Банк России

СТАНДАРТ БАНКА РОССИИ

СТО БР ФАПИ.СЕК-1.6-2024

Безопасность финансовых (банковских) операций

Прикладные программные интерфейсы обеспечения безопасности финансовых сервисов на основе протокола OpenID Connect

Требования

Москва
2024

ПРЕДИСЛОВИЕ

1. Разработан Ассоциацией развития финансовых технологий (Ассоциация ФинТех) и акционерным обществом «Информационные технологии и коммуникационные системы» (АО «ИнфоТеКС») при участии Центрального банка Российской Федерации (Банка России).

2. Принят и введен в действие приказом Банка России от 07.10.2024 № ОД-1615.

3. Взамен СТО БР ФАПИ.СЕК-1.6-2020.

Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29.06.2015 № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту, пересмотре (замене) или отмене стандарта размещается на сайте Банка России в сети Интернет (http://www.cbr.ru/).

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован или распространен в качестве официального издания без разрешения Банка России.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящий стандарт разработан для применения кредитными организациями, некредитными финансовыми организациями, субъектами национальной платежной системы, лицами, оказывающими профессиональные услуги на финансовом рынке (далее – организации финансового рынка), прикладных программных интерфейсов (application programming interface, API) и основан на спецификациях технологии OpenID Connect Core (OIDC), которые определяют порядок использования модели API со структурированными данными и модели токена для повышения безопасности финансовых технологий.

Настоящий стандарт устанавливает требования к информационной безопасности при реализации взаимодействия с использованием API, обеспечивающие необходимый уровень защищенности информации при передаче персональных данных и банковской тайны.

Настоящий стандарт учитывает методические рекомендации МР.26.2.002-2024 Технического комитета по стандартизации 26 (ТК 26) «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколах OpenID Connect» и дополняет в части требований информационной безопасности при реализации взаимодействия с использованием Открытых банковских интерфейсов.

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящий стандарт рекомендован к использованию при создании и оценке соответствия требованиям настоящего стандарта программных средств, предназначенных для безопасного обмена финансовыми сообщениями в среде Открытых банковских интерфейсов в соответствии со стандартами Банка России:

– «Открытые банковские интерфейсы. Общие положения» [1];

– «Открытые банковские интерфейсы. Получение публичной информации о кредитной организации и ее продуктах» [2];

– «Открытые банковские интерфейсы. Инициирование перевода денежных средств клиента третьей стороной в валюте Российской Федерации» [3];

– «Открытые банковские интерфейсы. Получение информации о счете клиента третьей стороной» [4].

Положения настоящего стандарта носят рекомендательный характер, если только обязательность применения отдельных из них не установлена нормативными правовыми актами, в том числе нормативными актами Банка России. Настоящий стандарт может быть использован для включения ссылок на него и/или прямого включения содержащихся в нем положений во внутренние документы организаций финансового рынка, а также в договоры, заключенные между организациями.

Положения настоящего стандарта применяются совместно с методическими рекомендациями МР.26.2.002-2024 Технического комитета ТК 26 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколах OpenID Connect» [5][1].

[1] В случае реализации трансграничного обмена криптографические механизмы и порядок их использования определяются в рамках соглашения (договора) между сторонами обмена. При одновременной реализации трансграничного и внутрироссийского обмена защищенные ресурсы должны быть разделены.

2. НОРМАТИВНЫЕ ССЫЛКИ

В настоящем стандарте использованы ссылки на следующие документы:

– ГОСТ Р 56939-2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования»;

– Р 1323565.1.020-2020 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.2)»;

– Р 1323565.1.030-2020 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.3)».

Примечание. При пользовании настоящим стандартом целесообразно проверить действие ссылочных документов в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен ссылочный документ, на который дана недатированная ссылка, рекомендуется использовать действующую версию этого документа с учетом всех внесенных в данную версию изменений. Если заменен ссылочный документ, на который дана датированная ссылка, рекомендуется использовать версию этого документа с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный документ, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, это положение рекомендуется применять без учета данного изменения. Если ссылочный документ отменен без замены, положение, в котором дана ссылка на него, применяется в части, не затрагивающей эту ссылку.

3. ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

3.1.

Авторизация (authorization): проверка, подтверждение и предоставление прав логического доступа при осуществлении субъектами доступа логического доступа.

[ГОСТ Р 57580.1-2017, пункт 3.15]

3.2.

Агент пользователя (user agent): клиентское приложение, использующее определенный сетевой протокол для доступа к серверу. Термин обычно используется для таких приложений, как браузеры, поисковые роботы, почтовые клиенты.

3.3.

Атака (attack): попытка уничтожения, раскрытия, изменения, блокирования, кражи, получения несанкционированного доступа к активу или его несанкционированного использования.

[ГОСТ Р ИСО/МЭК 27000-2021, пункт 3.2]

3.4.

Аутентификация (authentication): действия по проверке подлинности субъекта доступа и/или объекта доступа, а также по проверке принадлежности субъекту доступа и/или объекту доступа предъявленного идентификатора доступа и аутентификационной информации.

Примечание. Аутентификация рассматривается применительно к конкретному субъекту доступа и/или конкретному объекту доступа.

[ГОСТ Р 58833-2020, пункт 3.4]

3.5.

Взаимная аутентификация (mutual authentication): обоюдная аутентификация, обеспечивающая для каждого из участников процесса аутентификации – и субъекту доступа, и объекту доступа – уверенность в том, что другой участник процесса аутентификации является тем, за кого себя выдает.

[ГОСТ Р 58833-2020, пункт 3.10]

3.6.

Владелец ресурса (resource owner): субъект, способный предоставить доступ к защищенному ресурсу. [16]

3.7.

Доступ (access): получение одной стороной информационного взаимодействия возможности использования ресурсов другой стороны информационного взаимодействия.

Примечания:

1. В качестве ресурсов стороны информационного взаимодействия, которые может использовать другая сторона информационного взаимодействия, рассматриваются информационные ресурсы, вычислительные ресурсы средств вычислительной техники и ресурсы автоматизированных (информационных) систем, а также средства вычислительной техники и автоматизированные (информационные) системы в целом.

2. Доступ к информации – возможность получения информации и ее использования.

[ГОСТ Р 58833-2020, пункт 3.17]

3.8.

Защищенный ресурс (protected resource): ресурс с ограниченным доступом. [16]

3.9.

Идентификация (identification): действия по присвоению субъектам и объектам доступа идентификаторов и/или по сравнению предъявляемого идентификатора с перечнем присвоенных идентификаторов.

[Р 50.1.053-2005, пункт 3.3.9]

3.10.

Клиент (client): приложение, целью которого является получение доступа к защищенным ресурсам пользователя от его имени после выполнения процедуры авторизации. [16]

Примечание. Термин «клиент» не подразумевает каких-либо конкретных характеристик реализации (например, выполняется ли приложение на сервере, настольном компьютере или других устройствах).

3.11.

Ключ подписи (signature key): элемент секретных данных, специфичный для субъекта и используемый только данным субъектом в процессе формирования цифровой подписи.

[ГОСТ Р 34.10-2012, пункт 3.1.2]

3.12.

Ключ проверки подписи (verification key): элемент данных, математически связанный с ключом подписи и используемый проверяющей стороной в процессе проверки цифровой подписи.

[ГОСТ Р 34.10-2012, пункт 3.1.3]

3.13.

Код авторизации (authorization code): свидетельство, подтверждающее факт успешной аутентификации пользователя и предоставления клиенту права доступа к определенному ресурсу сервера ресурсов. Код авторизации является промежуточной формой разрешения

на доступ клиента к определенному ресурсу, которое обменивается клиентом на токен доступа [5]

3.14.

Код аутентификации [сообщения] (message authentication code): специальный набор символов, добавляемый к сообщению и предназначенный для обеспечения его целостности и аутентификации источника данных. [14]

3.15.

Код аутентификации сообщения на основе хэш-функции (hash-based message authentication code): механизм обеспечения аутентичности информации на основе симметричного ключа, построенный с использованием хэш-функции.

[Р 50.1.113-2016, пункт 3.1.1]

3.16.

Конечная точка (endpoint): адрес (как правило, URL) сетевого ресурса, через который осуществляется доступ к определенному сервису. [16]

Примечание. В процессе авторизации технология OAuth 2.0 использует две конечные точки сервера авторизации (ресурсы HTTP) – конечную точку авторизации и конечную точку токена – и одну конечную точку клиента.

3.17.

Конечная точка авторизации (authorization endpoint): конечная точка, используемая клиентом для получения авторизации от владельца ресурса посредством перенаправления агента пользователя. [16]

3.18.

Конечная точка клиента (client endpoint): конечная точка, на которую сервер авторизации возвращает ответы клиенту посредством агента пользователя. [16]

3.19.

Конечная точка токена (token endpoint): конечная точка, используемая клиентом для обмена разрешения на доступ на токен доступа, обычно с аутентификацией клиента. [16]

3.20.

Конечная точка UserInfo (UserInfo endpoint): защищенный ресурс, который при предоставлении клиентом токена доступа возвращает авторизованную информацию о конечном пользователе [17]

3.21.

Конечный пользователь (end user): владелец ресурса в случае, если он является человеком. [16]

3.22.

Конфиденциальный клиент (confidential client): клиент, который может обеспечить конфиденциальность своих учетных данных (например, клиент, реализованный на защищенном сервере с ограниченным доступом к учетным данным клиента) или может выполнить безопасную аутентификацию клиента с использованием других средств. [16]

3.23.

[Криптографический] ключ (key): изменяемый параметр в виде последовательности символов, определяющий криптографическое преобразование.

[ГОСТ Р 34.12-2015, пункт 2.1.8]

3.24.

Нативное приложение (native application): приложение, которое пользователь устанавливает на свое устройство, в отличие от веб-приложения, работающего только в контексте браузера. [16]

Примечание. Приложения, реализованные с использованием веб-технологий, но распространяемые как нативные приложения, так называемые гибридные приложения, считаются эквивалентными нативным приложениям для целей данных требований.

3.25.

Объект запроса (request object): токен JWT, который содержит набор параметров запроса аутентификации в качестве своих заявленных свойств. [17]

3.26.

Односторонняя аутентификация (one-way authentication): аутентификация, обеспечивающая только лишь для одного из участников процесса аутентификации (объекта доступа) уверенность в том, что другой участник процесса аутентификации (субъект доступа) является тем, за кого себя выдает предъявленным идентификатором доступа.

[ГОСТ Р 58833-2020, пункт 3.36]

3.27.

Параметр, заявленное свойство (claim): часть информации, заявленной о субъекте. Заявленное свойство представлено в виде пары имя/значение, состоящей из имени заявленного свойства (параметра) и значения заявленного свойства (параметра). [18]

3.28.

Процесс проверки подписи (verification process): процесс, в качестве исходных данных которого используются подписанное сообщение, ключ проверки подписи и параметры схемы ЭЦП, результатом которого является заключение о правильности или ошибочности цифровой подписи.

[ГОСТ Р 34.10-2012, пункт 3.1.8]

3.29.

Процесс формирования подписи (signature process): процесс, в качестве исходных данных которого используются сообщение, ключ подписи и параметры схемы ЭЦП, а в результате формируется цифровая подпись.

[ГОСТ Р 34.10-2012, пункт 3.1.9]

3.30.

Прикладной программный интерфейс (application program interface, API): интерфейс между прикладным программным средством и прикладной платформой, через который обеспечивается доступ ко всем необходимым службам (услугам).

[Р 50.1.041-2002, пункт 3.1.5]

3.31.

Протокол аутентификации (authentication protocol): протокол, позволяющий участникам процесса аутентификации осуществить аутентификацию.

Примечание. Протокол реализует алгоритм (правила), в рамках которого субъект доступа и объект доступа последовательно выполняют определенные действия и обмениваются сообщениями.

[ГОСТ Р 58833-2020, пункт 3.47]

3.32.

Публичный клиент (public client): клиент, который не может обеспечить конфиденциальность своих учетных данных (например, клиент, выполняющийся на устройстве, используемом владельцем ресурса, таком как установленное нативное приложение или приложение на основе веб-браузера) и не может выполнить безопасную аутентификацию клиента с помощью других средств. [16]

3.33.

Разрешение на доступ (authorization grant): свидетельство, подтверждающее авторизацию владельца ресурса (для доступа к его защищенным ресурсам), которое далее используется клиентом для получения токена доступа. [16]

3.34.

Сервер авторизации (authorization server): сервер, выдающий клиенту токены доступа после успешной аутентификации владельца ресурса и прохождения процедуры авторизации. [16]

3.35.

Сервер ресурсов (resource server): сервер, на котором размещены защищенные ресурсы, способный принимать и отвечать на запросы к защищенным ресурсам с использованием токенов доступа. [16]

3.36.

Токен доступа (access token): свидетельство, подтверждающее факт авторизации доступа к определенному ресурсу, выданное определенному клиенту сервером авторизации с одобрения владельца ресурса. Токен доступа указывает на конкретные области данных, к которым разрешен доступ, длительность доступа и другие параметры. [5]

3.37.

Токен обновления (refresh token): символьная строка, используемая для получения токенов доступа. Токен обновления выдается клиенту сервером авторизации и используется для получения нового токена доступа, когда текущий токен доступа становится недействительным или истекает его срок действия, или для получения дополнительных токенов

доступа с идентичной или более узкой областью действия (токены доступа могут иметь более короткий срок службы и меньше разрешений на доступ, чем разрешено владельцем ресурса). [16]

3.38.

Хэш-код (hash-code): строка бит, являющаяся выходным результатом хэш-функции.

[ГОСТ Р 34.10-2012, пункт 3.1.13]

3.39.

Хэш-функция (collision-resistant hash-function): функция, отображающая строки бит в строки бит фиксированной длины и удовлетворяющая следующим свойствам:

1) по данному значению функции сложно вычислить исходные данные, отображаемые в это значение;

2) для заданных исходных данных сложно вычислить другие исходные данные, отображаемые в то же значение функции;

3) сложно вычислить какую-либо пару исходных данных, отображаемых в одно и то же значение.

[ГОСТ Р 34.10-2012, пункт 3.1.14]

3.40.

[Электронная цифровая] подпись (signature): строка бит, полученная в результате процесса формирования подписи.

[ГОСТ Р 34.10-2012, пункт 3.1.15]

3.41.

Шифрование с имитозащитой и ассоциированными данными (Authenticated Encryption with Associated Data, AEAD): режим работы блочного шифра, который обеспечивает шифрование и имитозащиту открытого текста. При этом часть открытого текста (дополнительные имитозащищаемые данные, AAD) не шифруется. [5]

3.42.

Base64-кодирование (Base64-encoding): стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит алфавитно-цифровые латинские символы A–Z, a–z и 0–9 (62 знака) и 2 дополнительных символа, зависящих от системы реализации. [20]

3.43.

Base64url-кодирование (Base64url-encoding): Base64-кодирование строки байт с использованием набора символов, допускающих использование результата кодирования в качестве имени файла или URL. [20]

3.44.

ID токен, токен идентификации (ID Token): JSON веб-токен, который содержит параметры аутентификации конечного пользователя сервером авторизации. Может содержать также другие параметры. [18]

3.45.

JSON веб-токен (JWT): строка, представляющая набор параметров в формате объекта JSON, который закодирован в виде структуры JWS или JWE. При этом параметры объекта JSON сопровождаются цифровой подписью, кодом аутентификации и/или шифрованием. [18]

4. ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API (Application Program Interface) – прикладной программный интерфейс

ASCII (American Standard Code for Information Interchange) – стандарт США 8-битного кодирования некоторого набора печатных и непечатных символов

CSRF (Cross Site Request Forgery) – межсайтовая подделка запроса

DER (Distinguished Encoding Rules) – отличительные правила кодирования структур ASN.1

FAPI (Financial-grade API) – API обеспечения безопасности финансовых сервисов

HTTP (Hypertext Transfer Protocol) – протокол передачи гипертекстовых сообщений

HTTPS (Hypertext Transfer Protocol Secure) – протокол защищенной передачи гипертекстовых сообщений

HMAC (Hash-based Message Authentication Code) – код аутентификации сообщения на основе хэш-функции

IANA (Internet Assigned Numbers Authority) – Агентство по выделению имен и уникальных параметров протоколов Internet (Администрация адресного пространства Интернет)

JARM (JWT Secured Authorization Response Mode for OAuth 2.0) – защищенный с использованием токена JWT режим ответа на запрос авторизации в OAuth 2.0

JOSE (JavaScript Object Signing and Encryption) – технология подписи и шифрования объектов JavaScript

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript (нотация объектов JavaScript)

JWE (JSON Web Encryption) – структура данных в формате JSON, представляющая зашифрованное и защищенное от модификации сообщение

JWK (JSON Web Key) – структура данных в формате JSON, представляющая криптографический ключ

JWS (JSON Web Signature) – структура данных в формате JSON, представляющая сообщение с цифровой подписью или кодом аутентификации сообщений

JWT (JSON Web Token) – токен доступа, основанный на формате JSON

MAC (Message Authentication Code) – код аутентификации сообщения

MTLS (Mutual TLS) – процесс, в соответствии с которым при согласовании сеанса TLS выполняется взаимная аутентификация сервера TLS и клиента, а также подтверждение владения соответствующими ключами

Nested JWT – JWT, в котором используются вложенные подпись и/или шифрование. В Nested JWT вложенная структура JWT используется в качестве полезной нагрузки или значения открытого текста вмещающей ее структуры JWS или JWE соответственно

OAuth – открытый протокол (схема) авторизации

OIDC (OpenID Connect Core) – семейство протоколов, являющихся расширением протоколов OAuth 2.0, позволяющих расширить их функционал путем более точного описания процесса аутентификации владельца ресурса и возможности клиенту получить информацию о нем

OpenID Foundation – некоммерческая организация, созданная для управления авторскими правами, товарными знаками, маркетинговыми компаниями и другой деятельностью, связанной с сообществом OpenID

OpenID – открытый стандарт децентрализованной системы аутентификации

PKI (Public Key Infrastructure) – инфраструктура открытых ключей

RFC (Request for Comments) – предложения для обсуждения; серия нормативных документов, стандартизирующих протоколы сети Интернет

TLS (Transport Layer Security) – протокол защиты транспортного уровня

URI (Uniform Resource Identifier) – унифицированный идентификатор ресурса

URL (Uniform Resource Locator) – унифицированный адрес ресурса (единый указатель ресурса)

URN (Uniform Resource Name) – унифицированное имя ресурса

UTF-8 – стандарт кодирования символов, позволяющий компактно хранить и передавать символы Unicode, используя переменное количество байтов (от 1 до 4), и обеспечивающий полную обратную совместимость с кодировкой ASCII

5. ОБЩИЕ ПОЛОЖЕНИЯ

5.1. Структура стандарта

5.1.1. В настоящем стандарте представлены требования и рекомендации для обеспечения безопасного доступа к данным в финансовых сервисах реального времени с использованием модели обмена данными REST/JSON, защищенной технологией авторизации OAuth 2.0, включая профилирующий ее протокол OpenID Connect.

Стандарт состоит из следующих частей:

– базовый профиль безопасности OpenID API, обеспечивающий средний уровень доверия к идентификации и аутентификации в соответствии с СТО БР БФБО-1.8-2024 [6] при передаче финансовой информации;

– расширенный профиль безопасности OpenID API, обеспечивающий высокий уровень доверия к идентификации и аутентификации в соответствии с СТО БР БФБО-1.8-2024 [6] при передаче финансовой информации.

В разделе 5 настоящего стандарта приведены нормативные требования (подраздел 5.2), а также общие сведения о протоколах аутентификации OpenID Connect (подразделы 5.4–5.8).

Первая часть настоящего стандарта (раздел 6) основана на документе [21], выпущенном OpenID Foundation. В этой части определяются требования к системным (безопасность на уровне транспорта и алгоритмов преобразования данных) и прикладным параметрам протокола OpenID Connect, выполнение которых является базовым для обеспечения безопасного доступа к конфиденциальной финансовой информации.

Вторая часть (раздел 7), в основе которой источник [22], представляет расширенный профиль безопасности более высокого уровня – профиль API для доступа к финансовым данным и в других аналогичных ситуациях с повышенным риском несанкционированного доступа к данным. В ней определяются меры защиты от таких атак, как фальсификация запроса аутентификации, фальсификация ответа на запрос аутентификации, включая внедрение кода авторизации, внедрение состояния и фишинг запроса токена.

Положения настоящего стандарта применяются совместно с методическими рекомендациями МР.26.2.002-2024 Технического комитета ТК 26 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколах OpenID Connect» [5], в которых приведены требования по использованию российских криптографических алгоритмов ГОСТ Р 34.10-2012 [11], ГОСТ Р 34.11-2012 [12], ГОСТ Р 34.12-2015 [13], Р 1323565.1.026-2019 [23] для реализации протоколов OpenID Connect.

5.2. Нормативные требования

5.2.1. При реализации указанных в настоящем стандарте криптографических алгоритмов (механизмов) на территории Российской Федерации должны использоваться средства криптографической защиты информации (СКЗИ), соответствующие требованиям федерального органа исполнительной власти в области обеспечения безопасности. Класс СКЗИ, используемого для реализации требований настоящего стандарта, определяется по результатам анализа

системы (модели угроз информационной безопасности), к компонентам которой применяются эти требования, в соответствии с нормативными правовыми актами Российской Федерации.

5.2.2. Вовлеченные стороны должны следовать требованиям обеспечения безопасности персональных данных, определенным нормативными правовыми актами Российской Федерации, в частности:

– Федеральным законом от 27.07.2006 № 152-ФЗ «О персональных данных» [25];

– постановлением Правительства Российской Федерации от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных» [26];

– приказом Федеральной службы безопасности (ФСБ России) от 10.07.2014 № 378 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных с использованием средств криптографической защиты информации, необходимых для выполнения установленных Правительством Российской Федерации требований к защите персональных данных для каждого из уровней защищенности» [27];

– приказом Федеральной службы по техническому и экспортному контролю (ФСТЭК России) от 18.02.2013 № 21 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных» [28].

5.2.3. Прикладное программное обеспечение, реализующее требования настоящего стандарта, должно отвечать требованиям положений Банка России от 17.08.2023 № 821-П [29], от 17.04.2019 № 683-П [30], от 20.04.2021 № 757-П [31], от 25.07.2022 № 802-П [32], от 17.10.2022 № 808-П [54] в части прохождения сертификации в системе сертификации федерального органа исполнительной власти, уполномоченного в области противодействия техническим разведкам и технической защиты информации, или оценки соответствия по требованиям к оценочному уровню доверия (ОУД) не ниже чем ОУД 4 в соответствии с пунктом 7.6 национального стандарта Российской Федерации ГОСТ Р ИСО/МЭК 15408-3-2013 [33].

Разработка программного обеспечения, реализующего требования настоящего стандарта, должна вестись с учетом требований к разработке безопасного программного обеспечения, установленных следующими документами:

– ГОСТ Р 56939-2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования» или раздел 7.4 методического документа «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций» [34];

– методический документ ФСТЭК России «Руководство по организации процесса управления уязвимостями в органе (организации)» [35];

– методический документ ФСТЭК России «Методика тестирования обновлений безопасности программных, программно-аппаратных средств» [36];

– методический документ ФСТЭК России «Методика оценки уровня критичности уязвимостей программных, программно-аппаратных средств» [37].

Должен проводиться в том числе регулярный поиск информации, связанной с уязвимостями программ, в общедоступных источниках, включая использование банка данных угроз безопасности информации ФСТЭК России.

5.3. Технология авторизации OAuth 2.0

5.3.1. OAuth 2.0 – семейство протоколов авторизации, позволяющих одному приложению – *клиенту* – получить доступ к данным другого приложения – *сервера ресурсов*. При этом клиент получает разрешение на доступ от имени пользователя – *владельца ресурса*, который владеет необходимыми учетными данными, позволяющими его аутентифицировать. Непосредственно разрешение на доступ (grant) выдает клиенту *сервер авторизации*, на котором зарегистрирован владелец ресурса. Сервер ресурсов и сервер авторизации могут совпадать.

Примечание. Сведения о спецификации OAuth 2.0 приведены в документах RFC 6749 [16] и RFC 6750 [19].

5.3.2. Схематично сценарий протокола OAuth 2.0 представлен на рис. 1. Он описывает взаимодействие между упомянутыми выше четырьмя сторонами и включает следующие шаги:

1. Клиент запрашивает авторизацию у владельца ресурса. Запрос авторизации может быть направлен владельцу ресурса напрямую (рис. 1) или косвенно, через сервер авторизации. Предпочтительным является второй вариант.

2. Клиент получает *разрешение на доступ* (grant), структуру данных, представляющую авторизацию владельца ресурса, выраженную с использованием одного из четырех типов разрешений: кода авторизации (authorization code), неявного разрешения (implicit), пароля владельца ресурса (resource owner password credentials) и учетных данных клиента (client credentials). Тип разрешения на доступ зависит от метода, используемого клиентом для запроса авторизации, и типов разрешений на доступ, поддерживаемых сервером авторизации. Типы разрешений, поддерживаемые сервером авторизации, определяются при его разработке исходя из его прикладных целей и задач. Настоящий стандарт регламентирует использование в качестве типа разрешения код авторизации.

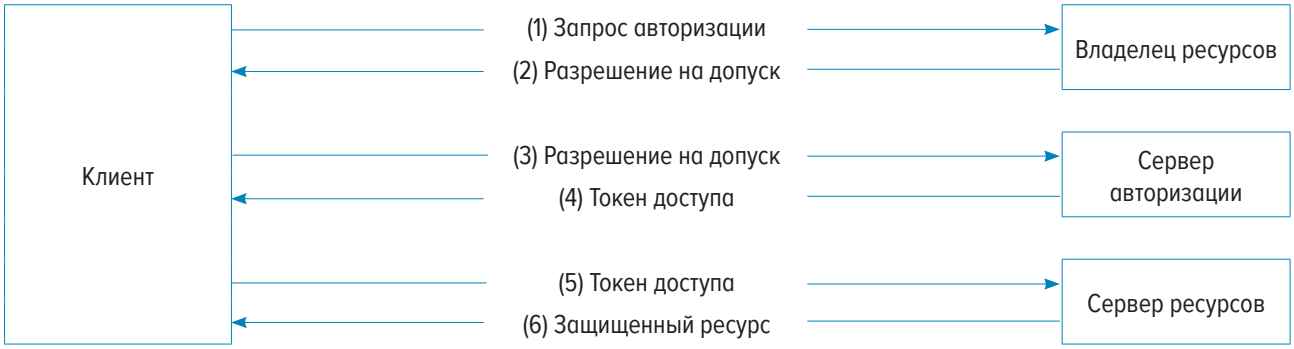
3. Клиент запрашивает токен доступа посредством аутентификации на сервере авторизации и предоставления разрешения на доступ.

4. Сервер авторизации аутентифицирует клиента, проверяет разрешение на доступ и, если оно действительно, выдает токен доступа.

5. Клиент запрашивает защищенный ресурс на сервере ресурсов и аутентифицируется, предоставляя токен доступа.

6. Сервер ресурсов проверяет токен доступа и, если он действителен, обслуживает запрос.

СЦЕНАРИЙ ПРОТОКОЛА OAUTH 2.0 *Рис. 1*



5.4. OpenID Connect

5.4.1. Протоколы OpenID Connect

5.4.1.1. OIDC – семейство протоколов, являющихся расширением протоколов OAuth 2.0, позволяющих расширить их функционал путем более точного описания процесса аутентификации владельца ресурса и возможности клиенту получить информацию о нем.

Протоколы OpenID Connect также позволяют клиенту проверять информацию о конечном пользователе на основе его (конечного пользователя) аутентификации у сервера авторизации.

5.4.1.2. Сценарий протоколов аутентификации OpenID Connect, которые используют код авторизации, выглядит следующим образом:

1. Клиент генерирует *запрос аутентификации* и, используя агент пользователя, отправляет запрос аутентификации на конечную точку авторизации сервера авторизации.

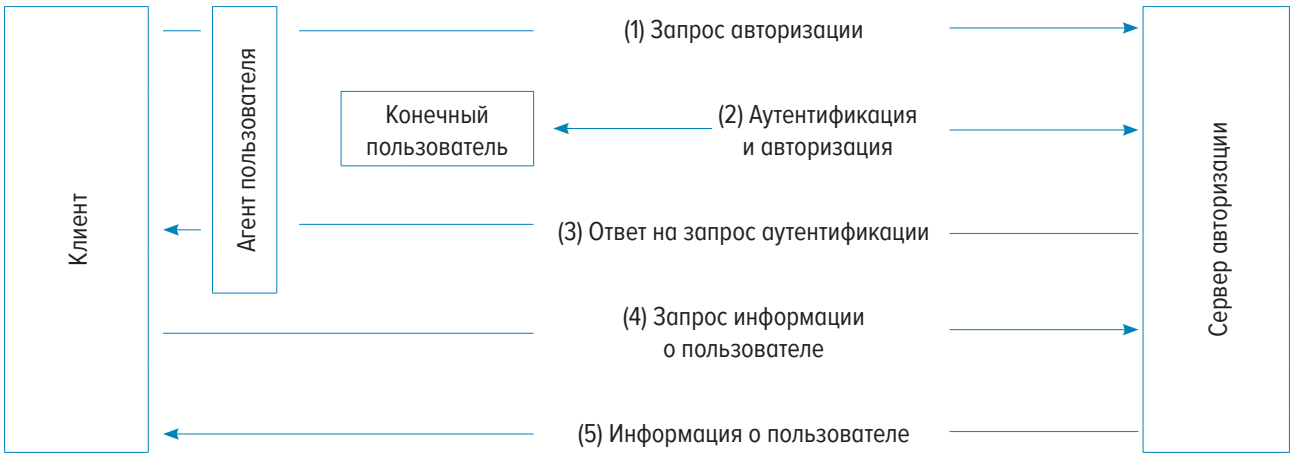
2. Сервер авторизации *аутентифицирует конечного пользователя* и получает разрешение конечного пользователя на доступ клиента к запрошенным защищенным ресурсам.

3. Сервер авторизации генерирует *код авторизации*, перенаправляет агент пользователя на клиента с кодом перенаправления 303, а также передает значение сгенерированного кода авторизации клиенту.

4. Клиент запрашивает на конечной точке токена сервера авторизации *токен доступа*, используя код авторизации.

5. Сервер авторизации формирует ответ, содержащий ID токен, токен доступа и токен обновления (опционально); клиент проверяет ID токен, получает информацию о конечном пользователе.

СЦЕНАРИЙ ПРОТОКОЛА АУТЕНТИФИКАЦИИ OPENID CONNECT *Рис. 2*



Далее клиент может отправить серверу авторизации запрос информации о пользователе, указав токен доступа, и сервер авторизации возвратит клиенту доступную информацию о конечном пользователе, авторизовавшем выдачу этого токена доступа.

5.4.1.3. Для связи сервер авторизации и клиент используют *конечные точки* – адрес (как правило, URL) сетевого ресурса, через который осуществляется доступ к определенному сервису. В процессе аутентификации используются как минимум две конечные точки сервера

авторизации – *конечная точка авторизации* (шаги (1) и (3) на рис. 2) и *конечная точка токена* (шаги (4) и (5) на рис. 2), а также одна *конечная точка клиента*. Для получения доступной информации о конечном пользователе может использоваться также конечная точка UserInfo сервера авторизации.

5.4.1.4. Сервер авторизации должен поддерживать использование метода HTTP GET на конечной точке авторизации и может также поддерживать использование метода POST. При осуществлении запросов токена доступа клиент должен использовать метод HTTP POST. Метод HTTP POST используется для отправки на сервер авторизации данных в теле запроса.

Передача сообщений между клиентом и сервером авторизации должна производиться с использованием протокола TLS.

5.4.1.5. Настоящий стандарт регламентирует использование сервером авторизации и клиентом OpenID Connect двух сценариев аутентификации: с генерацией кода авторизации (Authorization Code Flow [17]) и гибридный сценарий (Hybrid Flow [17]).

Примечание. В методических рекомендациях МР.26.2.002-2024 ТК 26 [5] принято другое именование сценариев протокола аутентификации OIDC:

– режим 1: базовый сценарий протокола OpenID Connect с генерацией кода авторизации (Authorization Code Flow [17]);

– режим 2: сценарий протокола OpenID Connect с генерацией кода авторизации (Authorization Code Flow [17]) и передачей ответа на запрос аутентификации с цифровой подписью сервера авторизации в формате JWT (режим JARM, пункт 5.4.5);

– режим 3: гибридный сценарий протокола OpenID Connect (Hybrid Flow [17]) с передачей ответа на запрос аутентификации с цифровой подписью сервера авторизации в формате ID токена.

5.4.2. Сценарий аутентификации с генерацией кода авторизации

5.4.2.1. Сценарий протокола аутентификации OpenID Connect с генерацией кода авторизации (режимы 1 и 2 [5]) требует выполнения следующих действий:

1. Клиент генерирует *запрос аутентификации* и, используя агент пользователя, отправляет запрос аутентификации на конечную точку авторизации сервера авторизации (подпункт 5.4.2.2).

2. Сервер авторизации *аутентифицирует конечного пользователя* и получает разрешение конечного пользователя на доступ клиента к запрошенным защищенным ресурсам (подпункты 5.4.2.5 и 5.4.2.6).

3. Сервер авторизации генерирует *код авторизации* и (при необходимости) подпись в формате JARM (режим 2 [5]), перенаправляет агент пользователя на клиента с кодом перенаправления 303, а также передает значение сгенерированного кода авторизации и (при необходимости) подписи в формате JARM (режим 2 [5]) на конечную точку клиента по адресу URI-переадресации `<redirect_uri>`, указанному в запросе аутентификации (подпункт 5.4.2.9).

4. Клиент запрашивает *токен доступа*, используя код авторизации, на конечной точке токена сервера авторизации (подпункт 5.4.2.10).

5. Сервер авторизации проверяет запрос токена и формирует ответ, содержащий ID токен, токен доступа и опционально токен обновления (подпункты 5.4.2.11 и 5.4.2.12).

6. Клиент получает ответ, содержащий ID токен, токен доступа и опционально токен обновления (подпункт 5.4.2.13), проверяет ID токен (пункт 6.7.2 [5]) и, если необходимо, получает информацию о конечном пользователе (подпункт 5.4.2.17).

5.4.2.2. *Запрос аутентификации* клиента включает следующие параметры (подраздел 6.2 [5]):

– `<scope>`: (обязательный) область запроса; определяет перечень свойств защищаемых данных конечного пользователя, к которым запрошен доступ; параметр `<scope>` должен содержать значение `"openid"`;

– `<response_type>`: (обязательный) тип ответа и сценарий протокола авторизации; в данном сценарии используется следующее значение:

- `"code"` – возвращает код авторизации; используется в сценарии аутентификации с генерацией кода авторизации (режимы 1 и 2 [5]);

– `<client_id>`: (обязательный) идентификатор клиента, полученный при регистрации на сервере авторизации;

– `<redirect_uri>`: (обязательный) URI-переадресации, на который будет отправлен ответ;

– `<state>`: (обязательный) строковое значение, используемое для синхронизации состояния между запросом и обратным вызовом; используется для защиты от атак межсайтовых запросов (CSRF) (пункт 6.2.5 [5]); генерируется как случайная строка длины не менее 20 байт;

– `<nonce>`: (обязательный) случайное строковое значение, используемое для связывания запроса аутентификации с ID токеном и для защиты от атак повторного воспроизведения; генерируется как случайная строка длины не менее 20 байт;

– `<code_challenge>`: (обязательный) запрос подтверждения кода по технологии PKCE (подпункт 5.4.2.4);

– `<code_challenge_method>`: (обязательный) метод вычисления `<code_challenge>` на основе `<code_verifier>`; возможное значение: `"St256"` (подпункт 5.4.2.4);

– `<request>`: (опциональный) объект запроса; позволяет передавать параметры запроса аутентификации с цифровой подписью или кодом аутентификации клиента в форме JWT (подпункт 5.4.2.3);

– `<request_uri>`: (опциональный) позволяет передавать параметры запроса аутентификации по ссылке, а не по значению. Значением `<request_uri>` является URL-адрес, использующий https-схему со ссылкой на ресурс, содержащий значение объекта запроса в форме JWT;

– `<response_mode>`: (опциональный) сообщает серверу авторизации о механизме, который будет использоваться для возвращения параметров из конечной точки авторизации; значением параметра должно быть одно из значений, указанных в параметре `<response_modes_supported>` метаданных сервера авторизации (подпункт 5.4.4.1);

– `<prompt>`: (опциональный) список строк, которые указывают, должен ли сервер авторизации запрашивать у конечного пользователя повторную аутентификацию и согласие на доступ клиента к ресурсу. Определены значения:

- `"none"` – не требуется интерфейс пользователя,
- `"login"` – серверу авторизации рекомендуется запросить повторную аутентификацию,

- "consent" – серверу авторизации рекомендуется запросить у пользователя согласие на доступ к ресурсу,
- "select_account" – серверу авторизации рекомендуется запросить у пользователя выбор учетной записи;

– <max_age>: (опциональный) максимальный срок аутентификации, определяет допустимое время в секундах, прошедшее с момента последней активной аутентификации конечного пользователя сервером авторизации. Если истекшее время больше этого значения, сервер авторизации должен пытаться активно повторно аутентифицировать конечного пользователя. Если в запросе аутентификации присутствует параметр <max_age>, возвращаемый ID токен должен включать значение параметра <auth_time>;

– <acr_values>: (опциональный) запрашиваемые значения ссылки на класс контекста аутентификации конечного пользователя; значением является разделенная пробелами строка, определяющая значения <acr>, которые сервер авторизации может использовать для обработки данного запроса аутентификации (значения располагаются в порядке предпочтения) в соответствии с [6]; класс контекста аутентификации, которому соответствует выполненная аутентификация пользователя, должен возвращаться в качестве значения параметра <acr> ID токена.

Значения параметра <acr_values> и их толкование должны устанавливаться при проектировании сервера авторизации. Значения параметров должны соответствовать уровням доверия СТО БР БФБО-1.8-2024 [6]. Например, могут использоваться следующие значения этого параметра:

– "urn:rubanking:sca": идентификатор, определяющий контекст аутентификации, указывающий на применение усиленной (двухфакторной) аутентификации пользователя,

– "urn:rubanking:ca": идентификатор, определяющий контекст аутентификации, указывающий на применение однофакторной аутентификации пользователя.

Примечание. Дополнительные сведения по параметрам запроса аутентификации приведены в RFC 6749 (пункт 4.1.1 [16]) и в спецификациях OIDC (подпункт 3.1.2.1 [17]).

5.4.2.3. Также параметры запроса аутентификации могут быть переданы в качестве параметров (claims) JSON *объекта запроса* – JWT токена (пункт 6.2.4 [5]). JWT токен должен быть подписан и может быть зашифрован. JWT токен передается в кодировании Base64url (раздел 5 [20]) по значению через параметр <request>.

Примечание. Дополнительные сведения по передаче запроса авторизации от клиента к серверу авторизации приведены в пункте 3.1.2 [17] и разделе 6 [17].

5.4.2.4. PKCE

Технология PKCE (Proof Key for Code Exchange, ключ подтверждения передачи кода) предназначена для защиты от атаки копирования кода авторизации вредоносным программным обеспечением устройства, на котором исполняется агент пользователя, с последующим воспроизведением его в ложном запросе токена.

Действия по защите от такой атаки выполняются следующим образом (пункт 6.2.6 [5]):

– при формировании запроса аутентификации клиент генерирует случайную последовательность и формирует ее символьное представление <code_verifier>;

– клиент вычисляет значение параметра <code_challenge> в соответствии с выбранным методом <code_challenge_method>;

– если значение параметра <code_challenge> отсутствует или если сервер авторизации не поддерживает указанный <code_challenge_method>, сервер авторизации должен ответить соответствующим сообщением об ошибке;

– в составе запроса токена клиент передает серверу авторизации сохраненное ранее значение <code_verifier>;

– сервер авторизации, получив запрос токена, сравнивает полученное ранее в запросе аутентификации значение <code_challenge> с вычисленным на основе значения <code_verifier>, полученного в запросе токена;

– при несовпадении этих значений делается вывод о возможном навязывании кода авторизации злоумышленником; клиенту передается сообщение об ошибке.

Примечание. Дополнительные сведения о вычислении и проверке параметров технологии PKCE приведены в RFC7636 [38].

5.4.2.5. Если запрос аутентификации действителен, сервер авторизации пытается аутентифицировать конечного пользователя или определяет, аутентифицирован ли конечный пользователь (подраздел 6.3 [5]). Методы, используемые сервером авторизации для аутентификации конечного пользователя (например, имя пользователя и пароль, файлы cookie-сеанса и так далее), не регламентируются настоящим стандартом.

Сервер авторизации должен запрашивать аутентификацию конечного пользователя в следующих случаях:

– конечный пользователь еще не аутентифицирован;

– в запросе аутентификации указан параметр <max_age> и время, прошедшее с момента аутентификации конечного пользователя, превышает значение <max_age>;

– запрос аутентификации содержит параметр <prompt> со значением "login". В этом случае сервер авторизации должен повторно аутентифицировать конечного пользователя, даже если конечный пользователь уже аутентифицирован.

Сервер авторизации не должен взаимодействовать с конечным пользователем в следующем случае:

– запрос аутентификации содержит параметр <prompt> со значением "none". В этом случае сервер авторизации должен возвратить ошибку, если конечный пользователь еще не прошел аутентификацию или не может быть аутентифицирован в режиме без вывода сообщений.

При взаимодействии с конечным пользователем сервер авторизации должен использовать соответствующие меры против подделки межсайтовых запросов (CSRF) в соответствии с пунктом 10.2.1 [5] и перехвата кликов (Clickjacking) в соответствии с пунктом 10.3.5 [5].

5.4.2.6. После аутентификации конечного пользователя сервер авторизации должен получить его разрешение об авторизации доступа к запрошенному ресурсу, прежде чем предоставлять информацию клиенту (пункт 6.3.2 [5]). В случае если это определено параметрами запроса, разрешение может быть получено одним из следующих способов:

– через интерактивный диалог с конечным пользователем, в ходе которого сервер авторизации отображает запрашиваемую клиентом область действия и запрашивает у конечного пользователя согласие на доступ;

– путем установления согласия с помощью условий обработки запроса;

– другими способами (например, с помощью предыдущего административного согласования).

5.4.2.7. После успешной аутентификации конечного пользователя и получения его разрешения на доступ клиента к защищенному ресурсу сервер авторизации генерирует код авторизации и передает его на конечную точку клиента, указанную в параметре `<redirect_uri>` запроса аутентификации (подраздел 6.4 [5]). Параметры *успешного ответа на запрос аутентификации*:

– `<code>`: (обязательный) значение кода авторизации;

– `<state>`: (обязательный) значение параметра `<state>` запроса аутентификации, полученного от клиента.

При использовании технологии JARM (режим 2 [5]) ответ формируется и проверяется в соответствии с пунктом 5.4.5.

5.4.2.8. В случае ошибки при проверке запроса аутентификации либо в процессе аутентификации конечного пользователя сервер авторизации отвечает клиенту с указанием информации об ошибке. Подробнее параметры и коды *сообщения об ошибке* приведены в пункте 6.4.2 [5].

5.4.2.9. Получив успешный ответ на запрос аутентификации (пункт 6.4.3 [5]), клиент, в частности, проверяет, что значение параметра `<state>`, полученное в составе ответа, совпадает со значением параметра `<state>` в запросе аутентификации, переданном клиентом.

5.4.2.10. В случае успешной проверки ответа сервера авторизации клиент формирует и отправляет на адрес конечной точки токена *запрос токена* (пункт 6.5.1 [5]). Параметры запроса токена:

– `<grant_type>`: (обязательный) тип разрешения на доступ; в данном случае значением должна быть строка `"authorization_code"`;

– `<code>`: (обязательный) значение кода авторизации, полученное от сервера авторизации;

– `<redirect_uri>`: (обязательный) URI-переадресации, на который будет отправлен ответ;

– `<client_id>`: (обязательный) идентификатор клиента, зарегистрированный сервером авторизации;

– `<code_verifier>`: (обязательный) сохраненное ранее подтверждение кода по технологии PKCE (подпункт 5.4.2.4).

Помимо запроса токена, серверу авторизации при этом должна быть передана информация, аутентифицирующая клиента в соответствии с выбранным методом аутентификации.

Примечание. Дополнительные сведения по запросу токена и процедурам его формирования и проверки указаны в пункте 4.1.3 [16] и подпункте 3.1.3.1 [17].

5.4.2.11. Сервер авторизации должен проверить запрос токена следующим образом (пункт 6.5.2 [5]):

– аутентифицировать клиента в соответствии с подразделом 5.5;

– убедиться, что код авторизации был выдан данному аутентифицированному клиенту;

– убедиться, что код авторизации действителен;

– убедиться, что код авторизации не использовался ранее;

– проверить, что значение <code_verifier> соответствует ранее полученному значению параметра <code_challenge> (пункт 6.2.8 [5] и подпункт 5.4.2.4);

– проверить, что значение параметра <redirect_uri> совпадает со значением параметра <redirect_uri>, которое было включено в начальный запрос аутентификации;

– убедиться, что использованный код авторизации был выдан в ответ на запрос аутентификации OpenID Connect.

5.4.2.12. Получив и успешно проверив запрос токена, сервер авторизации генерирует ID токен, токен доступа, а также, если необходимо, токен обновления и возвращает их клиенту как JSON-объекты в теле HTTP-ответа с кодом состояния 200 (OK) по адресу <redirect_uri> (пункт 6.6.1 [5]). В ответе используется тип содержимого "application/json". Ответ должен включать следующие поля и значения заголовка HTTP-ответа:

```
Cache-Control: no-store
Pragma: no-cache
```

Параметры JSON структуры *ответа на запрос токена*:

– <access_token>: (обязательный) токен доступа;

– <token_type>: (обязательный) тип токена доступа; может иметь значение "Bearer";

– <expires_in>: (обязательный) срок действия токена доступа в секундах;

– <refresh_token>: (опциональный) токен обновления;

– <scope>: (опциональный) область свойств токена доступа; определяет подмножество области доступа запроса авторизации <scope>, к которому разрешен доступ сервером авторизации;

– <id_token>: (обязательный) ID токен, связанный с фактом разрешения на доступ к ресурсу.

В случае ошибки проверки запроса токена сервер авторизации должен ответить сообщением об ошибке. Параметры и формат сообщения об ошибке приведены в подпункте 5.4.2.8. В теле HTTP-ответа используется тип содержимого "application/json" с HTTP-ответом с кодом состояния 400.

Примечание. Дополнительные сведения по параметрам ответа на запрос токена и процедурам работы с ними (подпункты 3.1.3.3 и 3.1.3.4 [17]).

5.4.2.13. Клиент должен проверить правильность ответа на запрос токена следующим образом (пункт 6.6.2 [5]):

– клиент должен игнорировать нераспознанные параметры ответа;

– клиент должен прекратить проверку, если отсутствует значение хотя бы одного из обязательных параметров;

– клиент должен проверить длины параметров с установленными в системе;

– клиент должен проверить правильность ID токена согласно пункту 6.7.2 [5].

5.4.2.14. *ID токен* – JSON веб-токен (JWT) (подраздел 5.7), который содержит параметры аутентификации конечного пользователя сервером авторизации.

ID токен включает по крайней мере следующие параметры (пункт 6.7.1 [5]):

– <iss>: (обязательный) идентификатор эмитента (сервера авторизации), источника ответа;

– <sub>: (обязательный) уникальный идентификатор субъекта, выданный сервером авторизации конечному пользователю;

– <aud>: (обязательный) аудитория, для которой предназначен данный ID токен; должна содержать идентификатор <client_id>;

– <exp>: (обязательный) срок действия; время, при наступлении и после наступления которого ID токен не должен приниматься к обработке;

– <nbf>: (опциональный) время, до которого ID токен не должен приниматься к обработке;

– <iat>: (обязательный) время, когда был выпущен ID токен; значением является число в формате JSON, представляющее количество секунд от 1970-01-01T0:0:0Z UTC до соответствующей даты/времени;

– <auth_time>: (обязательный, если в запросе аутентификации присутствует значение параметра <max_age>) время, когда выполнена аутентификация конечного пользователя;

– <nonce>: (обязательный в ответе на запрос токена) строковое значение, равное значению параметра <nonce> в запросе аутентификации;

– <acr>: (опциональный) регистрозависимая строка символов, определяющая класс контекста аутентификации в соответствии с [6], которому соответствует выполненная аутентификация конечного пользователя; значение этого параметра должно выбираться из списка, указанного в <acr_values> запроса аутентификации; сторонам, использующим этот параметр, следует согласовывать смысл используемых значений, которые могут зависеть от контекста; параметр <acr> должен присутствовать в ID токене, если в запросе аутентификации указано значение параметра <acr_values>;

– <c_hash>: (обязательный) хэш-значение кода авторизации;

– <s_hash>: (обязательный в ответе на запрос аутентификации) хэш-значение параметра <state>;

– <at_hash>: (обязательный в ответе на запрос токена) хэш-значение токена доступа;

– <azp>: (опциональный) идентификатор клиента, для которого был выпущен ID токен.

Примечание. Дополнительные сведения об использовании ID токена (раздел 2 [17]).

5.4.2.15. *Токен доступа* (access token) (пункт 6.7.3 [5]) – свидетельство, подтверждающее факт авторизации доступа к определенному ресурсу, выданное определенному клиенту сервером авторизации с одобрения владельца ресурса. Токен доступа указывает на конкретные области данных (<scope>), к которым разрешен доступ, длительность доступа и другие параметры. Токен доступа рассматривается как символьная строка, не несущая смысловой нагрузки для клиента и сервера авторизации. Требования и рекомендации по безопасному применению приведены в пунктах 10.3.2, 10.3.3 и 10.3.4 [5].

Примечание. Дополнительные сведения и рекомендации по безопасному применению токенов доступа (подраздел 16.18 [17]) (подраздел 3.1 и пункт 5.1.5 [53]).

5.4.2.16. *Токен обновления* (refresh token) (пункт Б.10 [5]) – символьная строка, используемая для получения токенов доступа. Токен обновления выдается клиенту сервером авторизации и используется для получения нового токена доступа, когда текущий токен доступа становится недействительным, или истекает его срок действия, или для получения дополнительных токенов доступа с идентичной или более узкой областью действия (раздел 12 [17]).

Данная функциональность не регламентируется настоящим документом и определяется на этапе разработки сервера авторизации исходя из его прикладных целей и задач.

5.4.2.17. Клиент может получить информацию о конечном пользователе либо с помощью извлечения ее из параметров ID токена, либо с помощью запроса к конечной точке UserInfo сервера авторизации (пункт Б.6 [5]).

Чтобы получить значения параметров информации о конечном пользователе, клиент может отправить запрос конечной точке UserInfo, используя токен доступа, полученный при аутентификации на сервере авторизации. Значения параметров в ответ возвращаются в виде JSON-объекта, который содержит набор пар – имя и значение параметра.

Набор доступных клиенту параметров информации о конечном пользователе определяется на этапе проектирования сервера авторизации.

Примечания:

1. Настоящий документ не регламентирует функциональность уточнения информации о конечном пользователе. В случае ее реализации должен быть проведен анализ безопасности в соответствии с установленным законодательством Российской Федерации порядком.

2. Дополнительные сведения об алгоритмах работы в конечной точке UserInfo представлены в подразделе 5.3 [17].

5.4.3. Гибридный сценарий аутентификации

5.4.3.1. Гибридный сценарий протокола аутентификации (режим 3, пункт 6.1.2 [5]) состоит из последовательности действий, аналогичных сценарию с генерацией кода авторизации (пункт 5.4.2). Последовательность действий различается только на шаге 3:

> «3. Сервер авторизации генерирует код авторизации и ID токен, перенаправляет агент пользователя на клиента с кодом перенаправления 303, а также передает значения сгенерированного кода авторизации и ID токена на конечную точку клиента по адресу URI-переадресации `<redirect_uri>`, указанному в запросе аутентификации (подпункт 5.4.2.9)».

5.4.3.2. При использовании гибридного сценария действия сервера авторизации и клиента, связанные с формированием запроса аутентификации, его передачей, проверкой, а также аутентификацией пользователя и получением его согласия, в целом те же, что и в сценарии с генерацией кода авторизации (подпункты 5.4.2.2–5.4.2.6). При этом параметр `<response_type>` запроса аутентификации должен иметь значение: `"code id_token"`.

5.4.3.3. В случае ошибки проверки запроса аутентификации либо конечный пользователь отклоняет запрос, либо аутентификация конечного пользователя завершается неудачно, сервер авторизации должен возвратить клиенту ответ об ошибке как в пункте 6.4.2 [5].

В случае успешного ответа на запрос, помимо параметров ответа на запрос аутентификации (подпункт 5.4.2.9), клиенту возвращается также значение параметра `<id_token>` – (обязательный) ID токен.

Формат ID токена в данном сценарии аналогичный подпункту 5.4.2.14. В составе ID токена на этом этапе отсутствует параметр `<at_hash>`.

5.4.3.4. Клиент должен убедиться в правильности ответа аутентификации, следуя правилам пункта 6.4.3 [5], при этом обязательно проверив целостность полученного кода авторизации с помощью параметра <c_hash>.

5.4.3.5. При использовании гибридного сценария запрос токена клиентом, проверка запроса токена, генерация токена доступа и ID токена на конечной точке токена, их проверка клиентом, а также реакция на ошибки выполняются в соответствии с аналогичными действиями в сценарии с генерацией кода авторизации в соответствии с подпунктами 5.4.2.10–5.4.2.13.

Поскольку при этом ID токен возвращается как из конечной точки авторизации, так и из конечной точки токена, значения <iss> и <sub> должны быть идентичны в обоих ID токенах. Все значения параметров события аутентификации, присутствующие в любом из них, должны присутствовать в обоих ID токенах. Если какой-либо ID токен содержит параметр конечного пользователя, присутствующий в обоих ID токенах, он должен иметь одинаковые значения в обоих ID токенах. Сервер авторизации может вернуть меньшее количество параметров о конечном пользователе из конечной точки авторизации – например, по соображениям конфиденциальности.

Примечание. Дополнительные сведения о гибридном сценарии аутентификации представлены в подразделе 3.3 [17].

5.4.4. Регистрация сервера авторизации (Discovery) и динамическая регистрация клиента

5.4.4.1. Метаданные сервера авторизации

Чтобы предоставить доступ клиентам к сервису аутентификации и авторизации, сервер авторизации должен обеспечить безопасную доставку каждому клиенту метаданных сервера авторизации, описывающих его адрес и параметры доступа. Настоящий документ не определяет конкретный способ передачи метаданных сервера авторизации.

Примечание. Для публикации и поиска метаданных сервера авторизации может использоваться процедура OpenID Connect Discovery [40] или [41]. В случае ее реализации требуется проведение исследований ее безопасности в соответствии с установленным законодательством Российской Федерации порядком. Передача сообщений Discovery, в том числе сообщений используемого там сервиса WebFinger [39], между клиентом и сервером авторизации должна быть защищена с помощью протокола TLS с обеспечением конфиденциальности и целостности. При этом должна выполняться аутентификация источника информации Discovery.

5.4.4.2. В настоящем стандарте используются следующие метаданные сервера авторизации (подраздел 5.4 [5]):

– <issuer>: (обязательный) https URL-адрес, являющийся уникальным идентификатором эмитента (Issuer Identifier); это значение должно быть указано в качестве параметра <iss> в ID токенах, выданных данным сервером авторизации;

– <authorization_endpoint>: (обязательный) URI конечной точки авторизации;

– <token_endpoint>: (обязательный) URI конечной точки токена;

– <userinfo_endpoint>: (рекомендуемый) URI конечной точки UserInfo, предназначенный для запроса информации об аутентифицированном конечном пользователе (подпункт 5.4.2.17);

– <registration_endpoint>: (рекомендуемый) URI конечной точки динамической регистрации клиентов сервера авторизации (подпункт 5.4.4.3 и [42], [43]);

– <jwks_uri>: (обязательный) URL документа Key Set (подпункт 5.7.3.3) сервера авторизации, где публикуются его ключи в формате JWK; публикуемые сертификаты должны передаваться доверенным образом;

– <grant_types_supported>: (опциональный) перечень поддерживаемых сервером авторизации типов разрешений на доступ (grants); сервера авторизации, соответствующие настоящему стандарту, должны поддерживать только значение "authorization_code" (значение по умолчанию);

– <scopes_supported>: (обязательный) перечень значений параметра <scope> запроса аутентификации, которые поддерживает сервер авторизации;

– <response_types_supported>: (обязательный) перечень поддерживаемых значений параметра <response_type>; сервера авторизации, соответствующие настоящим рекомендациям, должны поддерживать только значения "code" и "code id_token":

- "code" – сервер авторизации возвращает код авторизации (режим 1 или 2 [5]),
- "code id_token" – сервер авторизации возвращает код авторизации и ID токен (режим 3 [5]);

– <response_modes_supported>: (опциональный) перечень поддерживаемых значений параметра <response_mode>; по умолчанию ["query", "fragment"]; при использовании технологии JARM (пункт 5.4.5) также могут присутствовать следующие значения <response_mode>: "query.jwt", "fragment.jwt", "jwt";

– <claims_supported>: (рекомендуемый) список имен параметров пользователя, значения которых сервер авторизации может предоставить клиенту;

– <service_documentation>: (опциональный) URL-адрес страницы, содержащей информацию, предоставляемую разработчиками или которую нужно знать при использовании сервера авторизации;

– <id_token_signing_alg_values_supported>: (обязательный) JSON-массив, содержащий список алгоритмов цифровой подписи JWS (значения параметра <alg>), поддерживаемых сервером авторизации для ID токена;

– <id_token_encryption_alg_values_supported>: (опциональный) JSON-массив, содержащий список алгоритмов шифрования JWE (значения параметра <alg>), поддерживаемых сервером авторизации для ID токена;

– <id_token_encryption_enc_values_supported>: (опциональный) JSON-массив, содержащий список алгоритмов шифрования JWE (значения параметра <enc>), поддерживаемых сервером авторизации для ID токена;

– <request_object_signing_alg_values_supported>: (опциональный) JSON-массив, содержащий список алгоритмов цифровой подписи JWS (значения параметра <alg>), поддерживаемых сервером авторизации для объектов запроса (подпункт 5.4.2.3);

– <request_object_encryption_alg_values_supported>: (опциональный) JSON-массив, содержащий список алгоритмов шифрования JWE (значения параметра <alg>), поддерживаемых сервером авторизации для объектов запроса;

– <request_object_encryption_enc_values_supported>: (опциональный) JSON-массив, содержащий список алгоритмов шифрования JWE (значения параметра <enc>), поддерживаемых сервером авторизации для объектов запроса.

Адреса всех перечисленных выше конечных точек сервера авторизации должны быть разными.

При реализации настоящего стандарта могут использоваться также и другие метаданные сервера авторизации.

Примечание. Дополнительные сведения о метаданных сервера авторизации приведены в RFC8414 [41].

5.4.4.3. Метаданные клиента

Клиент перед первым обращением к серверу авторизации должен безопасным образом предоставить серверу авторизации метаданные клиента, описывающие его параметры. Настоящий документ не определяет конкретный способ передачи метаданных клиента серверу авторизации.

В целях данного документа используются следующие метаданные клиента (подраздел 5.5 [5]):

– <redirect_uris>: (обязательный) перечень поддерживаемых адресов переадресации клиента; одно из зарегистрированных значений URI-переадресации должно точно соответствовать значению параметра <redirect_uri>, используемому в каждом запросе аутентификации;

– <response_types>: (опциональный) перечень поддерживаемых клиентом типов ответа; настоящие рекомендации поддерживают один тип: "code";

– <grant_types>: (опциональный) перечень поддерживаемых клиентом типов разрешений на доступ (grants) из перечня: "authorization_code", "refresh_token"; по умолчанию – "authorization_code";

– <application_type>: (опциональный) тип клиента; настоящие рекомендации поддерживают только значение "web";

– <contacts>: (опциональный) список адресов электронной почты лиц, ответственных за этого клиента;

– <client_name>: (опциональный) имя клиента; оно будет показано конечному пользователю при его аутентификации;

– <logo_uri>: (опциональный) URL-ссылка на логотип клиентского приложения;

– <client_uri>: (опциональный) URL-адрес домашней страницы клиента;

– <policy_uri>: (опциональный) URL-адрес информации о том, как будут использоваться данные профиля конечного пользователя;

– <tos_uri>: (опциональный) URL-адрес, который клиент предоставляет конечному пользователю для ознакомления с условиями обслуживания клиента; если значение <tos_uri> присутствует, сервер авторизации должен показать этот URL-адрес конечному пользователю;

– <default_max_age>: (опциональный) максимальная допустимая продолжительность периода до следующей аутентификации конечного пользователя (в секундах);

– <require_auth_time>: (опциональный) логическое значение, указывающее, требуется ли указывать параметр <auth_time> в ID токене; значением по умолчанию является "false";

– <token_endpoint_auth_method>: (обязательный) поддерживаемый метод аутентификации клиента на конечной точке токена; возможные варианты, поддерживаемые настоящими спецификациями: "client_secret_jwt", "private_key_jwt", "tls_client_auth" (подраздел 5.5);

– <jwks_uri>: (опциональный) URI документа JWK Set (подпункт 5.7.3.3), содержащего ключи клиента в формате JWK; публикуемые сертификаты должны передаваться доверенным образом;

– <jwks>: (опциональный) документ JWK Set, передаваемый по значению; параметры <jwks_uri> и <jwks> не должны одновременно присутствовать в метаданных клиента; сертификаты открытых ключей должны передаваться доверенным образом;

– <id_token_signed_response_alg>: (опциональный) алгоритм JWS (параметр <alg>) для подписания ID токена, выданного данному клиенту;

– <id_token_encrypted_response_alg>: (опциональный) алгоритм JWE (параметр <alg>) для шифрования ID токена, выданного данному клиенту. Если это значение указано, ответ будет подписан, а затем зашифрован, и результатом будет вложенный JWT. По умолчанию, если это значение опущено, шифрование не выполняется;

– <id_token_encrypted_response_enc>: (опциональный) алгоритм шифрования JWE (параметр <enc>) для шифрования ID токена, выданного данному клиенту. Если этот параметр присутствует, также должен быть указан <id_token_encrypted_response_alg>;

– <request_object_encryption_alg>: (опциональный) алгоритм (параметр <alg>), который клиент предполагает использовать для шифрования объектов запроса, отправленных серверу авторизации. Этот параметр должен включаться, когда будет использоваться симметричное шифрование. Если указано и подписание, и шифрование, объект запроса будет подписан, затем зашифрован, и результатом будет вложенный JWT. По умолчанию, если этот параметр опущен, клиент не применяет шифрование объектов запроса;

– <request_object_encryption_enc>: (опциональный) алгоритм JWE (параметр <enc>), который клиент предполагает использовать для шифрования объектов запроса, отправленных на сервер авторизации. Если присутствует данный параметр, также должен быть указан <request_object_encryption_alg>.

5.4.4.4. Если все значения метаданных клиента корректны, сервер авторизации должен безопасным образом (с обеспечением контроля целостности и аутентификации источника данных) предоставить клиенту следующую информацию:

– <client_id>: (обязательный) уникальный идентификатор клиента, который далее будет использоваться клиентом в протоколах аутентификации;

– <client_secret>: (опциональный) секретная информация, которая используется для аутентификации клиента сервером авторизации, а также для защиты взаимодействия клиента и сервера авторизации с использованием симметричных криптографических механизмов; значение этого параметра обязательно при использовании механизма аутентификации клиента <client_secret_jwt>;

– <client_id_issued_at>: (опциональный) время выдачи идентификатора клиента;

– <client_secret_expires_at>: (обязательный, если присутствует <client_secret>) срок действия; время окончания действия <client_secret> или 0, если он не устаревает никогда; значением является число, представляющее количество секунд от 1970-01-01T0:0:0Z UTC до момента окончания действия <client_secret>.

Кроме того, сервер авторизации возвращает клиенту метаданные клиента, принятые сервером и, возможно, скорректированные им.

Должна быть обеспечена конфиденциальность передачи значения параметра <client_secret>. Механизм защиты должен быть определен на этапе разработки сервера авторизации.

Также безопасным образом (с обеспечением контроля целостности и аутентификации источника данных) должен быть доставлен клиенту TLS-сертификат сервера авторизации.

Примечания:

1. Доставка метаданных от клиента серверу авторизации и обратно может быть выполнена с помощью процедуры *динамической регистрации клиента* ([42], [43]). В случае ее реализации требуется проведение исследований безопасности ее использования.

2. Передача информации между клиентом и конечной точкой регистрации сервера авторизации должна быть защищена с помощью протокола TLS с аутентификацией сервера авторизации.

5.4.5. JARM – режим ответа на основе JWT

В режиме 2 [5] протокола с генерацией кода авторизации параметры успешного ответа на запрос аутентификации или ответа об ошибке передаются от сервера авторизации к клиенту в составе структуры JWT (пункт 6.4.4 [5]). Данная технология именуется JARM.

В этом случае JWT содержит базовые параметры <iss>, <aud>, <exp>. При успешном ответе в JWT передаются также значения обязательных параметров <code> и <state>. В составе JWT могут передаваться значения других параметров в зависимости от реализации. В ответе об ошибке в составе JWT передаются значения параметров <error>, <error_description>, <error_URI>, <state>.

JWT должен быть только подписан (JWS) либо подписан и зашифрован (JWS и JWE). Шифрование позволяет обеспечить конфиденциальность значений параметров <code> и <state>, а также, возможно, другой информации, которую предполагается передавать в составе структуры JWT. В случае передачи в составе JWT других параметров требуется проведение исследований безопасности используемых параметров.

Решение о необходимости применения шифрования ответа сервера авторизации клиенту в составе JWT принимается на этапе разработки сервера авторизации.

То, как передается значение JWT в составе HTTP-ответа, определяется значением параметра <response_mode> в составе запроса аутентификации: "query.jwt" (или синоним "jwt" в случае <response_type> = "code") и "fragment.jwt" (пункт 6.4.4 [5]).

Примечание. Подробное описание технологии JARM приведено в [24].

5.5. Аутентификация клиента

Для аутентификации клиента на конечной точке токена сервера авторизации могут использоваться несколько методов. Чтобы заявить серверу авторизации о поддержке клиентом определенного метода, клиент должен зарегистрировать в качестве значения параметра <token_endpoint_auth_method> метаданных клиента (подпункт 5.4.4.3) один из следующих идентификаторов:

– "client_secret_jwt": аутентификация с использованием структуры JWT на основе кода аутентификации HMAC и <client_secret>;

– "private_key_jwt": аутентификация с использованием структуры JWT на основе цифровой подписи;

– "tls_client_auth": аутентификация с использованием TLS с взаимной аутентификацией сторон (MTLS) и PKI для связывания сертификата с клиентом.

Подробнее спецификации этих методов аутентификации клиента приведены в разделе 7 [5].

5.6. Доступ к защищенному ресурсу

Доступ клиента к защищенному ресурсу обеспечивает сервер ресурсов. Выполняя запрос, клиент передает ему полученный от сервера авторизации токен доступа. Сервер ресурсов проверяет этот токен доступа: его срок действия, присутствие запрашиваемого ресурса в области действия токена, принадлежность токена запрашивающему клиенту (при этом необходимо использовать безопасный канал связи сервера ресурса с сервером авторизации) (пункт 6.7.5 [5]).

Примечание. Примером организации такой связи между сервером ресурсов и сервером авторизации может служить протокол интроспекции токена [48]. В случае его реализации требуется проведение исследований безопасности его использования.

Канал передачи информации между клиентом и сервером ресурса должен быть защищен с помощью протокола TLS с обеспечением конфиденциальности, контроля целостности и взаимной аутентификации сторон.

Описание протокола, реализующего доступ клиента к серверу ресурсов с использованием токенов доступа, приведено в пункте 6.7.5 [5].

Примечание. Дополнительные сведения о протоколе доступа клиента к серверу ресурсов приведены в RFC 6750 [19].

5.7. JWT

5.7.1. Цифровая подпись в формате JSON

5.7.1.1. JWS (JSON Web Signature) – структура данных в формате JSON, представляющая сообщение с цифровой подписью или кодом аутентификации сообщений.

5.7.1.2. JWS состоит из трех частей:

– <JOSE Header>: JOSE заголовок – JSON-объект, содержащий параметры, описывающие используемые криптографические операции и их параметры (пункт 8.1.4 [5]);

– <JWS Payload>: функциональное содержимое JWS (полезная нагрузка) – последовательность байтов, подлежащих защите, другими словами, сообщение; функциональное содержимое может состоять из произвольной последовательности байтов;

– <JWS Signature>: цифровая подпись или код аутентификации сообщений, вычисленные на основе JOSE заголовка JWS и функционального содержимого JWS (пункт 8.1.5 [5]).

5.7.1.3. <JOSE Header> может включать следующие параметры:

– <alg>: (обязательный) идентификатор криптографического алгоритма цифровой подписи или вычисления кода аутентификации, используемого для защиты JWS;

– <kid>: (опциональный) идентификатор ключа, который используется для защиты JWS;

– <x5t#St256>: (опциональный) отпечаток сертификата ключа проверки подписи отправителя; является Base64url-представлением 256-битного значения хэш-функции ГОСТ Р 34.11-2012 [12], вычисленной для DER-кодирования X.509 сертификата ключа проверки цифровой подписи отправителя JWS;

– <typ>: (опциональный) тип (media type) всего объекта, заданного структурой JWS;

– <cty>: (опциональный) тип (media type) функционального содержимого JWS;

– <crit>: (опциональный) список имен критичных параметров заголовка;

– <seed>: случайная последовательность длины 32 байта, которая используется только с алгоритмом вычисления кода аутентификации; параметр обязательный в случае использования алгоритма вычисления кода аутентификации и не должен присутствовать в случае использования алгоритма цифровой подписи.

В зависимости от значения параметра <alg> в структуру <JOSE Header> могут включаться также другие параметры.

Точное описание JWS с указанием криптографических алгоритмов и формул вычисления/проверки цифровой подписи или кода аутентификации с использованием российских криптографических алгоритмов приведено в подразделах 8.1, 9.1, 9.2 [5].

Примечание. Дополнительные сведения о структуре JWS приведены в [45].

5.7.2. JSON структура шифрованного текста

5.7.2.1. JWE (JSON Web Encryption) – структура данных в формате JSON, представляющая зашифрованное и защищенное от модификации сообщение.

Структура JWE состоит из 5 компонентов:

– <JOSE Header>: JOSE заголовок, содержащий параметры, описывающие криптографические операции и их параметры (подпункт 5.7.2.2);

– <JWE Encrypted Key>: зашифрованный ключ защиты содержимого JWE;

– <JWE Initialization Vector>: синхропосылка (вектор инициализации), используемая для шифрования содержимого JWE;

– <JWE Ciphertext>: шифротекст JWE;

– <JWE Authentication Tag>: имитовставка JWE.

Для обеспечения конфиденциальности и целостности открытого текста, а также целостности <JOSE Header> используется алгоритм шифрования с имитозащитой и ассоциированными данными (Р 1323565.1.026-2019 [23]).

5.7.2.2. Параметры JOSE заголовка:

– <alg>: (обязательный) идентификатор криптографического алгоритма, который используется для шифрования или вычисления ключа шифрования контента;

– <enc>: (обязательный) идентификатор алгоритма шифрования, используемого для выполнения аутентифицированного шифрования открытого текста;

– <zip>: (опциональный) идентификатор алгоритма сжатия незашифрованного текста перед шифрованием;

– <kid>, <typ>, <cty>, <crit>: как в JWS (пункт 5.7.1); при этом параметр <kid> определяет открытый ключ, который был использован при формировании ключа шифрования контекста CEK, получатель может воспользоваться этой информацией, чтобы выбрать соответствующий закрытый ключ;

– <x5t#St256>: (опциональный) отпечаток сертификата открытого ключа получателя; представляет собой Base64url-представление 256-битного значения хэш-функции ГОСТ Р 34.11-2012 [12], вычисленной для DER-кодирования X.509 сертификата открытого ключа получателя JWE;

– <seed>: случайная последовательность; этот параметр обязателен при использовании механизма вычисления ключа шифрования <alg> = "GKDF256" или "VK0256" и не используется в других алгоритмах вычисления ключа шифрования (подраздел 9.1 [5]);

– <epk>: эфемерный открытый ключ отправителя в формате JWK; этот параметр обязателен при использовании механизма вычисления ключа шифрования <alg> = "VK0256" и не используется в случае других алгоритмов (подраздел 9.1 [5]).

Точное описание JWE с указанием российских криптографических алгоритмов и формул вычисления ключа, а также зашифрования/расшифрования и аутентификации содержимого приведено в подразделах 8.2, 9.1, 9.3, 9.4 [5].

Примечание. Дополнительные сведения о структуре JWE приведены в [46].

5.7.3. JSON структура ключа

5.7.3.1. JSON Web Key (JWK) [47] – структура данных в формате JSON, представляющая криптографический ключ. Параметры JWK представляют в том числе свойства ключа и значение ключа.

5.7.3.2. В состав JWK входят следующие параметры:

– <kty>: (обязательный) тип ключа; определены следующие значения:

- "EC" – ключ для криптографического алгоритма на базе эллиптической кривой,
- "oct" – последовательность байтов, используемая для представления симметричного ключа;

– <use>: (обязательный) определяет предполагаемое использование открытого ключа; определены следующие значения:

- "sig" – подпись (открытый ключ проверки цифровой подписи данных),
- "enc" – шифрование (открытый ключ вычисления ключа зашифрования/расшифрования данных);

– <key_ops>: (опциональный) идентифицирует операции, для которых предполагается использовать ключ; определены следующие значения этого параметра:

- "sign" – вычисление цифровой подписи или кода аутентификации сообщений,
- "verify" – проверка цифровой подписи или кода аутентификации сообщений,

- "encrypt" – шифрование контента,
- "decrypt" – расшифрование контента и проверка расшифрованного, если возможно,
- "wrapKey" – шифрование ключа,
- "unwrapKey" – расшифрование ключа и проверка расшифрованного, если возможно,
- "deriveKey" – вычисление производного ключа,
- "deriveBits" – вычисление производных битов не для использования в качестве ключа;

– <alg>: (опциональный) идентифицирует криптографический алгоритм, в котором предполагается использование ключа (подраздел 9.1 [5]);

– <kid>: (опциональный) идентификатор ключа;

– <x5u>: (опциональный) URI, который ссылается на ресурс сертификата формата X.509 или цепочки сертификатов ключа проверки цифровой подписи JWS; данный ресурс должен содержать представление сертификата или цепочки сертификатов в соответствии RFC 5280 [44] в PEM-кодировании; сертификат ключа проверки цифровой подписи JWS должен быть первым сертификатом; в цепочке сертификатов каждый последующий сертификат должен использоваться для сертификации предыдущего; получатель должен проверить цепочку сертификатов в соответствии с RFC 5280 [44], считать сертификат или цепочку сертификатов и подпись JWS недействительными в случае отрицательного результата проверки; для получения ресурса должен использоваться HTTP-запрос GET и протокол TLS;

– <x5c>: (опциональный) сертификат формата X.509 или цепочка сертификатов проверки цифровой подписи JWS; сертификат или цепочка сертификатов представлены в виде JSON-массива строк значений сертификата; каждая строка массива содержит кодировку Base64 (раздел 4 RFC 4648 [20], не подлежит применению Base64url-кодирование) значение DER-представления сертификата формата X.509; сертификат ключа проверки цифровой подписи JWS должен быть первым сертификатом; в цепочке сертификатов каждый последующий сертификат должен использоваться для сертификации предыдущего; получатель должен проверить цепочку сертификатов в соответствии с RFC 5280 [44], считать сертификат или цепочку сертификатов и подпись JWS недействительными в случае отрицательного результата проверки;

– <x5t#St256>: (опциональный) отпечаток сертификата открытого ключа; представляет собой Base64url-представление 256-битного значения хэш-функции ГОСТ Р 34.11-2012 [12], вычисленной для DER-кодирования X.509 сертификата открытого ключа.

В зависимости от типа ключа <kty> в состав структуры JWK включаются также другие специфические параметры.

Подробное описание JWK для российских криптографических алгоритмов приведено в подразделе 9.5 [5].

5.7.3.3. JSON структура набора ключей JWK Set состоит из одного параметра <keys>, который является JSON-массивом ключей в формате JWK.

5.7.3.4. Документ Key Set, на который указывает значение параметра <jwks_uri> метаданных сервера авторизации (подпункт 5.4.4.1), содержит ключ(и) проверки подписи сервера авторизации в формате JWK Set. Он также может содержать открытые ключи, с помощью которых вырабатывается общий ключ шифрования запросов к серверу авторизации. Набор Key Set, на который указывает значение параметра <jwks_uri> метаданных клиента (подпункт 5.4.4.3),

содержит ключ(и) проверки подписи клиента. Он также может содержать открытые ключи, с помощью которых вырабатывается общий ключ шифрования запросов к клиенту.

Если в наборе Key Set присутствуют как ключи проверки подписи, так и ключи шифрования, в структуре JWK каждого ключа должно быть указано значение параметра <use>. Не допускается использование одного и того же ключа как для целей подписи, так и для шифрования. Параметр JWK <x5c> может использоваться для предоставления ключей в формате сертификата X.509. В этом случае значение открытого ключа также должно присутствовать в составе JWK и должно совпадать со значением в сертификате.

Примечания:

1. Дополнительная информация о структурах JWK и JWK Set приведена в RFC 7517 [47].

2. Точная структура JWK и JWK Set определяется на этапе разработки сервера авторизации и клиента.

5.7.4. Структура JSON веб-токена

5.7.4.1. JSON веб-токен (JWT) [18] – структура данных (токен), основанная на формате JSON. JWT – набор параметров (claims) в формате JSON-объекта, закодированных в виде структуры JWS или JWE с цифровой подписью, кодом аутентификации и/или шифрованием.

Структура параметров JWT следующая:

– <iss>: (опциональный) идентификатор эмитента JWT (сервера авторизации или клиента, выпустившего его);

– <sub>: (опциональный) идентификатор субъекта JWT;

– <aud>: (опциональный) перечень идентификаторов получателей, которым предназначен JWT;

– <exp>: (обязательный) срок действия; время, при наступлении и после наступления которого ID токен не должен приниматься к обработке; значением является число в формате JSON, представляющее количество секунд от 1970-01-01T0:0:0Z UTC до соответствующей даты/времени;

– <nbf>: (опциональный) время, до которого JWT не должен приниматься к обработке;

– <iat>: (опциональный) время выпуска JWT;

– <jti>: (опциональный) идентификатор JWT.

При использовании JWT для производных структур, например ID токена (подпункт 5.4.2.14), возможно добавление других параметров, а некоторые опциональные параметры могут быть объявлены обязательными.

Параметры JWT помещаются в компоненте <JWS Payload> (пункт 5.7.1) или в зашифрованном виде – в содержимое <JWE Ciphertext> (пункт 5.7.2).

5.7.4.2. JWT может быть либо подписан, либо подписан и зашифрован. Если JWT подписан и зашифрован, JSON-документ должен быть сначала подписан, затем зашифрован, а результат – структура *вложенного JWT* – Nested JWT.

5.7.4.3. Описание JWT для российских криптографических алгоритмов приведено в подразделе 8.3 [5].

Примечание. Дополнительные сведения по структурам JWT и Nested JWT приведены в RFC 7519 [18].

5.8. Механизмы защиты

5.8.1. Цифровая подпись и шифрование

5.8.1.1. В зависимости от транспорта, с помощью которого отправляются сообщения, целостность сообщения может не гарантироваться, а отправитель сообщения может не проходить проверку подлинности. Чтобы уменьшить эти риски, при обработке значений ID токена, ответа UserInfo, объекта запроса и аутентификации клиента может использоваться JWS для цифровой подписи содержимого этих структур данных. Для обеспечения конфиденциальности сообщений также может использоваться JWE для шифрования содержимого этих структур данных.

Сервер авторизации может объявлять о поддерживаемых им алгоритмах подписи и шифрования в своих метаданных (подпункт 5.4.4.1), например, в документе Discovery, или предоставлять эту информацию другими способами. Клиент может декларировать свои алгоритмы цифровой подписи и шифрования в своих метаданных (подпункт 5.4.4.3), например, в запросе динамической регистрации, или передавать эту информацию другими способами.

Сервер авторизации может объявлять свои открытые ключи цифровой подписи и шифрования в своих метаданных, например, через документ Discovery, или предоставлять эту информацию другими способами. Клиент может объявлять свои открытые ключи в своих метаданных, например, через запрос динамической регистрации, или передавать эту информацию другими способами.

5.8.1.2. Цифровая подпись

Подписывающая сторона должна выбрать алгоритм цифровой подписи на основе алгоритмов, поддерживаемых получателем.

Ключ асимметричной цифровой подписи

Ключ цифровой подписи, который используется для формирования подписи контента, должен быть связан с открытым ключом, используемым для проверки цифровой подписи и опубликованным отправителем в его документе JWK Set. Если в указанном документе JWK Set несколько ключей, в JOSE заголовке должно присутствовать значение параметра `<kid>`. Параметр `<use>` соответствующего ключа должен указывать на то, что этот ключ поддерживает алгоритм цифровой подписи (“`sig`”). Алгоритмы вычисления/проверки цифровой подписи с использованием российских криптографических алгоритмов, приведенных в пункте 9.2.2 [5].

Ключ кода аутентификации сообщения

При использовании подписи JWS на основе кода аутентификации сообщения (MAC) значение параметра `<alg>` JOSE заголовка должно быть равно идентификатору алгоритма MAC. Используемый ключ MAC – байты UTF-8 представления значения `<client_secret>`. Требования к энтропии значений `<client_secret>` в пункте 10.1.3 [5]. Алгоритмы вычисления/проверки кода аутентификации с использованием российских криптографических алгоритмов приведен в пункте 9.2.1 [5]. Необходима регулярная смена `<client_secret>`. Срок действия ключа определяется исходя из анализа безопасности информационной системы и требований используемой СКЗИ. Безопасная и своевременная доставка `<client_secret>` клиенту не регламентируется настоящим стандартом.

5.8.1.3. Смена асимметричных ключей подписи

Смена асимметричных ключей цифровой подписи может быть выполнена с помощью следующего подхода. Подписывающая сторона публикует свои ключи в документе JWK Set, местоположение которого указывает в качестве значения параметра <jwks_uri> метаданных (подпункт 5.4.4.1), и включает идентификатор ключа цифровой подписи в качестве значения параметра <kid> JOSE заголовка каждого JWS-сообщения, чтобы указать проверяющей стороне, какой ключ должен использоваться для проверки цифровой подписи. Ключи необходимо обновлять, периодически добавляя новые ключи проверки цифровой подписи в JWK Set по адресу <jwks_uri>. Подписывающая сторона может начать использовать новый ключ цифровой подписи по своему усмотрению, сигнализируя об этом проверяющей стороне, использовав новое значение параметра <kid>. Проверяющая сторона понимает, что, обнаружив незнакомое значение параметра <kid>, он должен обратиться по адресу <jwks_uri> для повторного получения ключей отправителя. Документ JWK Set по адресу <jwks_uri> должен сохранять недавно выведенные из действия ключи проверки цифровой подписи в течение периода времени, соответствующего требованиям к СКЗИ. Перед проверкой цифровой подписи или вычислением ключа шифрования/расшифрования контента следует проверить актуальность используемого открытого ключа. С этой целью используется цепочка сертификатов (параметры <x5u>, <x5c> структуры JWK) и другие механизмы PKI.

Необходима регулярная смена ключей. Срок действия ключа определяется исходя из анализа безопасности информационной системы и требований используемой СКЗИ.

5.8.1.4. Шифрование

Отправитель, выполняющий шифрование передаваемой информации, должен выбрать алгоритм шифрования из тех алгоритмов, которые поддерживает получатель. Перечень этих алгоритмов указан в метаданных сервера авторизации (подпункт 5.4.4.1) и клиента (подпункт 5.4.4.3). Перечень поддерживаемых настоящим стандартом российских криптографических алгоритмов приведен в подразделе 9.1 [5]. Алгоритмы шифрования/расшифрования контента с использованием российских симметричных криптографических алгоритмов приведены в подразделе 9.4 [5].

Асимметричное шифрование на основе арифметики эллиптической кривой

Генерируется эфемерный открытый ключ с использованием арифметики эллиптической кривой в качестве элемента <epk> JOSE заголовка JWE. Второй открытый ключ, используемый для ключевого соглашения, должен быть открытым ключом, опубликованным получателем в его документе JWK Set. Если в указанном документе JWK Set есть несколько ключей, в JOSE заголовке JWE должно быть указано значение параметра <kid>. Для согласования ключа шифрования контента, который будет использован для зашифрования подписанного JWT, следует использовать алгоритм ключевого соглашения с использованием арифметики эллиптической кривой. Параметр <use> соответствующего ключа должен включать шифрование ("enc"). Алгоритмы вычисления ключа шифрования контента с использованием российских асимметричных криптографических алгоритмов приведены в пункте 9.3.2 [5].

Симметричное шифрование

Симметричный ключ шифрования контента вычисляется из значения <client_secret>. Алгоритмы вычисления ключа шифрования контента с использованием российских симметричных криптографических алгоритмов приведены в пункте 9.3.1 [5].

Необходима регулярная смена <client_secret>. Срок действия этого ключа определяется исходя из анализа безопасности информационной системы и требований используемого СКЗИ. Методы реализации безопасной и своевременной доставки <client_secret> клиенту не регламентируются настоящим стандартом.

5.8.1.5. Смена асимметричных ключей шифрования

При смене ключей шифрования следует использовать процесс, отличный от процесса смены ключей цифровой подписи, поскольку процесс смены открытого ключа получателя запускает получатель, не являющийся отправителем (шифрующим), и, таким образом, отправитель не может полагаться на изменение параметра <kid> в качестве сигнала о необходимости смены ключей. Шифрующий продолжает использовать прежний параметр <kid> в заголовке JWE. При этом он должен выбрать наиболее подходящий ключ из предоставленных в JWK Set, расположенном по адресу <jwks_uri> получателя. При отсутствии в JWK Set открытого ключа получателя с разрешенным сроком использования шифрование не допускается. Об этом следует сообщить получателю, пользуясь другим надежным каналом связи.

Для смены ключей расшифровывающая сторона своевременно должна опубликовать новые ключи по своему адресу <jwks_uri> и удалить из JWK Set те ключи, которые выводятся из эксплуатации. Должны быть предприняты меры по корректному кешированию <jwks_uri> в соответствии с рекомендациями пункта 10.2.1 [17]. Получатель должен удалить отозванные ключи из JWK Set, но сохранять их у себя в течение некоторого времени, согласованного с продолжительностью кеша, чтобы обеспечить плавный переход между ключами, предоставляя отправителю некоторое время на загрузку новых ключей. Продолжительность кеша следует также координировать с выдачей новых ключей подписи, как описано в подпункте 5.8.1.3.

Необходима регулярная смена этих ключей. Срок действия ключей определяется исходя из анализа безопасности информационной системы и требований используемой СКЗИ.

5.8.2. Требования к энтропии ключевой информации и других параметров

Значения секрета клиента <client_secret>, токенов доступа, токенов обновления, кодов авторизации, параметров <nonce>, <state>, <code_verifier>, <seed>, <JWE Initialization Vector> и другая ключевая информация должны генерироваться с помощью СКЗИ, используемого при реализации криптографической защиты информации, в соответствии с требованиями пункта 10.1.3 [5].

5.8.3. Требования к TLS

Приложения, разрабатываемые в соответствии с настоящим стандартом, должны выполнять требования к использованию протокола TLS, сформулированные в пункте 10.1.4 [5][1].

5.8.4. Токен доступа, связанный с MTLS-сертификатом клиента

5.8.4.1. Если клиент использует взаимную аутентификацию по протоколу TLS (MTLS) при подключении к конечной точке токена, сервер авторизации может связать выданный токен доступа с предъявленным сертификатом клиента [49]. Такое связывание достигается путем встраивания хэш-кода сертификата в выданный токен доступа с использованием JWT-синтаксиса (подпункт 5.8.4.2) или посредством интроспекции токена (запроса информации

[1] В случае реализации трансграничного взаимодействия выбор конкретных криптонаборов, включенных в реестр идентификаторов IANA, определяется соглашением (договором) сторон обмена.

о токене) у сервера авторизации (подпункт 5.8.4.3). Эта привязка может выполняться как совместно с аутентификацией клиента по сертификату MTLS (подраздел 5.5), так и отдельно от аутентификации клиента сервером авторизации, что позволяет MTLS во время защищенного доступа к ресурсам служить исключительно в качестве механизма подтверждения владения закрытым ключом и тем самым ограничить использование токена доступа исключительно только клиентом, обладающим этим закрытым ключом (токен доступа с ограничением отправителя, sender-constrained access token) (пункт 4.10.1 [52]).

Чтобы сервер ресурсов мог использовать токены доступа с привязкой к сертификату, он должен заранее знать, что для обращения к защищенным ресурсам должен использоваться MTLS. В частности, сам токен доступа не может использоваться в качестве входных данных для принятия решения о том, запрашивать или нет установление MTLS-соединения.

В процессе доступа к ресурсам, защищенным протоколом TLS с взаимной аутентификацией сторон, клиент может выполнять запросы к защищенным ресурсам (подраздел 5.6), однако эти запросы должны быть выполнены по аутентифицированному MTLS-соединению с использованием того же сертификата, что и для MTLS-соединения в конечной точке токена при запросе токена доступа.

Сервер ресурсов должен получить TLS-сертификат клиента, используемый для установления взаимно аутентифицированного TLS-соединения, должен проверить, что сертификат соответствует сертификату, связанному с предъявленным токеном доступа. Если они не совпадают, попытка доступа к ресурсу должна быть отклонена со статусом HTTP-ответа с кодом состояния 401 и заголовком `"invalid_token"`.

Метаданные, необходимые для того, чтобы сервер и клиент сигнализировали о желании использовать токены доступа с привязкой к MTLS-сертификату клиента, определены в подпункте 5.8.4.4.

5.8.4.2. Метод подтверждения отпечатка сертификата с использованием JWT

Чтобы представить хэш-код X.509 сертификата в формате JWT (подпункт 6.7.3.2 [5]), в качестве значения параметра токена `<cnf>` используется строка, идентифицирующая метод подтверждения на основе хэш-функции с длиной образа не менее 256 битов, и значение хэш-кода, которое формируется как Base64url-кодирование значения хэш-функции, вычисленное от DER-представления (ГОСТ Р ИСО/МЭК 8825-1) сертификата формата X.509.

Примечание. Дополнительные сведения о методе подтверждения отпечатка сертификата с использованием JWT приведены в подразделе 3.1 [49].

Ниже приведен пример функционального содержимого JWT, содержащего подтверждение отпечатка сертификата.

```
{
    "iss": "https://server.example.com",
    "sub": "ty.webb@example.com",
    "exp": 1493726400,
    "nbf": 1493722800,
    "jti": "9cb4a49ff647111b28320c914ca845a6ebe42e981bacc5fd0ecf92ae1ae60d75",
    "cnf":{
        "x5t#St256": "bwcK0esc3ACC3DB2Y5_lESsXE8o9ltc05O89jdN-dg2"
        }
}
```

5.8.4.3. Метод подтверждения отпечатка сертификата с использованием интроспекции токена

Интроспекция токена OAuth 2.0 определяет способ, с помощью которого сервер ресурсов может запрашивать у сервера авторизации информацию о состоянии активности токена доступа, а также другую метаинформацию о токене доступа.

Для токена доступа, связанного с сертификатом MTLS, хэш-код сертификата, с которым связан токен, передается серверу защищенного ресурса в виде метаинформации в составе ответа интроспекции токена. Хэш-код передается с использованием того же параметра `<cnf>` с элементом, идентифицирующим метод подтверждения на основе хэш-функции, что и при использовании метода подтверждения отпечатка сертификата, описанного в подпункте 5.8.4.2, в качестве параметра верхнего уровня JSON ответа интроспекции. Сервер ресурсов сравнивает полученный таким образом хэш-код сертификата со значением хэш-кода, вычисленного на основе сертификата клиента, использованного для взаимной аутентификации сеанса TLS, и отклоняет запрос, если они не совпадают.

Примечания:

1. Дополнительные сведения о протоколе интроспекции токена доступа представлены в RFC 7662 [48].

2. Дополнительные сведения о методе подтверждения отпечатка сертификата с использованием интроспекции токена приведены в подразделе 3.2 [49].

Ниже приведен пример ответа интроспекции для активного токена с подтверждением отпечатка сертификата.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "active": true,
    "iss": "https://server.example.com",
    "sub": "ty.webb@example.com",
    "exp": 1493726400,
    "nbf": 1493722800,
    "jti": "9cb4a49ff647111b28320c914ca845a6ebe42e981bacc5fd0ecf92ae1ae60d75",
    "cnf":{
        "x5t#St256": "bwcK0esc3ACC3DB2Y5_lESsXE8o9ltc05O89jdN-dg2"
    }
}
```

5.8.4.4. Метаданные сервера авторизации и клиента для подтверждения отпечатка MTLS-сертификата

Следующий параметр метаданных сервера авторизации, публикуемый в соответствии с требованиями подпункта 5.4.4.1, указывает на способность сервера авторизации выдавать токены доступа с привязкой к сертификату:

– `<tls_client_certificate_bound_access_tokens>`: (опциональный) логическое значение, указывающее на поддержку сервером токенов доступа с привязкой к MTLS-сертификату клиента. Значением по умолчанию является `"false"`.

Следующий параметр метаданных клиента, публикуемый в соответствии с требованиями подпункта 5.4.4.3, позволяет сигнализировать намерение клиента использовать токены доступа с привязкой к сертификату:

– `<tls_client_certificate_bound_access_tokens>`: (опциональный) логическое значение, используемое для указания намерения клиента использовать токены доступа, привязанные к MTLS-сертификату клиента. Значением по умолчанию является "`false`".

Если клиент, который указал намерение использовать токены, привязанные к MTLS-сертификату клиента, отправляет запрос на конечную точку токена по соединению, не являющемуся MTLS, на усмотрение сервера авторизации остается принятие решения относительно того, следует ли возвращать ошибку или выдать токен, не связанный с сертификатом.

6. БАЗОВЫЙ ПРОФИЛЬ БЕЗОПАСНОСТИ API ПЕРЕДАЧИ ФИНАНСОВОЙ ИНФОРМАЦИИ

6.1. Вводная информация

Раздел 6 определяет базовый профиль безопасности OAuth 2.0, который подходит для защиты API со средним уровнем доверия к идентификации и аутентификации, при использовании которых не передаются банковская тайна или персональные данные. Этот профиль не обеспечивает неотказуемость (подписание запросов аутентификации и ответов) и токены доступа, связанные с MTLS-сертификатом клиента. Если API требуют высокий уровень доверия к идентификации и аутентификации в том числе используются для передачи банковской тайны и персональных данных, необходимо использовать «Расширенный профиль безопасности API передачи финансовой информации», приведенный в разделе 7.

6.2. Положения об обеспечении безопасности сервера авторизации

6.2.1. Сервер(у) авторизации:

1. Должен поддерживать конфиденциальных клиентов.

2. Не должен поддерживать публичных клиентов.

3. В случае использования симметричного ключа должен предоставлять секрет клиента `<client_secret>`, соответствующий требованиям пункта 5.8.2.

4. Должен аутентифицировать конфиденциального клиента в конечной точке токена, используя один из следующих методов (подраздел 5.5):

   - `client_secret_jwt`: аутентификация на основе кода аутентификации HMAC и `<client_secret>`;
   - `private_key_jwt`: аутентификация на основе цифровой подписи;
   - `tls_client_auth`: аутентификация с использованием MTLS и PKI для связывания сертификата с клиентом.

5. Должен требовать и использовать ключ, размер которого составляет 256 бит или больше для симметричных алгоритмов и алгоритмов на основе эллиптической кривой ([5]).

6. Должен требовать использования технологии PKCE (подпункт 5.4.2.4) со значением `"St256"` (пункт 6.2.6 [5]) в качестве метода запроса кода.

7. Должен требовать предварительной регистрации URI-переадресации клиентов.

8. Должен требовать наличия параметра `<redirect_URI>` в запросе аутентификации (подпункт 5.4.2.2).

9. Должен требовать, чтобы значение параметра `<redirect_URI>` точно соответствовало одному из предварительно зарегистрированных URI-переадресации клиента (подпункт 5.4.4.3).

10. Должен требовать аутентификацию конечного пользователя на соответствующем уровне гарантии ([6]) для операций, которые клиент будет уполномочен выполнять от имени пользователя; допустимые контексты аутентификации пользователя сервер авторизации получает в качестве значения `<acr_values>` запроса аутентификации.

11. Должен требовать явного согласия конечного пользователя на авторизацию доступа к запрашиваемой области действия (параметр `<scope>`), если она не была ранее авторизована.

12. Должен исключать повторное использование кодов авторизации в соответствии с подпунктом 5.4.2.11.

13. Должен возвращать ответ на запрос токена доступа в соответствии с подпунктом 5.4.2.9.

14. Должен возвращать список предоставленных областей действия (параметр `<scope>`) по выданному токену доступа (подпункт 5.4.2.15).

15. Должен предоставлять непредсказуемые токены доступа, коды авторизации и токены обновления (если необходимо) с достаточной энтропией, чтобы вероятность того, что злоумышленник угадает сгенерированный токен, была вычислительно неосуществима, согласно требованиям пункта 10.1.3 [5].

16. Рекомендуется точно определять детали предоставления конечному пользователю прав во время авторизации в соответствии с подразделом 16.18 [17].

17. Рекомендуется предоставлять конечному пользователю механизм аннулирования токенов доступа и токенов обновления, выданных клиенту.

18. При обращении к методам аутентификации клиента, которые могут передавать идентификатор клиента более чем одним разрешенным способом, в случае предоставления несовпадающих идентификаторов клиента должен возвращать код ошибки `"invalid_client"`.

19. Должен требовать, чтобы сервисы по адресу URI-переадресации использовали протокол HTTPS.

20. Рекомендуется выдавать токены доступа со сроком действия менее 10 минут, если только токены не ограничены отправителем.

21. Должен поддерживать OpenID Discovery [40], может поддерживать метаданные сервера авторизации в соответствии с RFC8414 [41] и не должен распространять метаданные обнаружения (такие как конечная точка авторизации) любым другим способом.

Примечания:

1. Рекомендуется использовать токены обновления вместо токенов доступа с длительным сроком действия.

2. Сервер может ограничивать область действия `<scope>`, чтобы не реализовывать определенные API.

3. Ожидается, что клиенты будут рассматривать токены доступа как неструктурированные, непрозрачные символьные строки.

4. Требование возвращать список разрешенных областей действия позволяет клиентам обнаружить, когда запрос аутентификации был изменен с целью навязывания других областей действия. Серверы всегда должны возвращать предоставленные области, если они отличаются от запрошенных.

6.2.2. При обработке параметра <scope> в запросе аутентификации сервер авторизации:

1. Должен требовать обязательного наличия непустого значения параметра <scope> в запросе аутентификации.

2. Должен возвращать сообщение об ошибке при получении запроса аутентификации без данного параметра.

3. Не должен предполагать значений <scope> по умолчанию.

4. Должен игнорировать неизвестные или неразрешенные значения <scope>.

5. Должен выдать токен доступа, перечислив в явном виде только разрешенные клиенту области действия, если доступ клиента хотя бы к одному значению из перечня <scope> разрешен.

6. Должен вернуть сообщение об ошибке, если в перечне <scope> нет известных или разрешенных клиенту значений.

6.2.3. Если *в ответе на запрос токена доступа* возвращается идентификатор аутентифицированного пользователя, сервер авторизации:

1. Должен поддерживать запрос аутентификации в соответствии с подпунктом 5.4.2.2.

2. Должен осуществлять проверку запроса аутентификации (пункт 6.2.7 [5]).

3. Должен аутентифицировать пользователя в соответствии с подпунктом 5.4.2.5.

4. Должен предоставлять ответ на запрос аутентификации в соответствии с подпунктом 5.4.2.7 и подпунктом 5.4.2.8 в зависимости от результата аутентификации.

5. Должен осуществлять проверку запроса токена в соответствии с подпунктом 5.4.2.11.

6. Если параметр <scope> запроса аутентификации включает "openid", должен генерировать ID токен в составе ответа на запрос токена в соответствии с подпунктом 5.4.2.12 со значением параметра <sub>, соответствующим аутентифицированному пользователю, и значением <acr> в составе ID токена, указывающим на класс аутентификации в соответствии с [6].

6.2.4. Если клиент запрашивает область действия "openid", сервер авторизации должен требовать наличия в запросе аутентификации значения параметра <nonce> (подпункт 5.4.2.2).

6.3. Положения об обеспечении безопасности клиента

6.3.1. Клиент:

1. Должен поддерживать технологию PKCE (подпункт 5.4.2.4).

2. Должен использовать "S256" (пункт 6.2.6 [5]) в качестве метода запроса кода технологии PKCE (подпункт 5.4.2.4).

3. Должен использовать разные URI-переадресации для каждого сервера авторизации, на котором зарегистрирован клиент.

4. Должен хранить значение URI-переадресации в сеансе агента пользователя владельца ресурса и сравнивать его со значением URI-переадресации, по которому был получен ответ авторизации; если URI не совпадают, клиент должен завершить процесс с ошибкой.

5. Должен реализовать эффективную защиту от CSRF.

6. Должен поддерживать следующие методы аутентификации на конечной точке токена: (подраздел 5.5):

   - `client_secret_jwt`: аутентификация на основе кода аутентификации HMAC и `<client_secret>`;
   - `private_key_jwt`: аутентификация на основе цифровой подписи;
   - `tls_client_auth`: аутентификация с использованием MTLS и PKI для связывания сертификата с клиентом.

7. Должен использовать ключи, размер которых составляет 256 бит или больше, для алгоритмов на основе эллиптической кривой.

8. Должен проверять, что значение секрета клиента `<client_secret>` имеет длину не менее 256 бит, если используется криптография с симметричным ключом.

9. Если по результатам аутентификации клиенту необходимо получить постоянный идентификатор аутентифицированного пользователя, клиент:

   - должен включать строку `"openid"` в перечень значений параметра `<scope>`;
   - должен включать параметр `<nonce>` (подпункт 5.4.2.2) в состав запроса аутентификации.

10. Если `"openid"` не входит в значение предметной области `<scope>`, публичный клиент:

    - должен включить параметр <state> в запрос аутентификации (подпункт 5.4.2.2);
    - должен генерировать состояние (контекст) агента пользователя при формировании запроса аутентификации и далее проверять соответствие этого состояния значению параметра `<state>` (пункты 6.2.1, 6.2.3, 6.2.5 [5]);
    - должен проверить, что область действия, полученная в ответе токена, либо в точности совпадает, либо является подмножеством области действия, отправленной в запросе аутентификации;
    - должен использовать только достоверные метаданные сервера авторизации, полученные из документа метаданных, опубликованного сервером авторизации по адресу его достоверной конечной точки (подпункт 5.4.4.1).

6.4. Положения об обеспечении безопасности доступа к защищенным ресурсам

6.4.1. Конечные точки ресурсов возвращают клиенту защищенную информацию владельца ресурса, связанного с предъявленным токеном доступа.

6.4.2. *Сервер ресурсов*, предоставляющий доступ к данным в соответствии с настоящим стандартом:

1. Должен принимать токены доступа в HTTP-заголовке в соответствии с подразделом 2.1 RFC6750 [19].

2. Не должен принимать токены доступа в параметрах запроса, указанных в подразделе 2.3 RFC6750 [19].

3. Должен проверять, что срок действия токена доступа не истек и токен доступа не отозван.

4. Должен проверять, что область действия (параметр <scope>), связанная с предъявленным токеном доступа, разрешает доступ к ресурсу, который он представляет.

5. Должен идентифицировать объект, связанный с токеном доступа.

6. Должен возвращать только ресурс, идентифицированный комбинацией объекта, подразумеваемого в доступе, и разрешенной области действия, соответствующей объекту, иначе возвращать ошибку в соответствии с разделом 3.1 RFC6750 [19].

7. Должен кодировать ответ в UTF-8.

8. Должен отправлять параметр <Content-Type> HTTP-заголовка в виде "Content-Type: application/JSON".

9. Должен отправлять дату сервера в заголовке HTTP <Date> в соответствии с подпунктом 7.1.1.2 RFC7231 [50].

10. Должен устанавливать в качестве значения заголовка ответа <x-FAPI-interaction-id> значение, полученное из соответствующего заголовка запроса клиента, либо значение UUID, если заголовок запроса не предоставлялся для отслеживания взаимодействия, например, "x-FAPI-interaction-id: c770aef3-6784-41f7-8e0e-ff5f97bddb3a".

11. Должен регистрировать значение <x-FAPI-interaction-id> в записи журнала регистрации событий.

12. Не должен отклонять запросы с заголовком <x-fapi-customer-ip-address>, содержащим действительный IPv4- или IPv6-адрес.

Примечания:

1. Форматы и алгоритмы формирования UUID определены в RFC 4122 [51].

2. Для получения информации об объекте, связанном с токеном доступа и предоставленной областью действия, сервер ресурсов может использовать протокол интроспекции токена, описанный в RFC 7662 [48]. В случае его реализации следует выполнить анализ его безопасности.

6.4.3. *Клиент*, реализующий доступ к защищенному ресурсу в соответствии с требованиями данного стандарта:

1. Должен отправлять токены доступа в заголовке HTTP в соответствии с положениями пункта 4.3.1 RFC7231 [50].

2. Может отправлять последнюю дату регистрации пользователя в клиенте в заголовке <x-FAPI-auth-date>, причем значение предоставляется в формате HTTP-даты; например, "x-FAPI-auth-date: Tue, 11 Sep 2012 19:43:31 GMT".

3. Может отправлять в заголовке <x-FAPI-customer-ip-address> IP-адрес пользователя, если эти данные ему доступны; например, "x-FAPI-customer-ip-address: 198.51.100.119".

4. Для синхронизации записей журналов регистрации клиента и сервера ресурсов может отправлять серверу заголовок <x-FAPI-interaction-id> запроса, значением которого является UUID; например, "x-FAPI-interaction-id: c770aef3-6784-41f7-8e0e-ff5f97bddb3a".

7. РАСШИРЕННЫЙ ПРОФИЛЬ БЕЗОПАСНОСТИ API ПЕРЕДАЧИ ФИНАНСОВОЙ ИНФОРМАЦИИ

7.1. Вводная информация

7.1.1. Ресурсы API могут обрабатывать конфиденциальные данные и/или иметь повышенные требования к безопасности. Чтобы удовлетворить различные потребности в обеспечении безопасности, в данном разделе формулируется расширенный профиль, который выходит за рамки базовых требований безопасности, определенных в разделе 6.

7.2. Положения об обеспечении безопасности сервера авторизации

7.2.1. *Сервер авторизации* должен поддерживать положения, указанные в подразделе 6.2. При этом положение пункта 6.2.1 в части перечисления 6 (PKCE) не должно использоваться.

Кроме того, сервер авторизации:

1. Должен требовать подписанный JWS (пункт 5.7.1) объект запроса JWT (пункт 5.7.4), переданный по значению с параметром `<request>` или по ссылке с параметром `<request_uri>`.

2. Должен требовать:

   a. значения `<response_type>` = `"code id_token"` (Режим 3 [5]), или

   b. значения `<response_type>` = `"code"` в сочетании со значением `<response_mode>` = “`jwt`” (Режим 2 [5]).

3. Должен выдавать только токены доступа с ограничением отправителя (sender-constrained access tokens) (пункт 5.8.4).

4. Должен поддерживать MTLS как механизм для ограничения легитимных отправителей токенов доступа (пункт 5.8.4).

5. Должен использовать только параметры, включенные в подписанный объект запроса, переданный через параметр `<request>` или `<request_uri>`.

6. Должен требовать, чтобы объект запроса содержал параметр `<exp>` с временем жизни не более 60 минут, начиная со значения параметра `<nbf>`.

7. В отличие от требований подраздела 6.2, должен аутентифицировать конфиденциального клиента, используя один из методов, указанных в подразделе 5.5:

   – `"private_key_jwt"`: аутентификация на основе цифровой подписи;

   – “`tls_client_auth`”: аутентификация с использованием MTLS и PKI для связывания сертификата с клиентом.

8. Должен требовать, чтобы параметр `<aud>` в объекте запроса содержал URL идентификатора эмитента.

9. Не должен поддерживать публичных клиентов.

10. Должен требовать, чтобы объект запроса содержал параметр <nbf>, который не должен быть старше 60 минут.

7.2.2. ID токен в качестве отделенной подписи

Кроме того, если используется значение <response_type> = "code id_token", сервер авторизации:

1. Должен поддерживать OpenID Connect в объеме методических рекомендаций [5].

2. Должен поддерживать подписанные ID токены.

3. Должен поддерживать подписанные и зашифрованные ID токены.

4. Должен возвращать ID токен в качестве отделенной подписи (detached signature) к ответу авторизации.

Примечание. ID токен не только подтверждает идентификацию владельца ресурса (субъекта), он также идентифицирует сервер авторизации (содержит идентификатор эмитента). В случае если ID токен включает идентификатор субъекта, он действует как отделенная подпись (detached signature) источника данных к ответу авторизации. Таким образом, ID токен также защищает ответ авторизации путем включения в ID токен хэш-кода незащищенных параметров ответа <code> и <state>.

5. Если клиент предоставил значение <state>, для его защиты должен включать в ID токен хэш-код состояния <s_hash>. <s_hash> может быть опущен в ID токене, возвращаемом с конечной точки токена, если <s_hash> присутствует в ID токене, возвращаемом с конечной точки авторизации.

6. Не должен возвращать конфиденциальные персональные данные в ID токене в ответе авторизации.

Примечание. Сервер авторизации может вернуть из конечной точки авторизации больше параметров в составе ID токена, чем в ответе авторизации, согласно синтаксису OAuth 2.0.

7.2.3. JARM

Кроме того, если значение <response_type> = "code" используется в сочетании со значением <response_mode> = "jwt", сервер авторизации должен создавать защищенные JWT ответы авторизации (пункт 5.4.5).

7.3. Положения об обеспечении безопасности клиента

7.3.1. *Клиент* должен поддерживать положения подраздела 6.3, за исключением поддержки PKCE.

Кроме того, конфиденциальный клиент:

1. Должен поддерживать MTLS токены доступа с привязкой к сертификату (пункт 5.8.4).

2. Должен включать в состав запроса аутентификации один из параметров <request> или <request_uri>.

3. Должен убедиться, что сервер авторизации аутентифицировал пользователя на требуемом уровне доверия в соответствии с [6] для предполагаемой цели клиента (допустимые значения: параметр <acr_values> запроса аутентификации; фактическое значение: параметр <acr> ID токена).

4. Должен отправлять все параметры запроса в составе подписанного объекта запроса аутентификации (подпункт 5.4.2.3).

5. Должен дополнительно отправлять дубликаты параметров/значений <response_type>, <client_id> и <scope>, используя синтаксис запроса OAuth 2.0 (подраздел 6.1 [17]).

6. Должен отправлять параметр <aud> в объекте запроса в качестве URL-идентификатора эмитента.

7. Должен отправлять в объекте запроса параметр <exp> с временем жизни не более 60 минут.

8. Должен посылать в составе объекта запроса параметр <nbf>.

7.3.2. ID токен в качестве отделенной подписи

Кроме того, если используется <response_type> = "code id_token", клиент:

1. Должен включить строку "openid" в значение параметра <scope>, чтобы активировать поддержку OIDC.

2. Должен требовать, чтобы от конечных точек возвращался ID токен, подписанный JWS.

3. Должен проверить, что ответ авторизации не был подделан (ID токен в качестве отделенной подписи).

4. Должен проверять, что значение <s_hash> равно значению, вычисленному на основе значения <state> в ответе авторизации.

5. Должен поддерживать подписанные, а также подписанные и зашифрованные ID токены.

7.3.3. JARM

Кроме того, если значение <response_type> = "code" используется в сочетании со значением <response_mode> = "jwt", клиент должен проверять ответы авторизации в соответствии с пунктом 6.4.4 [5] и пунктом 5.4.5.

7.4. Положения об обеспечении безопасности доступа к защищенным ресурсам (с использованием токенов)

7.4.1. Конечные точки – это конечные точки ресурсов, защищенных с помощью протокола OpenID Connect, которые возвращают защищенную информацию владельца ресурса, связанного с представленным токеном доступа.

7.4.2. *Сервер ресурсов, обрабатывающий запросы на предоставление доступа к защищенным ресурсам*, определенным в терминах настоящего документа:

1. Должен поддерживать положения, определенные в пункте 6.4.2.

2. Должен поддерживать механизм подтверждения владения ключом с привязкой токена к MTLS-сертификату (пункт 5.8.4) или с привязкой токена к TLS-соединению.

7.4.3. *Клиенты*, определенные в терминах настоящего стандарта, должны поддерживать положения, определенные в пункте 6.4.3.

БИБЛИОГРАФИЯ

[1] Стандарт Банка России «Открытые банковские интерфейсы. Общие положения». 23.10.2020. https://www.cbr.ru/StaticHtml/File/59 420/standart_1. pdf (дата обращения: 22.01.2024).

[2] Стандарт Банка России «Открытые банковские интерфейсы. Получение публичной информации о кредитной организации и ее продуктах». 08.07.2021. http://www.cbr.ru/statichtml/file/59420/standart_08072021. pdf (дата обращения: 22.01.2024).

[3] Стандарт Банка России «Открытые банковские интерфейсы. Инициирование перевода денежных средств клиента третьей стороной в валюте Российской Федерации». 23.10.2020. http://www.cbr.ru/statichtml/file/59420/standart_3. pdf (дата обращения: 22.01.2024).

[4] Стандарт Банка России «Открытые банковские интерфейсы. Получение информации о счете клиента третьей стороной». 23.10.2020. http://www.cbr.ru/statichtml/file/59420/standart_2. pdf (дата обращения: 22.01.2024).

[5] Методические рекомендации МР 26.2.002-2024 ТК 26 «Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколах OpenID Connect». 2024.

[6] Стандарт Банка России СТО БР БФБО-1.8-2024 «Безопасность финансовых (банковских) операций. Обеспечение безопасности финансовых сервисов при проведении дистанционной идентификации и аутентификации. Состав мер защиты информации». 28.02.2024.

[7] ГОСТ Р ИСО/МЭК 27000–2021 Национальный стандарт Российской Федерации «Информационные технологии. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология».

[8] ГОСТ Р 58833-2020 «Защита информации. Идентификация и аутентификация. Общие положения».

[9] Р 50.1.053-2005 «Информационные технологии. Основные термины и определения в области технической защиты информации».

[10] ГОСТ Р 57580.1-2017 «Безопасность финансовых (банковских) операций. Защита информации финансовых организаций. Базовый состав организационных и технических мер».

[11] ГОСТ Р 34.10-2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи».

[12] ГОСТ Р 34.11-2012 «Информационная технология. Криптографическая защита информации. Функция хэширования».

[13] ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры».

[14] Словарь криптографических терминов. Под ред.Б.А. Погорелова и В.Н. Сачкова. – М.: МЦНМО, 2006. – 94 с.

[15] Р 50.1.041-2002 «Информационные технологии. Руководство по проектированию профилей среды открытой системы (СОС) организации-пользователя».

[16] Hardt D. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012. https://datatracker.ietf.org/doc/html/rfc6749 (дата обращения: 07.07.2023).

[17] Sakimura N., Bradley J., Jones M., de Medeiros B., Mortimore C. OpenID Connect Core 1.0 incorporating errata set 1. https://openid.net/specs/openid-connect-core-1_0. html November 8, 2014 (дата обращения: 07.07.2023).

[18] Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. May 2015. https://datatracker.ietf.org/doc/html/rfc7519 (дата обращения: 07.07.2023).

[19] Jones M., Hardt D. The OAuth 2.0 Authorization Framework: Bearer Token Usage. RFC 6750, October 2012. https://datatracker.ietf.org/doc/html/rfc6750 (дата обращения: 07.07.2023).

[20] Josefsson S. The Base16, Base32, and Base64 Data Encodings. RFC 4648. October 2006. https://datatracker.ietf.org/doc/html/rfc4648 (дата обращения: 07.07.2023).

[21] N. Sakimura, J. Bradley, E. Jay. Financial-grade API Security Profile 1.0 – Part 1: Baseline. OpenID Foundation, 2021. https://openid.net/specs/openid-financial-api-part-1-1_0. html (дата обращения: 07.07.2023).

[22] N. Sakimura, J. Bradley, E. Jay. Financial-grade API Security Profile 1.0 – Part 2: Advanced. OpenID Foundation, 2021. https://openid.net/specs/openid-financial-api-part-2-1_0. html (дата обращения: 07.07.2023).

[23] Р 1323 565.1.026-2019 «Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров, реализующие аутентифицированное шифрование».

[24] Lodderstedt T., Campbell B., Financial-grade API: JWT Secured Authorization Response Mode for OAuth 2.0 (JARM). OpenID Foundation, FAPI WG, 9 November 2022. https://openid.net/specs/oauth-v2-jarm. html (дата обращения: 07.07.2023).

[25] Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных».

[26] Постановление Правительства Российской Федерации от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных».

[27] Приказ ФСБ России от 10.07.2014 № 378 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных с использованием средств криптографической защиты информации, необходимых для выполнения установленных Правительством Российской Федерации требований к защите персональных данных для каждого из уровней защищенности».

[28] Приказ ФСТЭК России от 18.02.2013 № 21 «Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных».

[29] Положение Банка России от 17.08.2023 № 821-П «О требованиях к обеспечению защиты информации при осуществлении переводов денежных средств и о порядке осуществления Банком России контроля за соблюдением требований к обеспечению защиты информации при осуществлении переводов денежных средств».

[30] Положение Банка России от 17.04.2019 № 683-П «Об установлении обязательных для кредитных организаций требований к обеспечению защиты информации при осуществлении банковской деятельности в целях противодействия осуществлению переводов денежных средств без согласия клиента».

[31] Положение Банка России от 20.04.2021 № 757-П «Об установлении обязательных для некредитных финансовых организаций требований к обеспечению защиты информации при осуществлении деятельности в сфере финансовых рынков в целях противодействия осуществлению незаконных финансовых операций».

[32] Положение Банка России от 25.07.2022 № 802-П «О требованиях к защите информации в платежной системе Банка России».

[33] ГОСТ Р ИСО/МЭК 15 408-3-2013 «Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Компоненты доверия к безопасности».

[34] Методический документ Банка России «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций» http://www.cbr.ru/content/document/file/132 666/inf_note_feb_0422. pdf. (дата обращения 22.01.2024).

[35] Методический документ ФСТЭК «Руководство по организации процесса управления уязвимостями в органе (организации)» (утв. ФСТЭК России 17.05.2023). https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-17-maya-2023-g (дата обращения: 22.01.2024).

[36] Методический документ ФСТЭК «Методика тестирования обновлений безопасности программных, программно-аппаратных средств» (утв. ФСТЭК России 28.10.2022). https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-28-oktyabrya-2022-g (дата обращения: 22.01.2024).

[37] Методический документ ФСТЭК «Методика оценки уровня критичности уязвимостей программных, программно-аппаратных средств» (утв. ФСТЭК России 28.10.2022). https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-28-oktyabrya-2022-g-2 (дата обращения: 22.01.2024).

[38] Sakimura N., Bradley J., Agarwal N. Proof Key for Code Exchange by OAuth Public Clients. RFC 7636. September 2015. https://datatracker.ietf.org/doc/html/rfc7636 (дата обращения: 07.07.2023).

[39] Jones P., Salgueiro G., Jones M., J. Smarr. WebFinger. RFC 7033. September 2013. https://datatracker.ietf.org/doc/html/rfc7033 (дата обращения: 22.01.2024).

[40] Sakimura N., Bradley J., Jones M., Jay E. OpenID Connect Discovery 1.0 incorporating errata set 1. November 8, 2014. https://openid.net/specs/openid-connect-discovery-1_0. html (дата обращения: 22.01.2024).

[41] Jones M., Sakimura N., Bradley J. OAuth 2.0 Authorization Server Metadata. RFC 8414. June 2018. https://datatracker.ietf.org/doc/html/rfc8414 (дата обращения: 22.01.2024).

[42] Sakimura N., Bradley J., Jones M. OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 1. November 8, 2014. https://openid.net/specs/openid-connect-registration-1_0. html (дата обращения: 22.01.2024).

[43] Richer J., Jones M. Bradley J., Machulak M., Hunt P. OAuth 2.0 Dynamic Client Registration Protocol. RFC 7591. July 2015. https://datatracker.ietf.org/doc/html/rfc7591 (дата обращения: 22.01.2024).

[44] Cooper D., Santesson S., Farrell S., Boeyen S., Housley R., Polk W. Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. May 2008. https://datatracker.ietf.org/doc/html/rfc5280 (дата обращения: 22.01.2024).

[45] Jones M., Bradley J., Sakimura N. JSON Web Signature (JWS). RFC 7515. May 2015. https://datatracker.ietf.org/doc/html/rfc7515 (дата обращения: 22.01.2024).

[46] Jones M., Hildebrand J. JSON Web Encryption (JWE). RFC 7516. May 2015. https://datatracker.ietf.org/doc/html/rfc7516 (дата обращения: 22.01.2024).

[47] Jones M. JSON Web Key (JWK). RFC 7517. May 2015. https://datatracker.ietf.org/doc/html/rfc7517 (дата обращения: 22.01.2024).

[48] Richer J. OAuth 2.0 Token Introspection. RFC 7662. https://datatracker.ietf.org/doc/html/rfc7662 (дата обращения: 07.07.2023).

[49] OAuth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens. RFC 8705. February 2020. https://datatracker.ietf.org/doc/html/rfc8705 (дата обращения: 22.01.2024).

[50] Fielding R., Reschke J. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231. June 2014. https://datatracker.ietf.org/doc/html/rfc7231 (дата обращения: 22.01.2024).

[51] A UniveRSAlly Unique IDentifier (UUID) URN Namespace. RFC 4122. https://datatracker.ietf.org/doc/html/rfc4122 (дата обращения: 22.01.2024).

[52] Lodderstedt T., Bradley J., Labunets A., Fett D. OAuth 2.0 Security Best Current Practice. 5 June 2023. https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics-23 (дата обращения: 22.01.2024).

[53] Lodderstedt T., McGloin M., Hunt P. OAuth 2.0 Threat Model and Security Considerations. RFC 6819. January 2013. https://datatracker.ietf.org/doc/html/rfc6819 (дата обращения: 22.01.2024).

[54] Положение Банка России от 17.10.2022 № 808-П «О требованиях к обеспечению защиты информации при осуществлении деятельности в сфере оказания профессиональных услуг на финансовом рынке в целях противодействия осуществлению незаконных финансовых операций, обязательных для лиц, оказывающих профессиональные услуги на финансовом рынке, к обеспечению бюро кредитных историй защиты информации, указанной в статье 4 Федерального закона «О кредитных историях», при ее обработке, хранении и передаче сертифицированными средствами защиты, а также к сохранности информации, полученной в процессе деятельности кредитного рейтингового агентства».

УДК 681.3.06

Ключевые слова: финансовая технология, открытые программные интерфейсы, токен доступа, авторизация, OpenID Connect