == user_input;
34 }
```

Listing 11. The code used for testing

6.2. Far Jump experiment results

The following results were observed for the program built with "machine code mismatch" obfuscation enabled with Far Jump mode switching implementation:

— IDA Pro has successfully identified the Far Jump instruction, but it was unable to find its destination, even though it was written in plain bytes. The code following this instruction was considered to be data. After correcting this mistake, the code was disassembled as a pair of XOR and JMP instructions, with no signs of REX prefix or INC instruction. The decompiler has been unable to follow the function past the Far Jump instruction.
— Ghidra has successfully identified the Far Jump instruction and its destination. The code following it was also disassembled, but the mode change was not noticed, so INC instruction wasn't found and REX prefix was omitted. Interestingly enough, Ghidra has also been able to disassemble an immediate Far Jump (EA opcode) correctly despite it being disallowed in 64-bit mode. The decompiler has been able to decompile the whole function "correctly" respectively to the generated disassembly, effectively making a decompiler show the fake program listing provided by the obfuscation.
— Valgrind has been unable to execute the program past the first (64-bit) Far Jump instruction, reporting "unhandled instruction bytes" error, with its bytes followed.

— Angr has been unable to execute the Far Jump instruction with an IR decoding error when PyVEX was used. After switching to the P-code lifter, the execution has succeeded, but the path found belonged to the fake code inserted by the obfuscator.

6.3. Stealth mode switch experiment results

The program was built in the same way as in the previous experiment, except that it used the implementation of stealth mode switching, and this change had a big impact on the results:

— Both IDA Pro and Ghidra have successfully disassembled the code in a 64-bit mode, ignoring the REX / INC byte. Both decompilers have successfully decompiled the fake function they were intended to show by the obfuscation, completely ignoring the XOR operation causing the memory error, that is used for the stealth mode switching.
— Valgrind has been unable to execute the program past mode switch with multiple "unhandled instruction bytes" and "invalid read" errors reported. Instead of crashing, the program has stuck.
— Angr has successfully executed the code and found a path in the fake code with PyVEX. In the P-code mode, execution has failed with error "CALLOTHER emulation not currently supported".

6.4. Deprecated instructions experiment results

In this experiment, the program was built with "deprecated instructions" obfuscation enabled in 32-bit mode, so there was no mode switching in the program. The following results were observed:

— IDA Pro has successfully disassembled the AAM and AAD instructions, along with their machine code-level base arguments. The decompiler has failed to optimize this construction, leaving it in the inline assembly form.
— Ghidra has successfully disassembled the AAM and AAD instructions, along with their arguments. The decompiler has been able to optimize the construction to the variable zeroing, beating the obfuscation.
— Valgrind has been unable to execute the program past the first non-standard AAM instruction, reporting "unhandled instruction bytes" error, with its bytes followed.
— Angr has been able to find a correct answer with both lifters. This is surprising, provided that Valgrind failed to execute the non-standard AAM / AAD forms, but as seen in the warning logs, this support has been manually implemented in PyVEX in 2022, according to the commit history [24].

6.5. Results summary and evaluation

Systemized results of the experiments are available in the Table 2.

Table 2

Observed results

Tool	Far Jump	Stealth	Deprecated instr.
IDA Pro	Failure	Fake code	Success
IDA Pro (dec.)	Failure	Fake code	Failure
Ghidra	Fake code	Fake code	Success
Ghidra (dec.)	Fake code	Fake code	Success
Valgrind	Failure	Failure	Failure
Angr (PyVEX)	Failure	Fake code	Success
Angr (P-code)	Fake code	Failure	Success

As can be seen in the table, mode switching defeats all the tools evaluated. Stealth mode switching is particularly efficient against reverse engineering, because it allows the attacker to see the fake code instead of a real one in most tools, as fake code is much more dangerous to the reverse engineer than a not working tool, because it draws no attention and possibly makes the attacker waste time on it.

Deprecated instructions obfuscation is not as efficient against modern tools, though: it still breaks IDA Pro decompiler, making reverse engineering inconvenient, but has little impact on the other tools. However, it may be effective against less popular and less actively supported tools than those supported at the time of writing, since even Angr only recently received proper support for these instructions (in PyVEX and general P-code support).

7. Conclusion

A novel obfuscation approach is presented, that is not only efficient against the most popular reverse engineering tools, but also has some unique properties.

First, it allows one to build a code that looks different from a disassembler and decompiler output dramatically without using any runtime code modification, an approach commonly used for this task in packers and protectors. This opens an opportunity to use this method on the most protected operating systems, which deny any modification of executable code in runtime. It is also extremely useful against casual reverse engineers that rely on the decompiler only: they have zero chance to know what is actually going on in the protected parts of the program, because this information is not shown in the decompiler output at all.

Secondly, fixing reverse engineering tools to fight this approach is complicated and most possibly time-consuming, because it involves implementing a possibility to switch architectures in the single program. While this was implemented in the past for the ARM-Thumb transitions, a motivation to do so for x86 is incomparably less, as this feature is not used in the real-world applications.

The deprecated instructions use doesn't look as promising on its own in a 32-bit mode, but it is still useful in combination with mode switching, because forcing the code to be disassembled as 64-bit will make these instructions impossible to disassemble for a standard-compliant disassembler, confusing the attacker.

While there is no performance evaluation of the methods in this paper, it is possible to eliminate most of the possible slowdowns with the ability to fine-tune methods usage straight from the source code, thanks to the proposed obfuscation control approach. Nevertheless, it is possible to approximate the time of a mode switching by analyzing the code to be equal to a few context switches worst case, which is barely noticeable unless used in heavy cycles.

REFERENCES

1. *Barak B., Goldreich O., Impagliazzo R., et al.* On the (im)possibility of obfuscating programs. LNCS, 2001, vol. 2139, pp. 1–18.
2. `https://github.com/SamboyCoding/Cpp2IL` — Cpp2IL: Work-in-progress tool to reverse unity's IL2CPP toolchain, 2023.
3. *Wang C., Davidson J., Hill J., and Knight J.* Protection of software-based survivability mechanisms. Proc. Intern. Conf. Dependable Syst. Networks, Goteborg, 2001, pp. 193–202.
4. *Collberg C., Thomborson C., and Low D.* Manufacturing cheap, resilient, and stealthy opaque constructs. Proc. 25th ACM SIGPLAN-SIGACT Symp. POPL'98, San Diego, California, USA, 1998, pp. 184–196.

5. *Collberg C., Thomborson C., and Low D.* Breaking abstractions and unstructured data structures. Proc. Intern. Conf. Computer Languages, Chicago, IL, USA, 1998, pp. 28–38.
6. *Warren H. S.* Hacker's Delight, Second Ed. Addison-Wesley, 2012. 512 p.
7. *Junod P., Rinaldini J., Wehrli J., and Michielin J.* Obfuscator-LLVM — software protection for the masses. IEEE/ACM 1st Intern. Workshop Software Protection, Florence, Italy, 2015, pp. 3–9.
8. `https://tigress.wtf` — the tigress c obfuscator, 2023.
9. *Ugarte-Pedrero X., Balzarotti D., Santos I., and Bringas P. G.* SoK: deep packer inspection: A longitudinal study of the complexity of run-time packers. EEE Symp. Security and Privacy, San Jose, CA, USA, 2015, pp. 659–673.
10. *Jamthagen C., Lantz P., and Hell M.* A new instruction overlapping technique for anti-disassembly and obfuscation of x86 binaries. Workshop Anti-malware Testing Research, Montreal, QC, Canada, 2013, pp. 1–9.
11. *Cohen F. B.* Operating system protection through program evolution. Computers and Security, 1993, vol. 12, no. 6, pp. 565–584.
12. *Lebedev R. K. and Koryakin I. A.* Primenenie rasshireniy arkhitektury x86 v zashchite programmnogo koda [Application of x86 extensions for code protection]. Prikladnaya diskretnaya matematika. Prilozhenie, 2021, no. 14, pp. 138–140. (in Russian)
13. `https://wbenny.github.io/2018/11/04/wow64-internals.html` — WoW64 internals, 2018.
14. `https://community.osr.com/discussion/246643` — Understanding Win 7 x64 GDT/LDT, 2013.
15. `https://github.com/torvalds/linux/blob/master/arch/x86/kernel/cpu/common.c` — Linux Kernel (GitHub), 2023.
16. Intel 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4. December 2022. 5060 p.
17. `https://github.com/NationalSecurityAgency/ghidra/issues/510` — Allow Different Instruction Sets for Different Memory Sections (Ghidra, GitHub), 2023.
18. Assembly language is too high level. DEF CON 25, 2017. `https://media.defcon.org/DEFCON25/DEFCON25presentations/DEFCON25-XlogicX-Assembly-Language-Is-Too-High-Level.pdf`
19. *Collberg C., Thomborson C., and Low D.* A Taxonomy of Obfuscating Transformations. Technical Report. Department of Computer Science, The University of Auckland, 1997, no. 148. `https://researchspace.auckland.ac.nz/bitstream/handle/2292/3491/TR148.pdf`.
20. `https://hex-rays.com/ida-pro` — Hex Rays — State-of-the-art binary code analysis solutions, 2023.
21. `https://github.com/NationalSecurityAgency/ghidra` — Ghidra Software Reverse Engineering Framework (GitHub), 2023.
22. *Nethercote N., and Seward J.* Valgrind: a framework for heavyweight dynamic binary instrumentation. SIGPLAN Not., 2007, vol. 42, no. 6, pp. 89–100.
23. *Shoshitaishvili Y., Wang R., Salls C., et al.* SOK: (State of) The art of war: Offensive techniques in binary analysis. IEEE Symp. Security Privacy (SP), San Jose, CA, USA, 2016, pp. 138–157.
24. `https://github.com/angr/pyvex/commit/46049a14985a8d78c6679d75f103540b94c22bc5` — Add generalized aam and aad instructions for x86, angr/pyvex (GitHub), 2022.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52 DOI 10.17223/20710410/61/7

О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ ПРОБЛЕМЫ ФАКТОРИЗАЦИИ ЦЕЛЫХ ЧИСЕЛ[1]

А. Н. Рыбалов

Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

E-mail: alexander.rybalov@gmail.com

Изучается генерическая сложность проблемы факторизации целых чисел. Данная проблема, восходящая ещё к Гауссу, имеет важное значение для современной криптографии. Например, на предположении о её трудноразрешимости основывается криптостойкость системы шифрования с открытым ключом RSA. В работе доказывается, что при условиях трудноразрешимости этой проблемы в худшем случае и P = BPP для её решения не существует полиномиального сильно генерического алгоритма. Сильно генерический алгоритм решает проблему не на всём множестве входов, а на подмножестве, последовательность относительных плотностей которого при увеличении размера экспоненциально быстро сходится к единице. Для доказательства используется метод генерической амплификации, который позволяет строить генерически трудные проблемы из проблем, трудных в худшем случае. Основным ингредиентом этого метода является объединение эквивалентных входов в достаточно большие множества. Эквивалентность входов означает, что рассматриваемая проблема на них решается одинаково.

Ключевые слова: *генерическая сложность, факторизация целых чисел.*

ON GENERIC COMPLEXITY OF THE INTEGER FACTORIZATION PROBLEM

A. N. Rybalov

Sobolev Institute of Mathematics, Omsk, Russia

In the paper, we study the generic complexity of the integer factorization problem. This problem, which goes back to Gauss, has important applications in modern cryptography. For example, the cryptographic strength of the famous public key encryption system RSA is based on the assumption of its hardness. We prove that under the condition of worst-case hardness and P = BPP for the problem of integer factorization there is no polynomial strongly generic algorithm. A strongly generic algorithm solves a problem not on the entire set of inputs, but on a subset whose frequency sequence converges exponentially to 1 with increasing size. To prove this theorem, we use the method of generic amplification, which allows to construct generically hard problems from the problems hard in the classical sense. The main component of this method is

[1]Работа выполнена в рамках госзадания ИМ СО РАН, проект FWNF-2022-0003.

the cloning technique, which combines problem inputs together into sufficiently large sets of equivalent inputs. Equivalence is understood in the sense that the problem is solved similarly for them.

Keywords: *generic complexity, integer factorization.*

Введение

Проблема факторизации (разложения на множители) целых чисел является классической алгоритмической проблемой теории чисел, восходящей ещё к Гауссу. Для неё до сих пор не найдены эффективные (полиномиальные) алгоритмы [1]. На предположении о трудноразрешимости проблемы факторизации основана знаменитая система шифрования с открытым ключом RSA [2].

В современной криптографии интересны такие проблемы, которые, являясь (гипотетически) трудными в классическом смысле, остаются трудными и в генерическом смысле, т. е. для почти всех входов. Это объясняется тем, что при случайной генерации ключей в криптографическом алгоритме происходит генерация входа некоторой трудной алгоритмической проблемы, лежащей в основе алгоритма. Если проблема будет легкоразрешимой почти всегда, то для почти всех таких входов ее можно будет быстро решить и ключи почти всегда будут нестойкими. Поэтому проблема должна быть трудной для почти всех входов. Например, таким поведением обладают классические алгоритмические проблемы криптографии: проблема распознавания квадратичных вычетов [3], проблема дискретного логарифма [4], проблема извлечения корня в группах вычетов [5] (проблема обращения функции RSA).

Генерический подход [6] — это один из подходов к изучению алгоритмических проблем для «почти всех» входов. В рамках этого подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Такие входы образуют так называемое генерическое множество. Понятие «почти все» формализуется введением естественной меры на множестве входных данных. С точки зрения практики алгоритмы, решающие быстро проблему на генерическом множестве, так же хороши, как и быстрые алгоритмы для всех входов.

В данной работе изучается генерическая сложность проблемы факторизации целых чисел. Доказывается, что при условии трудноразрешимости этой проблемы в худшем случае и P = BPP для неё не существует полиномиального сильно генерического алгоритма. Сильно генерический алгоритм решает проблему не на всём множестве входов, а на подмножестве, последовательность относительных плотностей которого при увеличении размера экспоненциально быстро сходится к единице. Класс BPP состоит из проблем, разрешимых за полиномиальное время на вероятностных машинах Тьюринга. Считается, что класс BPP совпадает с классом P, то есть любой полиномиальный вероятностный алгоритм можно эффективно дерандомизировать, построив полиномиальный алгоритм, не использующий генератор случайных чисел и решающий ту же самую проблему. Хотя равенство P = BPP до сих пор не доказано, имеются веские основания в пользу него [7].

1. Генерические алгоритмы

Пусть I — некоторое множество входов. Для подмножества $S \subseteq I$ определим *последовательность относительных плотностей*

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где I_n — множество входов размера n, а $S_n = S \cap I_n$. Заметим, что $\rho_n(S)$ — это вероятность попасть в S при случайной и равновероятной генерации входов из I_n. В данной работе множеством входов для алгоритмов является множество натуральных чисел, записанных в двоичной форме. Под размером натурального числа понимаем длину его двоичной записи.

Асимптотической плотностью множества S назовем верхний предел

$$\rho(S) = \overline{\lim_{n\to\infty}} \rho_n(S).$$

Множество S называется *генерическим*, если $\rho(S) = 1$, и *пренебрежимым*, если $\rho(S) = 0$. Очевидно, что S генерическое тогда и только тогда, когда его дополнение $I \setminus S$ пренебрежимо.

Следуя [6], назовём множество S *сильно пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к нулю, т. е. существуют константы σ, $0 < \sigma < 1$, и $C > 0$, такие, что для любого n

$$\rho_n(S) < C\sigma^n.$$

Теперь S называется *сильно генерическим*, если его дополнение $I \setminus S$ сильно пренебрежимо.

Алгоритм $\mathcal{A}$ с множеством входов I и множеством выходов $J \cup \{?\}$ $(? \notin J)$ называется (*сильно*) *генерическим*, если:

1) $\mathcal{A}$ останавливается на всех входах из I;
2) множество $\{x \in I : \mathcal{A}(x) \neq ?\}$ является (сильно) генерическим.

Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если $(\mathcal{A}(x) = y \in J) \Rightarrow$ $\Rightarrow (f(x) = y)$ для всех $x \in I$. Ситуация $\mathcal{A}(x) = ?$ означает, что $\mathcal{A}$ не может вычислить функцию f на аргументе x. Но условие 2 гарантирует, что $\mathcal{A}$ корректно вычисляет f на почти всех входах (входах из генерического множества). Множество $S \subseteq I$ называется (*сильно*) *генерически разрешимым за полиномиальное время*, если существует (сильно) генерический полиномиальный алгоритм, вычисляющий его характеристическую функцию.

2. Вероятностные алгоритмы

Напомним некоторые понятия классической теории сложности вычислений. *Время работы* $t_M(x)$ машины Тьюринга M на входе $x \in I$ — это число шагов машины от начала работы до остановки. Если M на x не останавливается, полагаем $t_M(x) = \infty$. Машина Тьюринга M *полиномиальна*, если существует полином $p(n)$, такой, что для любого $x \in I$ имеет место $t_M(x) < p(|x|)$. Класс P состоит из подмножеств I, распознаваемых полиномиальными машинами Тьюринга.

Вероятностная машина Тьюринга — это машина Тьюринга, в программе которой допускаются пары правил вида

$$(q_i, a) \to (q_j, b, S_1),$$
$$(q_i, a) \to (q_k, c, S_2).$$

В процессе работы такой машины с вероятностью $1/2$ выбирается первое правило и с вероятностью $1/2$ — второе. Обозначим через $\mathsf{P}[M(x) = y]$ вероятность того, что машина M на входе x выдаёт ответ y. Время работы $t_M(x, \tau)$ вероятностной машины Тьюринга на входе x зависит от вычислительного пути (последовательности выполненных команд) τ. Проблема $S \subseteq I$ принадлежит классу BPP, если существует вероятностная машина Тьюринга M и полином $p(n)$, такие, что:

1) для любого x и для любого вычислительного пути τ машины M на x имеет место $t_M(x,\tau) < p(|x|)$;
2) если $x \in S$, то $\mathsf{P}[M(x) = 1] > 2/3$;
3) если $x \notin S$, то $\mathsf{P}[M(x) = 0] > 2/3$.

Вероятностные машины Тьюринга формализуют понятие алгоритма, использующего генератор случайных чисел. Класс BPP — это класс проблем, эффективно решаемых такими вероятностными алгоритмами. Большинство специалистов по теоретической информатике сейчас считает, что имеет место равенство P = BPP. Это означает, что любой полиномиальный вероятностный алгоритм можно эффективно дерандомизировать, т. е. построить полиномиальный детерминированный алгоритм, решающий ту же задачу. Хотя равенство пока не доказано, имеются серьёзные результаты в его пользу [7].

3. Проблема факторизации целых чисел

Проблема факторизации целых чисел состоит в следующем. Дано натуральное число N, записанное в двоичной системе. Необходимо найти его разложение в произведение степеней простых чисел: $N = p_1^{k_1} \dots p_m^{k_m}$.

Лемма 1. Существует полиномиальный алгоритм, который для любых натуральных чисел N и M по разложению на простые множители их произведения $NM = p_1^{k_1} \dots p_m^{k_m}$ находит разложение на простые множители отдельно для чисел N и M.

Доказательство. Искомый полиномиальный алгоритм работает следующим образом. Для каждого простого числа p_i, входящего в степени k_i в разложение числа NM, пытаемся разделить N сначала на p_i. Если N делится на p_i без остатка, то N/p_i делим на p_i. И так далее до тех пор, пока не получим ненулевой остаток от деления. Тем самым находим максимальную степень s_i простого p_i, входящую в разложение числа N. Проделав это для всех p_i, найдём искомое разложение $N = p_1^{s_i} \dots p_m^{s_m}$. Тогда $M = p_1^{k_i - s_i} \dots p_m^{k_m - s_m}$.

Полиномиальность алгоритма следует из того, что операция деления с остатком проводится за полиномиальное время и количество простых множителей в разложении числа NM ограничено величиной $\log(NM)$, то есть размером входа. ■

4. Основной результат

Теорема 1. Если существует сильно генерический полиномиальный алгоритм, решающий проблему факторизации целых чисел, то существует вероятностный полиномиальный алгоритм, решающий эту проблему на всём множестве входов.

Доказательство. Допустим, что существует сильно генерический полиномиальный алгоритм $\mathcal{A}$, решающий проблему факторизации. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, решающий эту проблему на всём множестве входов. На натуральном числе N размера n ($2^n \leqslant N < 2^{n+1}$) алгоритм $\mathcal{B}$ работает следующим образом:

1) генерирует случайно и равновероятно натуральное число M размера $n^2 - n$;
2) запускает алгоритм $\mathcal{A}$ на числе NM;
3) если $\mathcal{A}(NM) \neq ?$, то есть алгоритм выдаёт разложение $NM = p_1^{k_1} \dots p_m^{k_m}$, то по лемме 1 находим за полиномиальное время разложение на простые множители для числа N;
4) если $\mathcal{A}(NM) = ?$, то выдаёт ответ 2.

Заметим, что полиномиальный вероятностный алгоритм $\mathcal{B}$ выдаёт правильный ответ на шаге 3, а на шаге 4 может выдать неправильный ответ. Нужно доказать, что вероятность того, что ответ выдаётся на шаге 4, меньше $1/3$.

Оценим вероятность выдачи ответа на шаге 4. Число M имеет размер n^2-n, значит, размер числа NM равен $n^2-n+n=n^2$. Вероятность того, что для NM имеет место $\mathcal{A}(NM)=?$, не больше

$$\frac{|\{K\in\mathbb{N}:\mathcal{A}(K)=?\}_{n^2}|}{|\{NM:M\in\mathbb{N}_{n^2-n}\}_{n^2}|}=\frac{|\{K\in\mathbb{N}:\mathcal{A}(K)=?\}_{n^2}|}{|\mathbb{N}_{n^2}|}\,\frac{|\mathbb{N}_{n^2}|}{|\{NM:M\in\mathbb{N}_{n^2-n}\}_{n^2}|}=$$
$$=\frac{|\{K\in\mathbb{N}:\mathcal{A}(K)=?\}_{n^2}|}{|\mathbb{N}_{n^2}|}\,\frac{2^{n^2}}{2^{n^2-n}}=2^n\frac{|\{K\in\mathbb{N}:\mathcal{A}(K)=?\}_{n^2}|}{|\mathbb{N}_{n^2}|}.$$

Так как множество $\{K\in\mathbb{N}:\mathcal{A}(K)=?\}$ сильно пренебрежимое, то существует константа $\alpha>0$, такая, что

$$\frac{|\{K\in\mathbb{N}:\mathcal{A}(K)=?\}_{n^2}|}{|\mathbb{N}_{n^2}|}<\frac{1}{2^{\alpha n^2}}$$

для любого n. Поэтому искомая вероятность ответа на шаге 4 не больше

$$\frac{2^n}{2^{\alpha n^2}}=\frac{1}{2^{\alpha n^2-n}}<\frac{1}{3}$$

при больших n. ■

Теорема 2. Если для проблемы факторизации не существует полиномиального алгоритма и P = BPP, то для неё не существует сильно генерического полиномиального алгоритма.

Доказательство. Пусть существует сильно генерический алгоритм, решающий проблему факторизации. Тогда, по теореме 1, существует вероятностный полиномиальный алгоритм, решающий её на всём множестве входов. Этот же алгоритм решает следующую проблему распознавания A, которая полиномиально эквивалентна проблеме факторизации: даны натуральные числа N и K, нужно определить, существует ли нееединичный множитель N меньше K. Таким образом, проблема A лежит в классе BPP. Так как P = BPP, то она лежит и в классе P, а значит, для проблемы факторизации существует полиномиальный алгоритм. Противоречие. ■

ЛИТЕРАТУРА

1. *Adleman L. M. and McCurley K. S.* Open problems in number theoretic complexity, II // LNCS. 1994. V. 877. P. 291–322.
2. *Rivest R., Shamir A., and Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. 1978. V. 21. Iss. 2. P. 120–126.
3. *Рыбалов А. Н.* О генерической сложности проблемы распознавания квадратичных вычетов // Прикладная дискретная математика. 2015. № 2 (28). С. 54–58.
4. *Рыбалов А. Н.* О генерической сложности проблемы дискретного логарифма // Прикладная дискретная математика. 2016. № 3 (33). С. 93–97.
5. *Рыбалов А. Н.* О генерической сложности проблемы извлечения корня в группах вычетов // Прикладная дискретная математика. 2017. № 38. С. 95–100.
6. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.
7. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma. Proc. 29th STOC. El Paso, ACM, 1997. P. 220–229.

REFERENCES

1. *Adleman L. M. and McCurley K. S.* Open problems in number theoretic complexity, II. LNCS, 1994, vol. 877, pp. 291–322.
2. *Rivest R., Shamir A., and Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM, 1978, vol. 21, iss. 2, pp. 120–126.
3. *Rybalov A. N.* O generitcheskoy slozhnosti problemy raspoznavaniya kvadratichnykh vychetov [On generic complexity of the quadratic residuosity problem]. Prikladnaya Diskretnaya Matematika, 2015, no. 2 (28), pp. 54–58. (in Russian)
4. *Rybalov A. N.* O generitcheskoy slozhnosti problemy diskretnogo logarifma [On generic complexity of the discrete logarithm problem]. Prikladnaya Diskretnaya Matematika, 2016, no. 3 (33), pp. 93–97. (in Russian)
5. *Rybalov A. N.* O generitcheskoy slozhnosti problemy izvlecheniya kornya v gruppakh vychetov [On generic complexity of the problem of finding roots in groups of residues]. Prikladnaya Diskretnaya Matematika, 2017, no. 38, pp. 95–100. (in Russian)
6. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
7. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma. Proc. 29th STOC, El Paso, ACM, 1997, pp. 220–229.

СВЕДЕНИЯ ОБ АВТОРАХ

ВОЛОШИН Савва Константинович — аспирант Омского государственного университета им. Ф. М. Достоевского, г. Омск. E-mail: **savva.voloshin@gmail.com**

ГАЙДАМАКИН Николай Александрович — доктор технических наук, профессор, профессор кафедры алгебры и фундаментальной информатики Уральского федерального университета имени первого Президента России Б. Н. Ельцина, г. Екатеринбург. E-mail: **n.a.gaidamakin@urfu.ru**

ЛЕБЕДЕВ Роман Константинович — ассистент кафедры компьютерных систем факультета информационных технологий Новосибирского государственного университета, г. Новосибирск. E-mail: **n0n3m4@gmail.com**

МАРТЫНЕНКОВ Игорь Владимирович — АО «КВАНТ-ТЕЛЕКОМ», г. Москва. E-mail: **mivpost@yandex.ru**

РОМАНЬКОВ Виталий Анатольевич — доктор физико-математических наук, профессор, главный научный сотрудник Института математики им. С. Л. Соболева СО РАН (Омский филиал), г. Омск. E-mail: **romankov48@mail.ru**

РЫБАЛОВ Александр Николаевич — кандидат физико-математических наук, старший научный сотрудник лаборатории комбинаторных и вычислительных методов алгебры и логики Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **alexander.rybalov@gmail.com**

РЯБОВ Владимир Геннадьевич — кандидат физико-математических наук, генеральный директор НП «ГСТ», г. Москва. E-mail: **4vryabov@gmail.com**

ЧЕРЕМУШКИН Александр Васильевич — доктор физико-математических наук, академик Академии криптографии Российской Федерации, г. Москва. E-mail: **avc238@mail.ru**

Журнал «Прикладная дискретная математика» входит в перечень ВАК рецензируемых научных изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание учёной степени кандидата и доктора наук по специальностям 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» (технические науки), 2.3.6. «Методы и системы защиты информации, информационная безопасность» (физико-математические и технические науки), 1.1.5. «Математическая логика, алгебра, теория чисел и дискретная математика» (физико-математические науки), 1.2.3. «Теоретическая информатика, кибернетика» (физико-математические науки), а также в перечень журналов, рекомендованных ФУМО ВО ИБ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал индексируется в базах данных Web of Science (Emerging Sources Citation Index (ESCI) и Russian Science Citation Index (RSCI)), Scopus, MathSciNet и Zentralblatt MATH. В сводном рейтинге журналов RSCI 2022 г. он отнесён к первому квартилю (Q1).

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте journals.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также правила подготовки рукописей статей для публикации в журнале.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надёжности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*