**Министерство образования и науки Российской Федерации**

**Государственное образовательное учреждение   
высшего профессионального образования   
«Московский физико-технический институт (государственный университет)»**

ФИЗТЕХ-ШКОЛА радиотехники и компьютерных технологий

КАФЕДРА защиты информации

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ДИПЛОМНОЙ РАБОТЕ***

***НА ТЕМУ:***

ЗАЩИЩЕННЫЙ ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС УПРАВЛЕНИЯ КРИПТОВАЛЮТНЫМИ АКТИВАМИ НА БАЗЕ ПЛАТФОРМЫ МКТ

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант по \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант по \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант по \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант по

организационно-экономической части **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант по охране труда и экологии **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc16523)

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 3](#_Toc8413)

[ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ 6](#_Toc29317)

[АННОТАЦИЯ 7](#_Toc22950)

[ВВЕДЕНИЕ 8](#_Toc13346)

[1 ПРЕДПРОЕКТНОЕ ОБСЛЕДОВАНИЕ 9](#_Toc18564)

[1.1 Постановка задачи на разработку 9](#_Toc25996)

[1.2 Классификация криптовалютных активов 9](#_Toc28376)

[1.2.1 Криптовалюты 9](#_Toc9467)

[1.2.2 Токены 11](#_Toc29420)

[1.3 Классификация средств управления криптовалютными активами 11](#_Toc17592)

[1.3.1 Криптовалютные кошельки 13](#_Toc13930)

[1.3.2 Криптовалютные обменники 15](#_Toc7745)

[1.3.3 Криптовалютные биржи 18](#_Toc6506)

[1.3.4 Боты-трейдеры 25](#_Toc5014)

[1.3.5 Майнинг пулы 29](#_Toc5896)

[1.3.6 Перечень потенциально опасных и безопасных сервисов 31](#_Toc22526)

[1.4 Обеспечение условий для безопасного управления криптовалютными активами 34](#_Toc4008)

[1.5 Аппаратный состав ПАК 35](#_Toc9144)

[1.5.1 Аппаратная генерация ключей подписи 35](#_Toc30284)

[1.5.2 Выбор способа реализации криптоядра 37](#_Toc21112)

[1.5.3 Выбор аппаратной платформы 38](#_Toc424)

[1.6 Функциональность ПАК 42](#_Toc10922)

[1.6.1 Функциональность РКБ 42](#_Toc18143)

[1.6.2 Функциональность, необходимая для холодного хранения 43](#_Toc15297)

[1.6.3 Функциональность, необходимая для горячего хранения 47](#_Toc4870)

[1.7 Программный состав ПАК 48](#_Toc29735)

[1.7.1 ПО для обеспечения необходимой функциональности РКБ 48](#_Toc16866)

[1.7.2 Перечень пользовательских приложений, необходимых для обеспечения требуемой функциональности 49](#_Toc21386)

[1.7.3 Обеспечение корректного взаимодействия ПО с РКБ 50](#_Toc894)

[1.8 Перечень задач, подлежащих решению в процессе разработки 51](#_Toc1165)

[2 КОНСТРУКТОРСКАЯ ЧАСТЬ 52](#_Toc20364)

[2.1 Реализация функциональности РКБ 52](#_Toc16696)

[2.2 Реализация демона для перехвата вызовов функций 57](#_Toc31917)

[2.2.1 Вызов криптографических функций при работе с ключами 58](#_Toc16271)

[2.2.2 Выбор метода перехвата и переадресации вызова функций 58](#_Toc25364)

[2.2.3 Полный процесс перехвата функции 64](#_Toc4154)

[2.3 ==/\*TO DO\*/== 67](#_Toc27932)

[3 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 67](#_Toc26410)

[4 ЗАКЛЮЧЕНИЕ 68](#_Toc19921)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 69](#_Toc18073)

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

**Блокчейн** — это распределенная база данных, в которой данные хранятся в виде последовательного набора блоков, при этом каждый блок включает в себя значение хэш-функции от предыдущего блока. Эта база данных хранится одновременно на множестве узлов сети.

**Блокчейн сеть** — совокупность узлов сети, на которых хранится база данных блокчейна, и объединенных через интернет, называется блокчейн сетью.

**Демон** — компьютерная программа в системах класса UNIX, запускаемая самой системой и работающая в фоновом режиме без прямого взаимодействия с пользователем.

**Доверенная вычислительная среда** – это фрагмент среды электронного взаимодействия, для которого установлена и поддерживается в течение заданного интервала времени целостность объектов и целостность взаимосвязей между ними.

**Доверенная загрузка** — это загрузка различных операционных систем только с заранее определенных постоянных носителей (например, только с жесткого диска) после успешного завершения специальных процедур: проверки целостности технических и программных средств ПК (с использованием механизма пошагового контроля целостности) и идентификации/аутентификации пользователя.

**Мастер ключи** — ключи из ключевой пары из криптографической системы с открытым ключом, из которой по протоколу BIP 32 генерируются дочерние иерархически детерминированные ключевые пары.

**Криптовалюта** — разновидность цифровой валюты, создание и контроль за которой базируются на криптографических методах и ведутся в блокчейн сетях.

**Криптографическая система с открытым ключом** — система асимметричного шифрования или электронной подписи (ЭП), при которой ключ проверки подписи (открытый ключ) передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для проверки ЭП и для шифрования сообщения. Для генерации ЭП и для расшифровки сообщения используется ключ подписи (закрытый ключ).

**Криптовалютные ключи** — ключи из ключевой пары из криптографической системы с открытым ключом, которая генерируется по семейству протоколов BIP.

**Криптоядро** — аппаратный модуль, осуществляющий все криптографические операции, включающие шифрование информации, защиту ключей шифрования, безопасную передачу ключей шифрования, обеспечение целостности данных, служебного ПО, и служебной информации.

**Майнинг** — деятельность по поддержанию распределенной платформы и созданию новых блоков блокчейна с возможностью получить вознаграждение в форме эмитированной валюты и комиссионных сборов. Клиент, выполняющий майнинг называется майнером.

**Несанкционированный доступ к информации (НСД)** — доступ к информации, нарушающий правила разграничения доступа с использованием штатных средств, предоставляемых средствами вычислительной техники или автоматизированными системами.

**Пользователь** — субъект доступа к объектам (ресурсам) СВТ.

**Системный вызов** — механизм исполнения процедур системной задачи по запросу прикладной задачи. Для прикладной задачи системный вызов представляется обычной функцией языка Си.

**Смарт-контракт** — это компьютерный алгоритм, хранимый и исполняемый в блокчейн сети. В теле смарт-контракта описаны его условия и исходы исполнения.

**Узел сети** — устройство, соединённое с другими устройствами как часть компьютерной сети.

**Управление криптовалютными активами** — деятельность клиента по манипуляции криптовалютами и токенами, направленная на сохранение и приумножение активов, а также совершение сделок и транзакций.

**Криптовалютная транзакция** — информация о действии или последовательности действий, направленных на создание, выпуск, обращение криптовалютных активов.

# 

# ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

|  |  |
| --- | --- |
| API | Application Programming Interface |
| BIP | Bitcoin Improvement Proposal |
| АРМ | Автоматизированное рабочее место |
| ВрПО | Вредоносное программное обеспечение |
| НСД | Несанкционированный доступ |
| ОС | Операционная система |
| ПАК | Программно-аппаратный комплекс |
| ПК | Персональный компьютер |
| ПО | Программное обеспечение |
| СКЗИ | Средство криптографической защиты информации |
| ФДСЧ | Физический датчик случайных чисел |
| ЭП | Электронная подпись |

# АННОТАЦИЯ

В данном документе приведена пояснительная записка к проекту программно аппаратного комплекса, предназначенного для безопасного управления криптовалютными активами, приведен аппаратный и программный состав разрабатываемого комплекса, описание связей между его компонентами, описание алгоритмов и функций программного комплекса.

Приложением к пояснительной записке является техническое задание на дипломный проект. Так же в процессе разработки сформирован набор программных документов, таких как спецификация, описание программы, программа и методика испытаний, ведомость эксплуатационных документов, формуляр, руководство программиста, руководство оператора, оформленных в соответствии с требованиями ЕСПД (ГОСТ 19).

# ВВЕДЕНИЕ

При совершении операций с криптовалютными активами неизбежен контакт с сетью. Любое подключение к интернету сопряжено с риском компрометации ключей подписи. Для снижения риска применяются отчуждаемые аппаратные криптовалютные кошельки, основное свойство которых заключается в том, что ключи подписи никогда не покидают кошелек и все криптографические операции выполняются также внутри устройства.

Все существующие подобные решения на рынке не дают гарантии, что операции формирования транзакции, ее передачи в аппаратный кошелек на подпись и ее трансляции в сеть проводятся на доверенном компьютере. Кроме того, для формирования и трансляции транзакции программному кошельку на компьютере необходимо обновить блокчейн, то есть за продолжительное время необходимо скачать большой объем информации. За время скачивания необходимой информации в систему также может скачаться и начать исполняться вредоносный код. Это может привести, к примеру, к подмене транзакции, отправляемой в отчуждаемое устройство на подпись, и к дальнейшему криптоанализу подписанной транзакции злоумышленником.

Таким образом возникает противоречие между потенциальными уязвимостями при проведении криптовалютных транзакций и потребностями общества в возможности безопасного проведения таких транзакций. Данная работа посвящена решению этого противоречия.

**Цель работы**: разработать программно-аппаратный комплекс управления криптовалютными активами на базе платформы МКТ, обеспечивающий достаточный уровень защищенности.

# ПРЕДПРОЕКТНОЕ ОБСЛЕДОВАНИЕ

## Постановка задачи на разработку

Разрабатываемый ПАК направлен на усиление защиты при совершении криптовалютных операций. Основанием для разработки является задание на дипломный проект, подписанное консультантами и руководителем дипломного проекта и утвержденное заведующим кафедрой защиты информации ФРКТ МФТИ.

## Классификация криптовалютных активов

Прежде чем приступить к описанию механизмов управления криптовалютными активами, необходимо понять, что входит в перечень этих активов. Перечислим и классифицируем все виды криптовалютных активов.

### Криптовалюты

Первый вид активов, криптовалюты — разновидность цифровой валюты, создание и контроль за которой базируются на криптографических методах и ведутся в блокчейн сетях.

Майнинг служит для производства денежной массы [1]. Каждый раз, когда майнер создает новый блок, ему за это полагается награда - определенное число монет, которые он может потом потратить, тем самым запуская в сеть новые средства. Таким образом майнинг является основным источником всех криптовалютных активов.

В большинстве криптовалют действует система UTXO (Unspent Transaction Output), аналогичная биткойну: каждая транзакция должна быть потрачена полностью. То есть если у Алисы на аккаунте есть 5 BTC, и она хочет перевести Бобу 1 BTC, то она должна сделать транзакцию с двумя выходами: один для боба (1 BTC) и второй для «сдачи» для Алисы на ее собственный адрес (4 BTC).

В структуре любого блока первой всегда идет так называемая coinbase транзакция — именно она отправляет вознаграждение на адрес майнера. В отличии от обычных транзакций, coinbase transaction не тратит в качестве входов выходы из UTXO pool. Вместо этого у нее указан только один вход, называемый coinbase, который "создает" монеты из ничего. Выход у такой транзакции тоже только один. Он отправляет на адрес майнера награду за блок плюс сумму комиссий со всех транзакций в блоке.

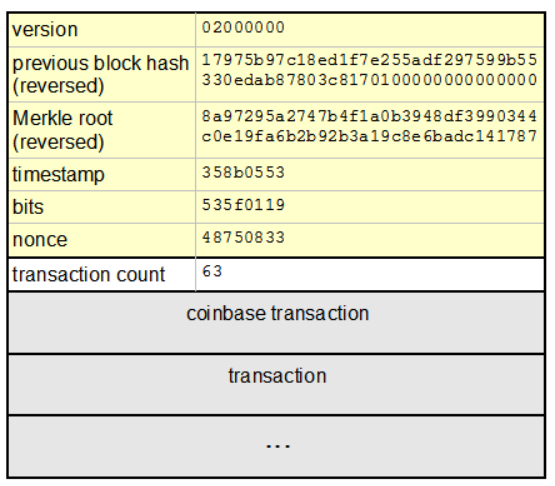


Рисунок 1. Логическая структура Bitcoin блока

В блокчейн имплементирована программа с внутренним хранилищем в которой хранится реестр монет - массив ассоциаций аккаунтов (ключей проверки подписи ) с суммами. Полный доступ к данному реестру имеет только эта программа, она же предоставляет всем желающим возможности просматривать суммы на аккаунтах и переводить суммы между ними, предварительно выполнив проверку на наличие у обратившегося доступа к тому аккаунту, с которого он пытается передать деньги, а также на не превышение суммы на балансе этого аккаунта. Проверка доступа основывается на проверке подписи транзакции, осуществленной с помощью ключа подписи. Если проверка прошла, то запрос на транзакцию будет отослан в сеть, после чего запрос будет принят или отвергнут на основании алгоритма консенсуса, принятого в данном блокчейне.

Подытожим. Для того чтобы клиент мог воспользоваться криптовалютой, как денежным средством, необходимо, чтобы:

* только у клиента был доступ к его ключу подписи;
* в реестре содержалась запись о том, что необходимое количество валюты проассоциировано с аккаунтом клиента;

### Токены

Второй вид криптовалютных активов, токен — это смарт-контракт определенного формата [2], оперирование которым осуществляется вторым, родительским смарт-контрактом. Основная часть наиболее популярных на сегодняшний день токенов формируется в сети Ethereum в соответствии со стандартом ERC-20. В этом стандарте описаны условия (необходимые методы, функции и поля), при которых смарт-контракт считается токеном [3].

Токены бывают трех видов [4]:

* Токены-акции, свидетельствующие о внесении криптовалютных средств в капитал эмитента токенов и дающие обладателю токена право на получение части прибыли в виде дивидендов. То есть для эмитента они нужны для привлечения финансирования, а для покупателя - для получения прибыли в будущем.
* Аппкойны, или токены приложений (токены жетоны). Эти токены предназначены в первую очередь для использования внутри приложения или сервиса.
* Кредитные токены, которые используются с целью краткосрочного заимствования денежных средств эмитентом с дальнейшей выплатой процентной ставки от суммы займа. Эти токены являются аналогом облигаций.

Подытожим информацию по двум подразделам. Криптовалютные активы делятся на криптовалюты и токены. Токены бывают трех видов: токены акции, кредитные токены и токены жетоны (аппкойны).

## Классификация средств управления криптовалютными активами

Перейдем к механизмам управления криптовалютными активами.

Существует несколько видов сервисов и средств работы с криптовалютными активами, которые и предоставляют механизмы управления этими активами. Перечень таких механизмов различается в зависимости от каждого конкретного сервиса.

Средства и механизмы, предоставляемые этими средствами, могут быть реализованы корректно и некорректно с точки зрения безопасности клиента. Перечислим и систематизируем все средства работы с криптовалютными активами, а также поделим их на две категории: безопасные средства и потенциально опасные средства. Защищенный ПАК управления криптовалютными активами должен обеспечивать корректную работу пользователя с выделенными безопасными средствами.

При описании процесса работы сервисов в первую очередь будем обращать внимание на то, как в системе ведется работа с ключами подписи пользователя, так как именно они представляют наибольшую ценность для потенциальных злоумышленников.

Если ключевая пара генерируется или хранится третьей стороной, то существует риск того что эта сторона может сохранить себе копию ключей подписи и в будущем без ведома клиента несанкционированно совершить какую-либо транзакцию. Подпись транзакции третьей стороной, подразумевает передачу ключа подписи этой стороне, что также подразумевает риск несанкционированного копирования ключа.

Часть сервисов предоставляет возможность установки приложений на компьютер пользователя. Если выбирать, как взаимодействовать с сервисом: через браузер или через официальное приложение, устанавливаемое на компьютер пользователя, то для минимизации риска целесообразно выбрать второй вариант. В этом случае ПО, обеспечивающее функциональность сервиса, работает на стороне клиента, благодаря чему появляется возможность контролировать целостность этого ПО и корректность его работы.

Поэтому безопасный сервис, должен удовлетворять как минимум следующим требованиям:

* Для работы с сервисом клиент использует самостоятельно сгенерированную ключевую пару.
* Ключ подписи хранится только на стороне клиента.
* Подпись транзакций осуществляется также на стороне клиента.

### Криптовалютные кошельки

Для взаимодействия пользователя с блокчейном используют семейство программ, называемых криптовалютными кошельками. Их основная функция заключается в получении из управляющей программы блокчейна информации о сумме на аккаунте пользователя и отправлении в эту программу информации о передаче монет, с сигнатурой для идентификации пользователя как владельца. Криптовалютный кошелек должен уметь просканировать реестр блокчейна, прочитать там баланс пользователя, изменить его и баланс получателя транзакции.

Кошельки различаются по количеству поддерживаемых монет. Они бывают одновалютные, то есть поддерживающие операции только с одной криптовалютой, и мультивалютные, то есть поддерживающие операции с несколькими криптовалютами, а также с токенами стандарта ERC-20.

Рассмотрим классификацию криптовалютных кошельков:

* Горячие — это кошельки, имеющие постоянное соединение с интернетом.
  + - Онлайн кошельки — горячий кошелёк, запускаемый через интернет браузер на сайте. В этом варианте ключи подписи генерируются и хранятся на сервере.
    - «Тонкие» локальные кошельки — кошельки, устанавливаемые на компьютере пользователя, при этом блокчейн на устройство не скачивается (в том числе и при первой синхронизации). Когда возникает необходимость прочитать данные из блокчейна или произвести транзакцию, то идет обращение к серверам или к соседним узлам сети.
    - «Толстые» локальные кошельки — кошельки, устанавливаемые на компьютере пользователя, которые общаются только со своим, заранее скаченным и постоянно обновляемым блокчейном.
* Холодные — это кошельки, не имеющие постоянное соединение с интернетом и блокчейном [5]. При этом, с них не совершаются транзакции.
  + - Аппаратные кошельки — кошельки на аппаратных отчуждаемых носителях (обычно флеш накопителях или смарт картах), ведущие взаимодействие с блокчейном через локальный кошелек или же через сайт. Ключ подписи не покидает устройство, все криптографические операции выполняются также внутри кошелька.

Все типы горячих кошельков являются потенциально небезопасными, потому что канал связи или сервера могут быть скомпрометированы. Холодные кошельки в пассивном режиме (без соединения с интернетом и без совершения транзакций) являются безопасными, при условии, что компьютер является доверенным, и мы можем гарантировать, что у злоумышленников нет к нему доступа. Однако при переходе кошелька в активный режим (совершение транзакции через интернет) на определённый промежуток времени риск компрометации значительно возрастает, при этом, чем выше хранимая сумма, тем выше риск. После совершения транзакции и отключения интернета риск компрометации ключей снижается, но остается, так как злоумышленники, применяя аналитические методы и вредоносное программное обеспечение, все равно могут получить доступ к ключам подписи.

Существует модификация модели холодного хранения, которая применяется для уменьшения риска компрометации в момент совершения транзакции. Такая схема называется офлайн подписью транзакций. Рассмотрим схему подробно:

* Система состоит из двух компьютеров, один из них всегда отключен от интернета, второй - подключен к сети
* На первом компьютере с помощью кошелька создается ключевая пара, затем импортируется ключ проверки подписи на второй компьютер. После чего на адрес, ассоциированный с этим ключом, можно перевести необходимую сумму для хранения.
* Когда возникает необходимость в переводе, то на втором компьютере создается неподписанная транзакция (указывается адреса отправителя и получателя, а также объём средств), которая затем копируется на съемный носитель и импортируется на нем на первый компьютер, неподключенный к интернету.
* На первом компьютере транзакция подписывается после чего точно также на съемном носителе импортируется на первый компьютер и транслируется в сеть.

Подытожим. Безопасным можно считать только «холодный» вариант хранения ключей. Для совершения транзакций целесообразно применять вышеописанную схему с офлайн подписью.

### Криптовалютные обменники

Для покупки/продажи криптовалюты, а также ее обмена на другие валюты используются криптовалютные обменники. В ходе обследования были изучены 20 наиболее популярных криптовалютных обменников [6].

1. Netex24
2. Kassa
3. WW-pay
4. Xchange
5. 365Cash
6. 24paybank
7. NixExchange
8. F-Change
9. Baksman
10. FastChange
11. Payget
12. 60cek
13. El-Change
14. Top-Exchange
15. Bitoс
16. Ychanger
17. Prostocash
18. Platov
19. Ramon Cash
20. BTCBIT

Принцип их работы следующий:

* Пользователь выбирает валютную пару, то есть указывает, какие валюты он хочет обменять, указывает сумму для обмена и размещает заявку на обмен.
* В случае покупки криптовалюты за реальные деньги пользователь вводит номер электронного кошелька/банковской карты, с которой будут списываться средства, и вводит адрес криптовалютного кошелька, на который будет поступать купленная криптовалюта. После чего ему будет предложено совершить платеж на банковскую карту получателя.
* В случае продажи криптовалюты за реальные деньги пользователь вводит номер кошелька/банковской карты, на которую будет получены средства. И вводит адрес кошелька, с которого будет проводиться транзакция. После чего ему будет предложено совершить платеж на адрес кошелька получателя.
* Важной деталью является то, что операции и транзакции проводятся не на самом сайте, а в сторонних сервисах. К примеру, сделка по продаже Bitcoin инициируется через обменник на одном компьютере, а подпись транзакции ключом подписи осуществляется через криптовалютный кошелек на другом компьютере.
* В зависимости от политики сервиса, платеж пройдет только после нескольких подтверждений транзакции в блокчейне. Обычно требуется от 1 до 4 подтверждений.
* Зачастую, при совершении сделки требуется указать контакты для связи: электронная почта, телефон. Это необходимо на случай внештатной ситуации: изменение курсов, задержки обработки, сбоев на сервере.

Как уже было сказано выше, ключи подписи пользователя не контактируют с данными сервисами, и подпись транзакций осуществляется отдельным кошельком или сервисом. Главные риски для клиента при работе с обменниками заключаются в следующем:

1. Нет никаких гарантий, что в случае совершения транзакции по нужному адресу кошелька сервис выполнит свою часть сделки обмена.
2. Адрес кошелька получателя передается клиенту вместе с WEB страницей по протоколу htpps. Если сервера или интернет соединение скомпрометированы, то злоумышленники могут подменить адрес кошелька получателя, и после завершения транзакции валюта будет переведена злоумышленникам.

Подытожим. Криптовалютный обменник предназначен для обмена криптовалютных пар, для покупки/продажи криптовалют за фиатные деньги. Закрытые ключи пользователя не контактируют с данным типом сервисов, поэтому данный тип сервисов можно считать надежным с точки зрения сохранности ключей подписи клиента. Однако остаются риски того, что получатель может не выполнить свою часть обмена, а также, что адрес получателя могут подменить.

### Криптовалютные биржи

Следующий тип криптовалютных сервисов — криптовалютные биржи (криптобиржи). Криптобиржа является торговой платформой, позволяющей пользователям производить покупку и продажу, хранение и обмен криптовалют. Принцип функционирования таких площадок аналогичен валютным, товарным и фондовым биржам: цены определяются рынком, по принципу «предложение/спрос» для каждого токена. Криптобиржи бывают централизованные и децентрализованные.

#### Централизованные биржи

Начнем с централизованных бирж. Централизация в данном случае означает, что пользователи доверяют свои средства третьей стороне, доверенному посреднику, при работе с любыми криптовалютными активами.

В некоторых странах централизованные криптобиржи регулируются государством, хотя и менее строго, нежели фиатные биржи. В связи с этим, централизованные биржи выставляют на торги лишь проверенные криптовалюты. Поэтому на таких биржах меньше вероятность наткнуться на мошенников.

В ходе обследования были изучены 15 наиболее популярных криптовалютных централизованных бирж с наибольшим объемом торгов [7]:

1. Binance
2. EXMO
3. BTC-Alpha
4. Huobi
5. Kucoin
6. OKEx
7. LiveCoin
8. YoBit
9. Coinbase
10. HitBTC
11. Poloniex
12. Kraken
13. Bitfinex
14. Bittrex
15. CEX

Рассмотрим, как пользователь взаимодействует с централизованными криптобиржами.

Пользователь проходит регистрацию на сервисе, указывает телефон и почту. Если он хочет получить доступ ко всему функционалу биржи, то ему необходимо пройти полную процедуру KYC (know your customer) и указать свои паспортные и некоторые банковские данные. Авторизация на большинстве сервисов двухфакторная: ввод логина/пароля и смс кода. После авторизации пользователю открывается доступ к финансовым операциям.

Для начала работы необходимо пополнить баланс аккаунта. Это можно сделать, переведя фиатные деньги на банковский счет биржи и получив взамен криптовалюту на баланс аккаунта, а также с помощью стороннего криптовалютного кошелька, осуществив транзакцию по адресу кошелька, привязанного к аккаунту на бирже.

Адрес для перевода каждый раз разный, но все эти адреса принадлежат пользователю, так как генерация ключей для пользователей осуществляется биржей по протоколу иерархической генерации ключей BIP32 [8].

Суть этого протокола заключается в том, что все используемые ключевые пары порождаются из одного общего для всех ключей секрета. Особенность состоит в том, что есть возможность из одного секрета породить сколько угодно пар ключей для электронной подписи. Можно использовать новые адреса для каждого входящего платежа и сдачи, при этом все порожденные из основного секрета личные ключи, друг с другом никак не связаны, то есть нельзя проследить связь между порожденными адресами (определить, что все они принадлежат одному пользователю), а имея порожденный личный ключ, нельзя восстановить изначальный общий секрет. Все закрытые ключи хранятся в базе данных сервиса в зашифрованном виде.

Как и в случае классических бирж, пользователи доверяют свои капиталы криптобирже, и она хранит их, а точнее она хранит закрытые ключи клиентов. В этом заключается главный нюанс - у пользователя нет доступа к закрытому ключу из ключевой пары криптовалютного кошелька. Пользователь может видеть свой открытый ключ, но непосредственно подпись транзакций осуществляется на стороне сервиса.

На биржах существует возможность вывода криптовалютных средств на произвольный адрес криптовалютного кошелька, который вводится в соответствующее поле на сайте. Нужный адрес получателя передается сервису по протоколу https. Если сервера или интернет соединение скомпрометированы, то злоумышленники могут подменить адрес кошелька получателя, и после заверешения транзакции валюта будет переведена злоумышленникам.

Несмотря на все меры безопасности, использование таких сервисов все равно является источником повышенного риска для пользователей.

Во-первых, авторизация по логину/паролю является ненадежной (логин и пароль можно подсмотреть, можно считать кейлогером, можно похитить с помощью фишинга и так далее). Доступ к возможности осуществлять подпись ключом подписи осуществляется с помощью логина и пароля. Система является безопасной ровно настолько, насколько безопасен ее самый уязвимый элемент. Таким образом получается, что из-за уязвимости обычных паролей - уязвима вся система, и злоумышленник при краже паролей, получает доступ к закрытому ключу.

Во-вторых, генерация и хранение ключей подписей не на стороне пользователя, а на стороне сервера - это грубое нарушение основ идеологии ассимитричной криптографии, нивелирующее все ее основные достоинства. Существует вероятность того, что сторона, осуществляющая хранение закрытых ключей, захочет осуществить несанкционированный доступ к этим ключам или дать этот доступ третьим лицам. Это дополнительный экономический риск.

В-третьих, из-за уязвимостей протокола https существует вероятность: 1) подмены адреса кошелька клиента при пополнении баланса аккаунта; 2) подмены адреса получателя при выводе криптовалюты с биржи.

В результате пользователь вынужден полагаться на то, что злоумышленники не украдут у него логин и пароль, или не взломают базы данных с ключами; на то, что владельцы сервиса не захотят воспользоваться ключевой парой в своих корыстных целях. Возникает дополнительный риск потери своих денежных средств.

Классические финансовые биржи строго регулируются законодательством, потому в случае атаки на систему, пользователю возместят потерянные средства. Для криптовалютных активов пока нет соответствующих законов, поэтому, в случае атаки и утери средств, пользователь не сможет вернуть свои средства.

#### Децентрализованные биржи

Перейдем к децентрализованным криптовалютным биржам (DEX, decentralized exchange). Это биржи, которые работают на основе распределенного реестра, при этом не хранят закрытые ключи и персональные данные пользователей на своих серверах и выступают только платформой для поиска совпадений по заявкам на покупку или продажу активов пользователей.

DEX биржи стали развиваться относительно недавно, поэтому на рынке их не так много. В ходе обследования были изучены 10 наиболее популярных децентрализованных бирж [9]:

1. IDEX
2. Radar Relay
3. Kyber Network
4. Switcheo Network
5. Waves DEX
6. DEEX
7. Bisq Network
8. Airswap
9. Crypto Bridge
10. EtherDelta

Торговля на таких платформах происходит напрямую между участниками (peer-to-peer) без каких-либо финансовых посредников. Хоть биржи и используют свои собственные серверы для хранения данных о торгах и заявок на покупку/продажу активов пользователей, однако закрытые ключи при этом хранятся у самих пользователей. Таким образом DEX биржа устраняет необходимость в доверенном посреднике, и обеспечивают полную анонимность пользователя.

Большая часть DEX бирж имеют открытый исходный код, с которым можно ознакомиться на Git. Часть бирж для работы с собой предоставляет сайт или приложение, часть - только приложение (к примеру Bisq Network). Вариант с приложением, устанавливаемым на компьютер пользователя, является предпочтительным, так как в этом случае можно более полно контролировать процесс взаимодействия пользователя и сервиса.

Серверная часть DEX состоит из смарт-контракта, алгоритма для трейдинга и арбитра обработки транзакций. Смарт-контракт осуществляет хранение всех активов и выполнение всех транзакций. Если злоумышленники получат доступ к серверам, то они не смогу вывести средства из смарт-контрактов, если они не будут знать ключей подписи пользователей. Поэтому атаки криптоаналитиков или закрытие самой биржи не приведут к потери средств клиентов. Также у DEX биржи нет единой точки входа, через которую можно было бы получить доступ ко всем данным, что усложняет работу для взломщиков и делает саму атаку бессмысленной.

Децентрализованные биржи работают по семейству протоколов «0x». В нем описывается принципы децентрализованного обмена валютой/токенами внутри смарт-контракта, а не внутри самого блокчейна [10]. Согласно этому протоколу, чтобы совершить сделку и записать ее в смарт контракт, оба участника обмена должны подписать транзакцию своими закрытыми ключами, при чем подпись осуществляется вне биржи, то есть на стороне пользователей (в браузере или в установленном приложении).

Каждая транзакция представляет собой пакет данных, содержащий параметры заказа и соответствующую подпись. Параметры транзакции конкатенируются и хэшируются до 32 байт с помощью функции Keccak SHA3. Инициатор заказа подписывает хэш заказа своим закрытым ключом для получения подписи ECDSA. Смарт-контракт может аутентифицировать подпись инициатора транзакции с помощью функции ecrecover(), которая принимает на вход хэш транзакции и подпись хэша в качестве аргументов и возвращает открытый ключ кошелька, который произвел подпись. Если открытый ключ, возвращенный ecrecover(), равен адресу заявленного отправителя, подпись является подлинной.

Процесс работы с DEX биржей выглядит следующим образом. При регистрации на сервисе, пользователю необходимо добавить адрес своего кошелька в смарт-контракт биржи. Для этого ему необходимо подписать транзакцию с помощью своего закрытого ключа. Подпись осуществляется либо приложением, либо скриптами внутри браузера. Есть несколько способов предъявить ключ подписи:

1. Ввести информацию файла хранилища ключей
2. Ввести ключ подписи, или seed фразу
3. Использовать для подписи браузерный плагин, обеспечивающий работу горячего кошелька (к примеру Metamask)
4. Использовать аппаратный холодный кошелек (к примеру Trezor)

Ключи по алгоритму «0х» не отправляются сервису, а отправляется только подписанная транзакция. Если, мы можем гарантировать целостность и корректную работу браузера, то 1 и 2 варианты - безопасны. Горячие кошельки не безопасны, поэтому третий вариант также небезопасен. При использовании аппаратных кошельков, закрытые ключи не покидают отчуждаемое устройство, и подпись осуществляется также на этом устройстве. Поэтому вариант 4 - также безопасен.

После чего необходимо пополнить баланс аккаунта на бирже. Клиент переводит криптовалюту/токены со своего кошелька на свой адрес внутри смарт-контракта. Для того чтобы пополнить баланс, вывести оттуда средства или совершить перевод на другой адрес, необходимо совершить подпись транзакции закрытым ключом пользователя. Эту транзакцию в неподписанном виде можно найти на бирже, или найти с помощью сервисов чтения блокчейна, и импортировать на компьютер пользователя, чтобы уже в доверенной среде осуществить подпись.

Подытожим. Криптовалютные биржи используются для купли-продажи, хранения и обмена криптовалют. Криптовалютные биржи бывают двух видов: централизованные и децентрализованные. Их сходства и различия рассмотрим в таблице 3.

Таблица 1.

Сравнение централизованных и децентрализованных криптовалютных бирж.

|  | **Централизованные биржи** | **Децентрализованные** |
| --- | --- | --- |
| **Где хранятся ключи?** | Закрытые ключи генерируются и хранятся на сервере. | Закрытые ключи генерируются и хранятся у пользователя. |
| **Где подписываются транзакции?** | Транзакции подписываются на стороне сервера. | Транзакции подписываются на стороне клиента. |
| **Можно ли отслеживать транзакции?** | Все данные транзакций скрыты от пользователей и доступны только серверу. | Все транзакции публичные, и их можно отслеживать через сервисы сканирования блокчейна. |
| **Есть ли доступ к исходному коду биржи?** | Закрытый исходный код у всех бирж. | Открытый исходный код у большей части бирж. |
| **Анонимность** | При регистрации за частую необходимо указывать свои персональные данные. | Полная анонимность. На большей части сервисов регистрация осуществляется только с помощью ключевой пары. |
| **Взлом или блокировка активов пользователя** | Аккаунт пользователя может быть заблокировать или взломан. В этом случае средства будут безвовзратно утеряны. | Аккаунт невозможно заблокировать или взломать на серверной стороне. |
| **Гарантии выполнения сделок.** | Надежность сделок гарантируется только самой биржей. Возможны ситуации, в которых один из контрагентов не выполнит свою часть сделки. | Все сделки фиксируются в смарт-контрактах. Поэтому обе стороны гарантировано выполнят свою часть сделки. |

### Боты-трейдеры

Следующий вид сервисов предназначен для пользователей, которые не хотят или не могут торговать сами. Боты-трейдеры - программы, созданные для автоматической торговли криптовалютой. Бот может представлять собой приложение, установленное на компьютере клиента, или алгоритм, запущенный на сайте, то есть облачно. Облачные боты бывают реализованы внутри самой биржи, или же на отдельном сервисе. Боты бывают и для централизованных, и для децентрализованных бирж.

В ходе обследования были изучены 11 наиболее популярных ботов трейдеров [11]:

1. Centobot
2. Gekko (приложение для ПК)
3. Cap.Club
4. Zenbot (приложение для ПК)
5. RevenueBot
6. Crypto Trader
7. BTC Robot
8. Gunbot (приложение для ПК)
9. Leonardo
10. HaasOnline
11. PHP Trader (приложение для ПК)

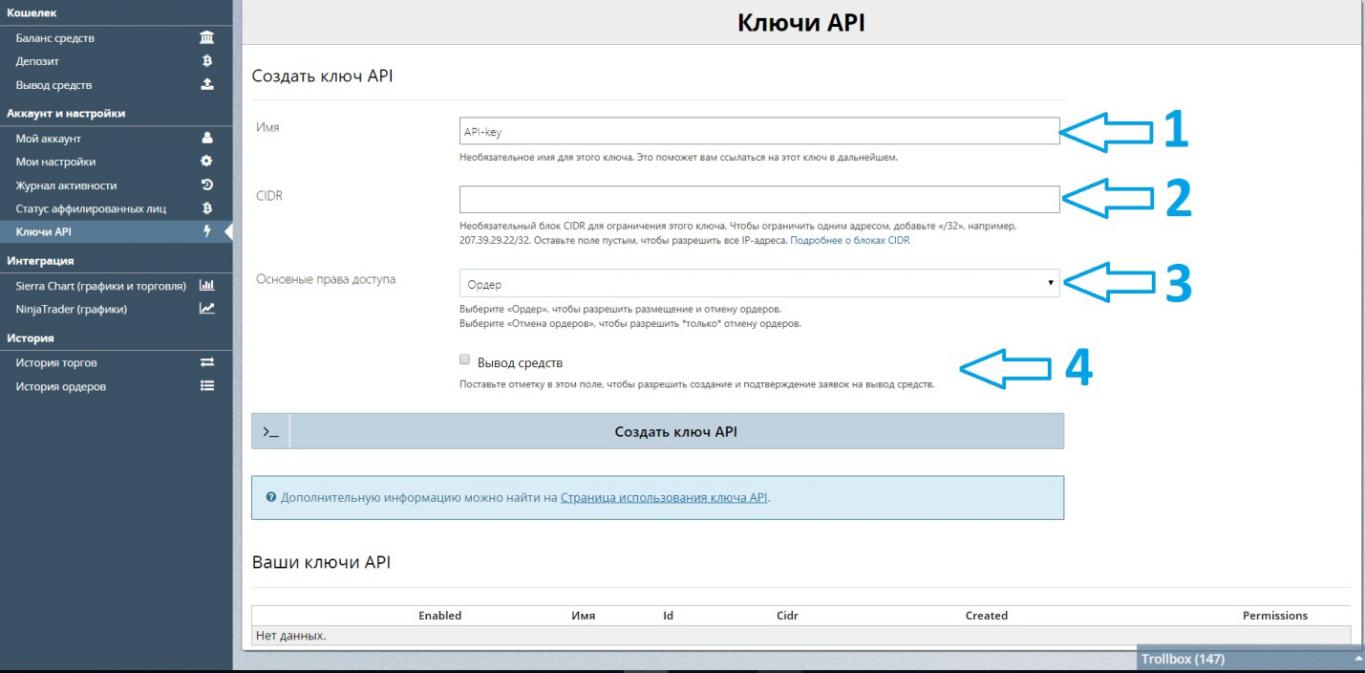
Боты работают по алгоритмам, торговым стратегиям, которые, в свою очередь, создаются на основе технического анализа закономерностей. Для анализа требуются исторические данные об изменениях курса за продолжительный период. После определения закономерностей ведётся проверка стратегии на тех же исторических данных, подсчитывается потенциальные убытки и прибыли. Если результаты устраивают, на их основе создаются правила торговой стратегии робота для биржи криптовалют. Некоторые боты дополнительно используют индикаторы, которые позволяют анализировать текущую ситуацию на рынке, другие выполняются только при возникновении определённых рыночных условий.

У крупных бирж список поддерживаемых ботов ограничен, так же как ограничен список поддерживаемых бирж у ботов.

Есть два наиболее распространённых вида ботов:

* Биржевой: он действует на определенной торговой площадке. Заработок основывается на разнице цен при купле продаже.
* Арбитражный: он действует на двух и более торговых площадках. Заработок основывается на разнице курсов между ними.

Для выполнения торговых операций на биржах боты трейдеры используют торговые API (application programming interface), открытый набор функций, предлагаемых биржей для использования в сторонних приложениях [12]. Для того, чтобы передать свою криптовалюту в дистанционное управление при помощи API, необходимо создать API-ключ на любой бирже, которая это поддерживает.

  
Рисунок 2. Пример работы централизованной криптовалютной биржи Bitmax (страница создания API ключей)

Рассмотрим процесс создания API ключей:

1. Поле «Имя»: имя для ключа. Оно необходимо для удобства, чтобы не запутаться в ключах, если их много.
2. Поле «CIDR»: поле на случай, если клиент решил ограничить доступ по ключу только для определенных IP-адресов. При пустом поле разрешены все IP-адреса.
3. «Основные права доступа»: в этом поле устанавливаются права на перечень сделок, которые может совершать бот.
4. «Вывод средств»: это поле определяет права трейдера, работающего через API, на вывод криптовалюты с клиентского счета.

Ключевая пара создается на стороне сервиса. У пользователя в любой момент есть возможность отключить ее работу, к примеру в случае, если бот, осуществляющий торговлю, начнет вести себя некорректно или подозрительно.

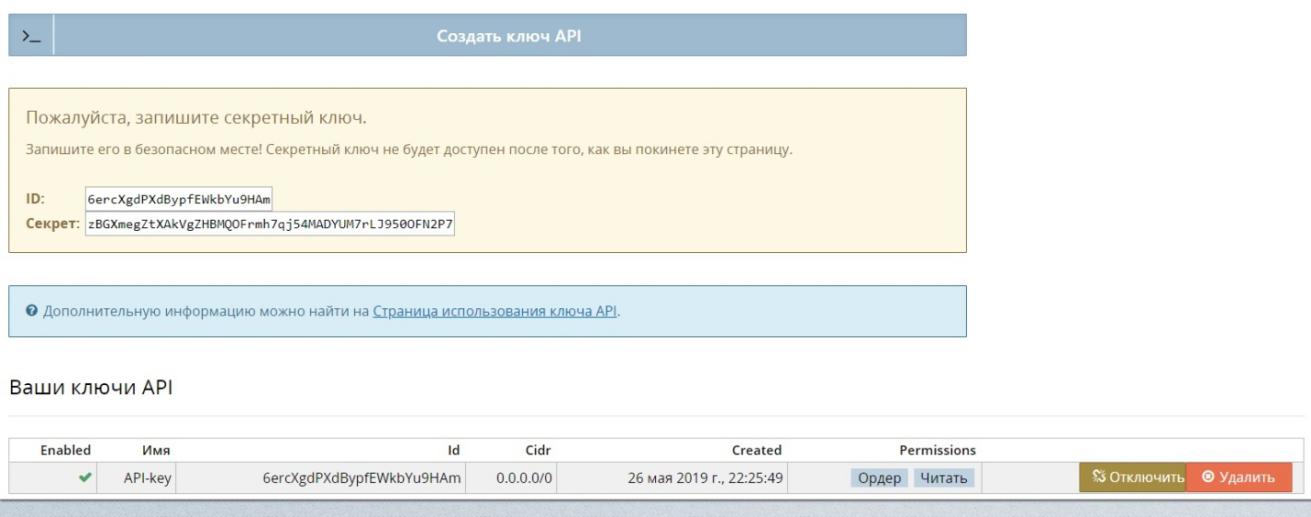


Рисунок 3. Пример работы централизованной криптовалютной биржи Bitmax (страница управления API ключами)

Полученную ключевую пару необходимо ввести в сервис бот трейдинга.

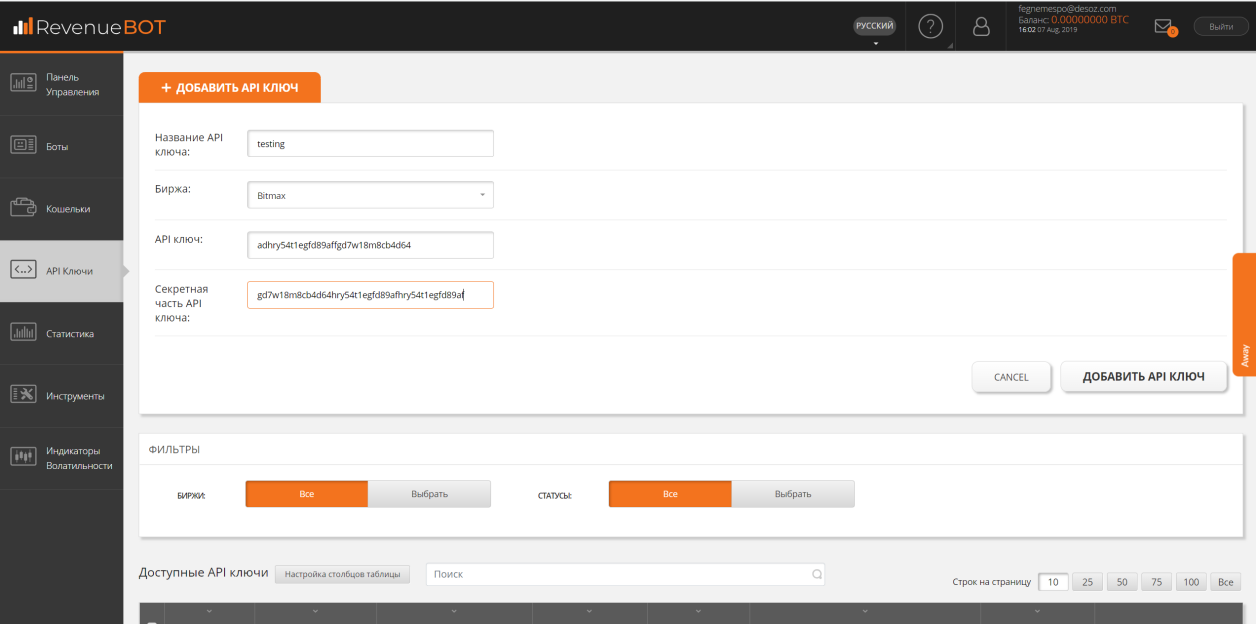


Рисунок 4. Пример работы сервиса бот-трейдинга RevenueBOT (страница ввода API ключей)

Если использовать бота, в виде проверенного ПО, установленного в доверенной среде, то мы можем быть уверены, что бот будет делать именно то, что в нем заложено. Во всех остальных случаях - отдавать криптовалюту на доверенное управлению боту, означает подвергать себя дополнительному риску. Потому что нет никакой гарантии, что бот, расположенный в облаке, вдруг не начнет действовать в интересах сервиса или хакеров. Механизмом отключения API ключей можно просто не успеть воспользоваться. Кроме того, этот механизм так же располагается в облаке, поэтому нет гарантии, что злоумышленники не смогут поменять права API ключей, или выключить/выключить их.

Подытожим. Боты трейдеры - это класс сервисов, предназначенных для автоматической торговли криптовалютными активами на биржах. Некоторые биржи имеют встроенных ботов. Для осуществления дистанционной автоматической торговли на остальных биржах, необходимо предоставить стороннему бот сервису API ключи, которые генерируются биржей. Если биржа устанавливается на ПК пользователя, то ключи генерируются на стороне клиента.

### Майнинг пулы

Пользователь может заниматься майнингом с своего ПК, для этого необходимо установить соответствующее приложение и в настройках указать адрес кошелька, на который будет поступать награда. Однако на момент написания данной работы процесс майнинга является крайне энергозатратным и требовательным к железу, которое быстро изнашивается. Из-за этого майнить самому - экономически не выгодно.

Поэтому майнеры объединяются в майниг пулы. Майнинг пул – это сервер, который делит большую задачу по вычислению подписи блока  на маленькие задачи и раздает их подключенным устройствам.

В ходе обследования были изучены 10 наиболее популярных майнинг пулов [13]:

1. MinerGate
2. Kryptex
3. NiceHash
4. F2pool
5. AntPool
6. ZPool
7. 2Miners
8. Batpool
9. BitClub Network
10. Coinmine

Вклад в общую работу каждого майнера, участвующего в пуле, оценивается с помощью «шар» (от английского «share»). «Шара» – небольшая часть работы по поиску решения хеш-функции для подписания блока, которая выдается пулом майнеру. Собирая шары от майнеров, сервер пула проверяет их валидность. Как только какая-то «шара» удовлетворяет текущим значениям сложности, сервер пула объявляет о подписании блока. После этого пул получает награду за блок и распределяет между майнерами пропорционально количеству переданных «шар» независимо от того, была ли среди них подписавшая блок [14] .

С технической точки зрения для клиента процесс майнинга в пуле выглядит следующим образом: пользователь устанавливает майнер (к примеру Ethminer), и в настройках указывает информацию, необходимую для подключения к пулу. Там же он указывает адрес своего кошелька, на который будет переводиться награда.

При работе с майнинг пулами закрытые ключи пользователя не используются, однако:

* для майнинга внутри майнинг пула необходимо постоянное соединение с интернетом, что значительно повышает риск компрометации системы;
* для получения значимой прибыли, система должна обладать значительными вычислительными мощностями. Чем больше вычислительная мощность, тем универсальнее система, тем сложнее ее защищать. В случае несанкционированного доступа к системе, злоумышленник может использовать вычислительные мощности для перебора паролей или каких-либо других криптографических атак.

Поэтому, в данной работе будем считать работу с данным видом сервисов не безопасным для клиента.

Существует еще одна модель майнинга - облачный майнинг. Если в предыдущей модели вычисления велись на стороне клиентов, а сервер лишь выполнял роль координатора, то в облачной модели все вычисления на себя берет сервер, а клиент лишь платит за аренду вычислительных мощностей.

В ходе работы были изучены 8 наиболее популярных сервисов облачного майнинга:

1. IQ Mining
2. BitDeer
3. Hashing24
4. Hashflare
5. Genesis Mining
6. AlienCloud
7. Nuvoo Mining
8. CCG Mining

Главный минус такой схемы - у пользователя нет доступа к ключевой паре, которая генерируется и хранится на стороне сервиса. Соответственно данный вариант сервисов является рискованным по тем же самым причинам, что и в случае централизованных бирж.

Подытожим. Майнинг, и аналогичные ему процессы, является источником всей существующей криптовалюты. Майнить можно:

* самому, что безопасно, но экономически не выгодно;
* с помощью майнинг пула, в этом случае закрытые ключи никак не задействованы;
* с помощью облачного майнинга, в этом случае у пользователя нет доступа к закрытым ключам.

### Перечень потенциально опасных и безопасных сервисов

Подытожим информацию по всем видам сервисов в таблице 2.

Таблица 2

Сервисы для работы с криптовалютными активами.

| **Тип сервиса** | | | **Описание** | **Безопасность ключей** |
| --- | --- | --- | --- | --- |
| Кошельки | | Горячие | Кошельки, для работы которых необходим интернет. | Ключи могут храниться на сервере или на стороне клиента. При хранении на компьютере клиента, постоянное соединение с интернетом повышает риски компрометации ключей. |
| Холодные | Кошельки, работающие без доступа в интернет. | Если ключи хранятся в доверенной среде без включенного интернета, то им ничего не угрожает. |
| Обменник | | | Сайт, который используют для обмена валюты, покупки и продажи валюты за фиатные деньги. | Ключ подписи криптовалютного кошелька не используется на сервисе, и транзакция осуществляется сторонними средствами. |
| Биржа | Централизованная | | Выступает в роли третьей доверенной стороны. Используется для купли-продажи, хранения и обмена криптовалют. Хранит все данные пользователей на своих серверах. | Ключи подписи генерируются и хранятся на стороне сервиса. Пользователь имеет доступ только к открытому ключу. |
| Децентрализованная | | Выступает в роли площадки для агрегации запросов на сделки. Используется для купли-продажи, хранения и обмена криптовалют. Не сохраняет каких либо данных пользователей. | Сервис позволяет использовать собственно сгенерированные ключевые пары. |
| Бот трейдер | Сторонний | | Предназначен для автоматической торговли криптовалютными активами на биржах. Все торговые процессы инициируются на стороне сервиса. | Для осуществления дистанционной автоматической торговли на биржах, необходимо предоставить бот сервису API ключи от аккаунта клиента на бирже. Эти ключи и сам бот располагаются в облаке. |
| Установленный на стороне клиента | | Предназначен для автоматической торговли криптовалютными активами на биржах. Все торговые процессы инициируются на стороне клиента. | Для осуществления дистанционной автоматической торговли на биржах, необходимо предоставить бот сервису API ключи от аккаунта клиента на бирже. Бот установлен на компьютере пользователя |
| Майнинг сервис | Самостоятельный | | Решение задачи алгоритма консенсуса с помощью приложения для получения награды . | Для получения награды в сеть передается лишь ключ проверки подписи. |
| С помощью пула | | Решение определенной доли от задачи алгоритма консенсуса. Доля назначается сервисом. | Для получения награды в сеть передается лишь ключ проверки подписи. |
| Облачный | | Майнинг осуществляется сторонним сервисом. | Генерация и хранение ключей происходит на стороне сервиса. У пользователя нет доступа к ним. |

Разделим вышеописанные сервисы на потенциально опасные и безопасные.

Подробно уязвимости и недостатки потенциально опасных для клиента сервисов были расписаны в предыдущих разделах. С такими сервисами нежелательно вести работу. Перечислим их:

* «горячие» кошельки;
* централизованные биржи;
* бот трейдеры, запущенные не на стороне клиента;
* сервисы облачного майнига;
* майнинг пулы.

Перечислим безопасные для работы сервисы управления криптовалютными активами:

* «холодные» кошельки;
* обменники;
* децентрализованные биржи;
* бот трейдеры, запущенные на стороне клиента.

На рынке на данный момент не существует бот трейдеров, которые были настроены на работу с децентрализованными биржами. Поэтому в данной работе бот трейдеры использоваться не будут.

## Обеспечение условий для безопасного управления криптовалютными активами

Доверенная вычислительная среда – это фрагмент среды электронного взаимодействия, для которого установлена и поддерживается в течение заданного интервала времени целостность объектов и целостность взаимосвязей между ними [16].

В недоверенной среде возможно осуществление множества различных атак: перехват управления, перехват информации на портах ввода и вывода, работа под гипервизором злоумышленников, копирование ключей из оперативной памяти и так далее. К примеру в недоверенной среде возможна атака, в которой злоумышленник осуществляет подмену транзакции при передаче ее на подпись в токен с неизвлекаемыми ключами (для хранения ключей используются токены Trezor и Ledger Nano S). Из-за этого пользователь видит на экране монитора одну информацию, а на подпись отправляется другая, предоставленная злоумышленником.

Поэтому ПАК управления криптовалютными активами должен обеспечивать доверенную среду. Только в этом случае, ключи подписи будут защищены от утраты, а подписываться будут только санкционированные пользователем транзакции.

В качестве средства, обеспечивающего функционирование доверенной среды будем использовать микрокомпьютер на базе платформы МКТ с динамически изменяемой архитектурой, запатентованной под названием «Новая Гарвардская»[17]. МКТ – это устройство, обладающее специальной подсистемой управления доступом к памяти: память устройства, в которой расположена ОС, находится в режиме Read Only (только для чтения), что исключает возможность несанкционированных модификаций критичного программного обеспечения и обеспечивает неизменность доверенной среды функционирования. Кроме того МКТ обладает блоками сеансовой памяти, что дает возможность исполнения программ, для корректной работы которых необходима возможность записи.

Перечисленные выше ключевые особенности МКТ позволяют решить несколько вопросов при работе с криптовалютными активами. К примеру, если в качестве изолированного компьютера в схеме с офлайн подписью транзакций использовать МКТ в Read Only режиме, то можно гарантировать, что даже при попадании вредоносного ПО в систему, оно не сможет исполниться, и таким образом транзакция не будет скомпрометирована.

Подытожим. В ходе обследования были изучены существующие виды сервисов управления криптовалютными активами, среди них были выделены те, с которыми оптимально вести работу с точки зрения безопасности. Также было предложено использовать платформу МКТ для безопасного управления криптовалютными активами в качестве устройства, обеспечивающего доверенную вычислительную среду.

## Аппаратный состав ПАК

### Аппаратная генерация ключей подписи

Ключи подписи представляют наибольшую ценность для злоумышленников. Рассмотрим, как на аппаратном уровне можно обеспечить максимально возможную защиту генерации и хранения ключей.

Протокол BIP32 (Hierarchical Deterministic Wallets) отвечает за генерацию мастер ключа, и генерацию иерархически детерминированных ключей из этого мастер ключа. В официальном репозитории биткойна существует несколько реализаций этого протокола на различных языках программирования [8]. Эти реализации используются в большинстве из существующих криптовалютных кошельков и сервисов. Рассмотрим, какие функции отвечают за генерацию закрытых мастер ключей в наиболее часто используемых реализациях:

* Python
  + - Реализация в PyCoin:  
       private\_key = os.urandom(32).encode('hex')
    - Реализация в BIP32Utils:   
      private\_key = ''.join(['%x' % random.randrange(16) for x in range(0, 64)])
* Java
  + - Реализация bitsofproof:  
      SecretKey key = new SecretKeySpec (chainCode, "HmacSHA512");
* C++
  + - pkey = EC\_KEY\_new\_by\_curve\_name(NID\_secp256k1)  
      Эта функция взята из стандартной библиотеки openssl
* JavaScript
  + - Реализация Bitpay  
      var priv\_key = new ECKey(conv(privateKeyHex, {in: 'hex', out: 'buffer'}), false);

Все эти функции объединяет то, что закрытый мастер ключ генерируется из стандартных библиотек, которые обращаются к генератору псевдослучайных чисел (ГПСЧ). Таким образом, можно повлиять на генерацию мастер ключа (а значит и на генерацию всех дочерних ключей) с помощью атаки на ГПСЧ [18]. Перечислим потенциальные программная атаки на ГПСЧ:

* Прямая криптографическая атака основанная на анализе выходных данных алгоритма.
* Атаки, основанные на входных данных, могут быть разделены на атаки с известными входными данными, атаки с воспроизводимыми входными данными.
* Атаки, основанные на вскрытии внутреннего состояния при которых злоумышленник знает начальное или исходное состояние генератора.
* Закладки в программное обеспечение, при которых создатель алгоритма знает любое из хэшированных псевдослучайных чисел и последующие в цепочке. Такой алгоритм сложно определить со стороны, так как числа выглядят равномерно распределенными по всему диапазону.

Этих атак можно избежать, если вместо псевдослучайных чисел использовать случайные числа, которые можно генерировать с помощью аппаратного датчика, выступающего в роли физического генератора случайных чисел. Таким образом, чтобы минимизировать риски компрометации ключей, в ПАК для генерации случайных чисел целесообразно использовать ФДСЧ.

### Выбор способа реализации криптоядра

Если мы хотим сократить риски, контакт с сетью нужно минимизировать. Однако в случае, если пользователь хочет торговать на бирже, то возникает необходимость длительного подключения к интернету. Из-за этого возрастает риск компрометации ключей. При биржевой торговле часто роль играют лишние минуты и даже секунды при совершении сделок, поэтому использовать для высокочастотной торговли схему с офлайн подписью транзакций не целесообразно.

Для минимизации риска компрометации ключей подписи целесообразно вынести функционал генерации и хранения ключей (а также все криптографические функции) в отдельный аппаратный модуль - криптоядро. Ключи в этом модуле должны быть неизвлекаемыми, и все операции (шифрование, хеширование, подпись данных) выполняются также внутри модуля.

Имплементацию криптоядра в ПАК можно сделать двумя способами: либо модифицировать архитектуру МКТ, интегрировав в нее криптоядро, либо использовать отчуждаемое криптоядро, к примеру, в качестве такого модуля в аппаратном отчуждаемом кошельке Ledger Nano S используется микроконтроллер ST31H320, соответствующий банковским стандартам безопасности согласно общим критериям уровня EAL6+[19].

Использование отчуждаемого носителя сопряжено со следующими недостатками:

* оно делает возможным сценарии атак с подменой или иными вариантами компрометации ключей за счет отчуждаемости носителя [20];
* подключенние к порту USB-устройства подразумевает проведение процедуры взаимной аутентификации между отчуждаемым устройством и компьютером, что является трудоемкой и времязатратной задачей, которая значительно усложняет систему;

Вместо внешнего отчуждаемого криптоядра целесообразно использовать внутреннее встроенное криптоядро - аппаратный модуль, имплементированный в архитектуру ПАКа и при этом автономный от него, то есть резидентный компонент безопасности (РКБ).

Так как РКБ должен быть максимально автономным [20], он должен иметь как можно меньше точек соприкосновения с вычислительной системой. В идеале он должен взаимодействовать с системой только по одному каналу обмена информацией (к примеру по USB шине). Система может обратиться к РКБ только предоставляемому набору API функций, который должна быть построен таким образом, что система вызовов API исключает возможность целенаправленного или случайного извлечения ключей подписи программным способом (как именно построить такую систему будет рассмотрено ниже).

Таким образом, ПАК, включающий в себя МКТ с имплементированным РКБ, является безопасной платформой, на которой можно реализовать АРМ для управления криптовалютными активами.

### Выбор аппаратной платформы

Необходимо выбрать аппаратное устройство, которое будет выполнять роль РКБ. Перечислим требования, которым должен удовлетворять аппаратный модуль [20]:

* Модуль должен быть узко специализированным, примитивным вычислительным устройством с собственной операционной средой и памятью, так как для достижения высокого уровня защищенности модуль должен иметь меньше уязвимостей по сравнению со средой, в которую он будет встраиваться.
* Модуль предназначен для использования в качестве подсистемы, встраиваемой в защищаемую более сложную систему, поэтому он должен одновременно сочетать в себе необходимую производительность, при малой потребляемой мощности и низкой стоимости интеграции в систему.
* Модуль должен обладать датчиком, который будет выступать в роли физического источника случайных чисел.
* Модуль должен обладать блоками памяти для хранения ключевых пар.
* Модуль должен быть программируемым.

В качестве вычислительного устройства для РКБ можно использовать либо микроконтроллер, либо микропроцессор. Для того, чтобы понять, что лучше подходит для решения поставленной задачи, проведем сравнение в таблице 3.

Таблица 3. Сравнение микроконтроллера и микропроцессора

|  | Микропроцессор | Микроконтроллер |
| --- | --- | --- |
| Внутреннее устройство | Содержит центральный процессор, регистры общего назначения, указатели стека, счетчики программы, таймер и цепи прерываний | Cодержит микропроцессор и имеет встроенные ПЗУ, ОЗУ, устройства ввода/вывода, таймеры и счетчики. |
| Запоминающее устройство | Основано на архитектуре фон Неймана. Программа и данные хранятся в одном и том же модуле памяти. | Основано на Гарвардской архитектуре. Программы и данные хранятся в разных модулях памяти. |
| Память данных | Имеет много инструкций для перемещения данных между памятью и процессором. | Имеет одну-две инструкции для перемещения данных между памятью и процессором. |
| Число регистров | Имеет меньшее количество регистров, операции в основном производятся в памяти. | Имеет большее число регистров, поэтому проще писать программы |
| Время доступа | Время доступа к памяти и устройствам ввода/вывода больше. | Меньшее время доступа для встроенной памяти и устройств ввода/вывода. |
| Где используется | В качестве управляющего элемента в компьютерных системах | В встраиваемых системах |
| Затраты на интеграцию | Высокие затраты на интеграцию в систему | Низкие затраты на интеграцию в систему |

Исходя из требований, описанных выше, под решение поставленной задачи больше подходит использование микроконтроллера.

Итак, ПАК управления криптовалютными активами должен быть сделан на базе платформы МКТ и обладать встроенным микроконтроллером. Возникает выбор:

* разработать с нуля устройство, архитектура которого основывается на базе МКТ и в которую будет встроен модуль в виде микроконтроллера, который будет выполнять роль РКБ;
* модифицировать архитектуру какого-либо существующего устройства на базе МКТ таким образом, чтобы в него можно было встроить модуль в виде микроконтроллера, который будет выполнять роль РКБ;
* использовать готовое решение на баз архитектуры МКТ со встроенным модулем в виде микроконтроллера, который будет выполнять роль РКБ.

Разработка архитектуры микрокомпьютера является трудозатратным, времязатратным и дорогим процессом, так как подразумевает большой объем работы при проектировании, разработке, тестировании и отладке устройства и его ПО. Модификация существующей архитектуры равносильна разработке новой архитектуре.

Поэтому целесообразно использовать уже готовое решение из существующей линейки компьютеров семейства МКТ.

На рынке существует устройство m-TrusT, основанное на архитектуре МКТ и обладающее встроенным модулем в виде микроконтроллера, предназначенным для выполнения функций безопасности [21]. Микрокомпьютер m-TrusT изначально проектировался для защищенной сетевой коммуникации между элементами критической информационной инфраструктуры (КИИ).

Это одноплатный микрокомпьютер на базе новой гарвардской архитектурой, с имплементированным микроконтроллером at91sam3u на базе процессора ARM Cortex-M3. Также устройство обладает физическим датчиком случайных чисел (ФДСЧ) в виде двухплечевого решения с использованием шумовых диодов 2Г103А9 по схеме «Дебют».

Микрокомпьютер m-TrusT включает в себя док-станцию, которая стационарно включается в состав элемента КИИ, и подключаемого к ней универсального по своему аппаратному исполнению модуля – мезонина. В зависимости от того, что какие элементы КИИ должны взаимодействовать ПО активной части m-TrusT, «мезонина» – будет разным. Оно создается для каждой конкретной задачи. Такая универсальность, заложенная на уровне архитектуры, позволяет относительно легко настроить устройство под задачи данной работы. Требуется лишь написать собственное микропрограммное обеспечение, «прошивку», для устройства и для модуля безопасности.

Архитектура платформы позволяет осуществлять полную доверенную загрузку, при которой полностью контролируется (с пошаговым обеспечением целостности) загрузка не только ОС, но и начальная загрузка устройства. РКБ контролирует процесс такой такой доверенной загрузки. Процессор устройства сконфигурирован таким образом, чтобы старт загрузки всегда производился из памяти, физически переведенной в режим «только для чтения». Из этой памяти стартует загрузчик, который проверяет целостность модуля, который, после запуска производит настройку СДЗ, который контролирует запуск ОС. Для повышения уровня защищенности в ПАК целесообразно использовать данные механизмы контроля целостности и доверенной загрузки.

Подытожим. В данной работе в качестве платформы для ПАК будем использовать микрокомпьютер m-TrusT с имплементированным микроконтроллером at91sam3u, который будет выполнять роль РКБ.

## Функциональность ПАК

### Функциональность РКБ

РКБ должен поддерживать определенный набор функций (криптографические, криптовалютные и так далее), а также систему API, позволяющую внешним приложениям вызывать часть функций из этого набора, разрешенных для вызова и не нарушающих безопасности ключей. Остальные функции должны быть доступны только для внутреннего вызова.

Пользователю, и как следствие пользовательским приложениям, для управления криптовалютными активами должны быть доступны только те функции, которые нужны для реализации верхнеуровневых процессов бизнесс логики. К этим процессам относятся:

* создание криптовалютной пары ключей и доступ к ней (генерация и хранение ключевой пары);
* совершение и получение переводов (подпись транзакций);
* удаление использованных ключей подписи из памяти;
* проверка состояния баланса аккаунтов (вызов списка ключей проверки подписи).

Иерархически детерминированные кошельки поддерживают создание мнемонической фразы из мастер ключа по протоколу BIP 39 [22]. Эта фраза нужна для восстановления мастер ключа и всех дочерних ключей в случае утраты устройства или удаления ключа из постоянной памяти. По фразе можно однозначно восстановить мастер ключ, поэтому компрометация фразы равняется компрометации мастер ключа. К at91sam3u, имплементированному в m-trust, аппаратно нет возможности подключить визуализирующую периферию (к примеру LCD монитор). Поэтому если возникает необходимость вывести полученную мнемоническую фразу на монитор для пользователя, то ее придется экспортировать из РКБ, что равняется экспорту закрытого мастер ключа. Это противоречит основной логике неизвлекаемости ключей подписи. Поэтому в данной работе нецелесообразно использовать мнемонические фразы из протокола BIP 39.

Выше уже было сказано, что РКБ должен, помимо выполнения криптовалютных операций, обеспечивать доверенную загрузку и контроль целостности. Перечень необходимых для этого функций был составлен в [20].

### Функциональность, необходимая для холодного хранения

АРМ на основе ПАК управления криптовалютными активами должен обеспечивать возможность выполнения определенных функций, необходимых для хранения криптовалюты, а также для совершения транзакций. Эти транзакции могут совершаться редко, к примеру, раз в день, или часто, к примеру, несколько раз в минуту в случае высокочастотной биржевой торговли.

В первом случае постоянное соединение с интернетом не требуется, поэтому ПАК можно использовать в качестве холодного хранилища. В [15] было показано, что для холодного хранения на платформе МКТ целесообразно применять схему с офлайн подписью транзакций. В этой схеме все операции, связанные с закрытыми ключами, выполняются в защищенном и изолированном компьютере (в данном случае в m-TrusT), а все остальные операции могут выполняться на произвольном компьютере. Распишем в таблице 4 функции необходимые для реализации этой схемы.

Таблица 4. Функциональность АРМ в схеме холодного хранения

| **Название функции** | **Описание** | **Среда исполнения** |
| --- | --- | --- |
| Генерация ключей, мастер-ключа | Генератор псевдослучайных чисел выдает на выходе, в зависимости от выбранного режима, либо ключевую пару, либо seed, из которого дальше по протоколу BIP32 будут развернуты остальные ключевые пары | m-TrusT |
| Развертывание ключей из мастер ключа | Пользователь вводит ключ, а кошелек на основе введенных данных выдает список всех ключевых пар пользователя, после чего ими можно пользоваться | m-TrusT |
| Создание нового аккаунта в кошельке | После генерации и развертывания ключей, на выбор пользователя будет создан файл аккаунта в кошельке, в этом файле будет записано, что за данным пользователем закреплены соответствующие открытые ключи, и этот же файл будет хранить историю операций | m-TrusT |
| Чтение баланса из скачанного блокчейна | Идет чтение из блокчейна, расположенного в постоянной памяти. По завершению операции на выходе получаем баланс адреса на момент последнего обновления блокчейна | Произвольный компьютер |
| Скачивание обновление блокчейна | Обновление блокчейна, для последующего совершения транзакции или чтения баланса аккаунта | Произвольный компьютер |
| Чтение баланса из блокчейна в интернете | Чтобы не скачивать гигабайты данных, идет чтение из блокчейна, расположенного в интернете, то есть идет обращение к серверам или соседним узлам. По завершению операции на выходе получаем баланс аккаунта на момент последнего обновления блокчейна | Произвольный компьютер |
| Создание транзакции | Пользователь вводит адрес отправителя, адрес получателя и количество валюты, которую нужно перевести. В случае если сумма переводится не полностью, то необходимо добавить адрес для возвращения сдачи. Также в транзакцию необходимо добавить ряд служебной информации, включая адреса и хэши транзакций, с которых были получены средства для корректной работы этого этапа необходимо обновить блокчейн до актуального состояния. На выходе получаем сформированную, но не подписанную транзакцию | Произвольный компьютер |
| Подпись транзакции через интерфейс кошелька | Чтобы подписать транзакцию, пользователю необходимо ввести ключ подписи в соответствующее поле (или идет чтение из файла, в котором в зашифрованном виде хранятся закрытые ключи), после чего система произведет все необходимые операции и вычисления | m-TrusT |
| Трансляция транзакции в блокчейн-сеть | Подписанная транзакция посылается в Сеть, на ближайший доступный узел | Произвольный компьютер |

Изобразим на рисунке 5 алгоритм совершения транзакций в схеме холодного хранения.

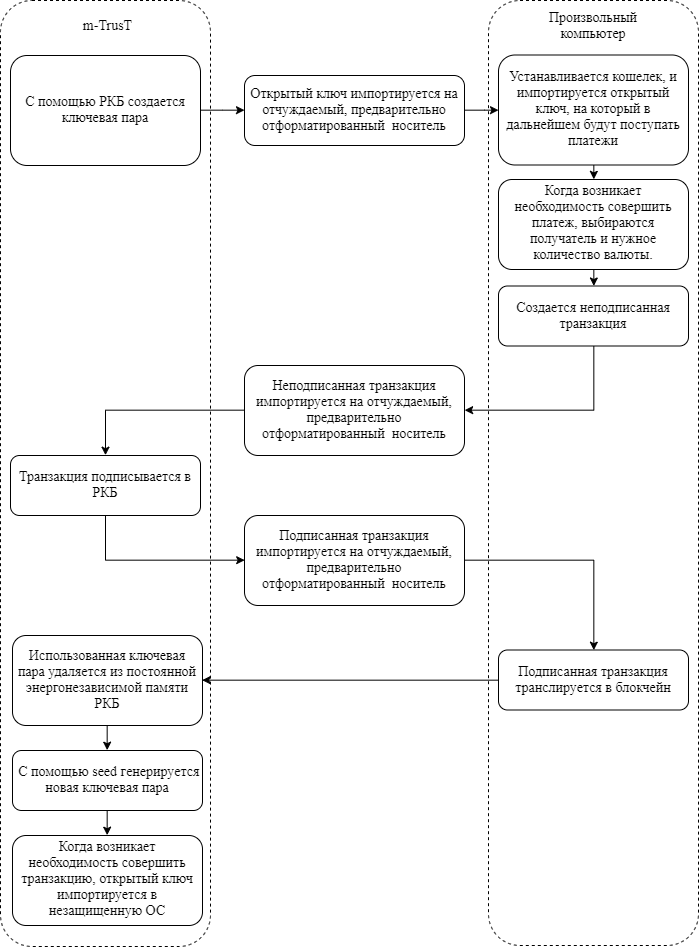


Рисунок 5. Алгоритм совершения транзакций в схеме холодного хранения

### Функциональность, необходимая для горячего хранения

При совершении сделок на бирже роль играют лишние минуты и даже секунды, поэтому использовать для высокочастотной торговли схему с офлайн подписью транзакций не целесообразно. Поэтому, все операции необходимо будет выполнять в пределах одного устройства, и при этом возникает необходимость длительного подключения к интернету, из-за чего возрастает риск компрометации ключей. Если пользователь не готов принять этот риск, то ему следует выбрать вариант исключительно холодного хранения с офлайн подписью транзакций. Если пользователь готов к такому риску, то для совершения операций ему придется использовать только m-TrusT.

DEX биржи позволяют использовать самостоятельно сгенерированные ключевые пары, а также импортировать/экспортировать транзакции. В этом они похожи на работу с обычными криптовалютными кошельками, поэтому необходимая функциональность для совершения транзакций остается такой же, как и в таблице 5. Остальные экономические функции (анализ котировок, сравнение потенциальной доходности и так далее) - зависят от используемого ПО и при этом не задействуют ключевые пары.

Алгоритм совершения транзакций в схеме горячего хранения принимает следующий вид (рисунок 6).

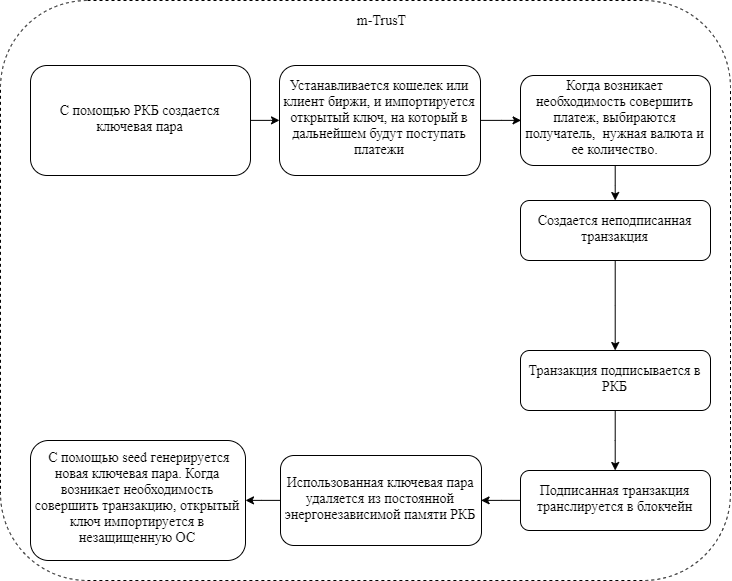


Рисунок 6. Алгоритм совершения транзакций в схеме горячего хранения

## Программный состав ПАК

### ПО для обеспечения необходимой функциональности РКБ

На рынке уже существуют готовые программные решения, которые обеспечивают функциональность доверенной загрузки и контроля целостности, к примеру «Аккорд-МКТ».

Микропрограммного обеспечения, которое бы выполняло криптовалютные операции, на рынке в открытом доступе не существует. Поэтому необходимо такое ПО разработать. Данное ПО должно стать расширением набора функций уже существующего ПО, осуществляющего доверенную загрузку.

Данная задача легла в основу Технического задания на данный дипломный проект (Приложение №1)

### Перечень пользовательских приложений, необходимых для обеспечения требуемой функциональности

Формирование состава ПО начнем с выбора операционной системы. Будем использовать Linux, так как данное семейство ОС распространяется по свободной лицензии, и предоставляет возможность более гибкой настройки системы на всех уровнях. В данной работе будем использовать Linux Ubuntu 16.04, как одну из наиболее популярных и стабильных версий.

Выше было показано, что для управления криптовалютными активами необходимо следующее ПО: криптовалютные кошельки и клиенты децентрализованных бирж (DEX бирж).

Приложения должны удовлетворять следующим требованиям:

1. Приложение устанавливается на компьютер пользователя.
2. При совершении всех операции есть возможность использовать самостоятельно сгенерированную ключевую пару.
3. Ключи подписи не передаются сторонним серверам.
4. Поддерживается возможность импорта и экспорта транзакций.

Многообразие такого ПО весьма велико, поэтому нет смысла перечислять его полностью. Однако итоговый состав ПО ПАКа управления криптовалютными активами должен состоять из ограниченного числа приложений, которые должны не только удовлетворять вышеперечисленным критериям, но и в идеале должны быть проверены на отсутствие логических и программных уязвимостей. На эту роль подойдут наиболее популярные приложения, которые уже давно существуют на рынке и успешно прошли многократные проверки криптоаналитиков.

В данной работе ограничим список ПО управления криптовалютными активами следующим образом: оставим официальные кошельки для двух наиболее популярных и дорогих криптовалют (Bitcoin и Etherium), один мультивалютный кошелек, одна DEX биржа, требующая установки клиента. Браузер также будет являться средством управления криптовалютными активами, так пользователю может быть необходимо воспользоваться криптовалютным обменником или DEX биржей, которую можно запускать как с помощью клиента, так и с помощью браузера. Выберем для этих целей два наиболее распространенных и популярных браузера, две наиболее популярных DEX биржи и два наиболее популярных обменника (Netex24 и Kassa).

Составим перечень ПО, необходимого для управления криптовалютными активами:

* криптовалютные кошельки
  + - для Bitcoin: Electrum, Bitcoin Core;
    - для Etherium: Etherium Wallet (Mist);
    - для остальных криптовалют (мультивалютный): Exodus;
* децентрализованная (DEX) биржа, устанавливаемая на компьютер: Waves DEX;
* децентрализованные (DEX) биржи, запускаемые через браузер: IDEX, EtherDelta.
* Браузеры: Mozilla Firefox, Google Chrome.

### Обеспечение корректного взаимодействия ПО с РКБ

Необходимо добиться того, чтобы каждое приложение из перечня разрешенного ПО совершало криптографические операций с криптовалютными ключами с помощью РКБ, а не стандартными средствами, заложенными разработчиками в код приложения.

Переписывать и модифицировать каждую программу нецелесообразно. Следовательно надо сделать так, чтобы все существующие приложения обращались к РКБ без вмешательства во внутреннюю логику каждой программы. Это можно сделать с помощью вмешательства внешнего, с помощью перехвата функций. Нужно таким образом настроить вычислительную среду, чтобы при попытке приложения совершить криптографическую операцию, код операции исполнял РКБ, а не стандартные средства системы, то есть необходимо осуществлять перехват вызовов функций.

Когда пользователь будет работать с криптовалютными активами, он может использовать приложение (или несколько приложений одновременно) в произвольные промежутки времени. Поэтому целесообразно использовать программу, которая постоянно будет следить за системой и осуществлять перехват вызовов функций. Для этих целей подойдет демон — компьютерная программа в системах класса UNIX, запускаемая самой системой во время загрузки и работающая в фоновом режиме.

## Перечень задач, подлежащих решению в процессе разработки

Задача разработки ПАК управления криптовалютными активами свелась к подзадачам разработки недостающего ПО, состоящего из:

* микропрограммного обеспечения для микроконтроллера at91sam3u (расширение функционала уже существующего ПО), которое будет выполнять всю работу с криптовалютными ключами;
* демона перехватчика для Linux, который будет осуществлять перехват вызовов криптографических функций. В рамках данной работы, с учетом сроков и сложности разработки, Заказчиком было принято решение, что данный пункт будет выполнен другой командой разработчиков.

В ходе дальнейшей разработки необходимо решить следующие подзадачи:

* Конкретизировать функциональность ПО РКБ, определить состав входных и выходных данных этих функций.
* Выбрать оптимальное место для хранения криптографических ключей.
* Составить полный перечень требований к ПО.

Перечисленные выше задачи легли в основу технического задания на данный дипломный проект (Приложение №1).

# КОНСТРУКТОРСКАЯ ЧАСТЬ

## Реализация функциональности РКБ

Перечислим и опишем все функции РКБ в таблице 5.

Таблица 5. Функциональность РКБ

| Функция | Описание функции | Входные параметры  /\*TO DO\*/ | Выходные данные  /\*TO DO\*/ |
| --- | --- | --- | --- |
| **Подсистема обеспечения доверенной загрузки** | | | |
| Переход в рабочий режим | Команда подается ПО СДЗ после успешного завершения процедур самотестирования начальным кодом СДЗ. По этой команде открывается раздел РКБ с базой данных пользователей на чтение и раздел с журналами на дополнение. |  |  |
| Вход администратора | Команда подается ПО СДЗ при необходимости доступа администратора к ресурсам РКБ. По этой команде раздел с базой данных и журналами становятся доступны на чтение и запись. |  |  |
| Переход в закрытый режим | Команда подается ПО СДЗ после завершения работы СДЗ. По этой команде все разделы РКБ становятся недоступны. |  |  |
| **Подсистема работы с неизвлекаемым ключом подписи** | | | |
| Генерация неизвлекаемого ключа подписи | Команда подается прикладным ПО. По этой команде на основе случайной последовательности байт, полученных с ФДСЧ, firmware РКБ создается ключ подписи и записывается в постоянную энергонезависимую память РКБ. |  |  |
| Получение ключа проверки подписи | Команда подается прикладным ПО. По этой команде вырабатывается ключ проверки подписи, соответствующий неизвлекаемому ключу подписи, и передается в процессор. |  |  |
| Выработка сессионного ключа | Команда подается прикладным ПО. По этой команде вырабатывается сессионный ключ на основе случайной последовательности байт, полученных с ФДСЧ и передается в процессор. Кроме того, по этой команде вырабатывается ключ защиты ключей на основе неизвлекаемого ключа подписи и переданного ключа проверки подписи, на котором зашифровывается сессионный ключ, и зашифрованный сессионный ключ также передается в процессор. |  |  |
| Расшифрование сессионного ключа | Команда подается прикладным ПО. По этой команде полученный зашифрованный сессионный ключ расшифровывается на ключе защиты ключей, который в свою очередь вырабатывается на основе неизвлекаемого ключа подписи и переданного ключа проверки подписи. |  |  |
| **Подсистема работы с ФДСЧ** | | | |
| Генерация последовательности случайных чисел | По этой команде происходит получение случайных чисел с шумовых диодов, их обработка, контроль качества и передача в процессор. Функция закрытая, то есть доступна только для вызова функциями firmware РКБ. | Требуемое количество байт | Массив байт с ФДСЧ |
| **Подсистема работы с криптовалютами** | | | |
| Генерация неизвлекаемого мастер ключа | По этой команде происходит генерация случайного числа с помощью ФДСЧ и его дальнейшая обработка с помощью хэш-функции HMAC-SHA512 по протоколу BIP 32. Созданный ключ записывается в постоянную энергонезависимую память РКБ. Функция доступна для вызова по API извне РКБ. | Имя пользователя | На выход подается код, обозначающий сообщение об успешно завершенной операции |
| Генерация дочерних ключевых пар требуемой криптовалюты | По этой команде происходит развертывание иерархически детерминированных мультивалютных ключей из мастер ключа по протоколам BIP 32, BIP 43 [23] и BIP 44 [24]. Созданные ключи записываются в постоянную энергонезависимую память РКБ. Функция доступна для вызова по API извне РКБ. | Имя пользователя, требуемая криптовалюта, требуемое количество дочерних ключевых пар | На выход подается код, обозначающий сообщение об успешно завершенной операции |
| Развертывание дерева для начала работы | По этой команде происходит развертывание полного дерева используемых ключевых пар из мастер ключа | Имя пользователя | На выход подается код, обозначающий сообщение об успешно завершенной операции |
| Подпись криптовалютной транзакции | По этой команде происходит подпись транзакции по алгоритму ECDSA на необходимом дочернем ключе подписи. Функция доступна для вызова по API извне РКБ. | Неподписанная транзакция, имя пользователя,  Ключ проверки подписи | Подписанная транзакция |
| Получение списка дочерних ключей проверки подписи | По этой команде вырабатываются дочерние ключи проверки подписи, соответствующие созданным дочерним ключам подписи, и передаются в процессор. Функция доступна для вызова по API извне РКБ. | Отсутствуют | Массив ключей проверки подписи |
| Удаление неизвлекаемого мастер ключа | Команда подается прикладным ПО. По этой команде происходит удаление неизвлекаемого мастер ключа из постоянной энергонезависимой памяти РКБ. Функция доступна для вызова по API извне РКБ. | Имя пользователя ОС | Данные об успешном или неуспешно завершении операции |
| **Подсистема журналирования и аудита** | | | |
| Генерация записей аудита | По этой команде данные о произошедшем событии добавляются в постоянную энергонезависимую память РКБ. Функция закрытая, то есть доступна только для вызова функциями firmware РКБ. | Дата и время события,  тип события,  результат события,  Имя пользователя | На выход подается код, обозначающий сообщение об успешно завершенной операции |
| Просмотр журнала событий аудита. | По этой команде осуществляется чтение из постоянной энергонезависимой памяти РКБ массива данных с логами, и эти данные передаются в процессор микрокомпьютера. Функция доступна для вызова по API извне РКБ. | Отсутствуют | Журнал событий аудита |

## Реализация демона для перехвата вызовов функций

Демона перехватчика необходимо разработать. Для этого необходимо понять что именно нужно перехватывать и как это делать.

### Вызов криптографических функций при работе с ключами

Рассмотрим, какие библиотеки используют программы для вызова криптографических функций при работе с ключами. Для этого изучим исходники приложений на GitHub:

* В Electrum [25] и IDEX [26] используются следующие библиотеки Python: Hashlib, ssl, pyaes, hmac, ecdsa.
* В Bitcoin Core [27] используются следующие библиотеки C++: openssl, crypto.
* В Etherium Wallet (Mist) [28], Exodus[29], Waves DEX [30], EtherDelta [31] используются следующие библиотеки JavaScript: CryptoJS, pidCrypt

Необходимо добиться того, чтобы ОС осуществляла перехват обращений к этим стандартным библиотекам и перенаправляла их на РКБ.

### Выбор метода перехвата и переадресации вызова функций

Перечислим существующие методы перехвата и переадресации функций в среде Linux и затем выберем среди них наиболее оптимальный [32].

**Linux Security API**

Linux Security API — это специальный интерфейс для обращения к тем местам кода ядра ОС, в которых расположены вызовы функций безопасности, которые в свою очередь вызывают callback функции, установленные модулем безопасности. Этот модуль может изучать контекст операции и принимать решение о её разрешении или запрете.

У Linux Security API есть пара важных ограничений. Во-первых, модули безопасности не могут быть загружены динамически, так как они являются частью ядра и требуют его пересборки. Это сделано для того, чтобы обеспечивать безопасность постоянно, с момента загрузки. Во-вторых, в системе может быть только один модуль безопасности.

Таким образом, для использования Security API необходимо собирать собственную сборку ядра, а также интегрировать дополнительный модуль с системой принудительного контроля доступа. Это сложный вариант, сопряженный с большими затратами сил на разработку.

**Модификация таблицы системных вызовов**

Перехват, требуемый для действий, выполняемых пользовательскими приложениями, может быть реализован на уровне системных вызовов. Linux хранит все обработчики системных вызовов в таблице sys\_call\_table. Подмена значений в этой таблице приводит к смене поведения всей системы. Таким образом, сохранив старое значения обработчика и подставив в таблицу собственный обработчик, возможно перехватить любой системный вызов.

У этого подхода есть ряд преимуществ и недостатков. Преимущества:

* Полный контроль над любыми системными вызовами — единственным интерфейсом к ядру у пользовательских приложений.
* Малая ресурсоемкость. Помимо нагрузки мониторинга, единственным лишними действиями является дополнительный вызов функции (для вызова оригинального обработчика системного вызова).
* Минимальные требования к ядру. Этот подход не требует каких-либо дополнительных конфигурационных опций в ядре, поэтому реализуем на широком перечне систем.

Недостатки:

* Техническая сложность реализации. Сама по себе замена указателей в таблице не является сложной, но сопутствующие задачи требуют дополнительных усилий при реализации, тестировании, отладке и дальнейшей поддержке.
* Невозможность перехвата некоторых обработчиков. Некоторые из функция ядра требуют того, что обработчик системного вызова должен являться специальным переходником, реализованным на ассемблере. Подобные обработчики сложно, а иногда невозможно заменить на свои, написанные на Си.
* Перехватываются только системные вызовы. Этот подход позволяет заменять обработчики системных вызовов, что ограничивает точки входа только ими. Все дополнительные проверки выполняются либо в начале, либо в конце, при этом известны лишь аргументы системного вызова и его возвращаемое значение. Иногда это приводит к необходимости дублировать проверки аргументов и проверки доступа, иногда - вызывает лишние накладные расходы, когда требуется дважды копировать память пользовательского процесса.

Из-за перечисленных выше недостатков в данной работе нецелесообразно использовать этот вариант перехвата.

**Kprobes**

Kprobes - это специализированное API, предназначенное для отладки и трассирования ядра. Этот интерфейс позволяет устанавливать пред- и постобработчики для любой инструкции в ядре, а также обработчики на вход и возврат из функции. Обработчики получают доступ к регистрам и могут их изменять. Таким образом, можно получить как мониторинг, так и возможность влиять на дальнейший ход работы.

Преимущества kprobes:

* Kprobes обладает проработанной и отлаженной системой API, хорошо задокументированным интерфейсом.
* Перехват любого места в ядре. Kprobes реализуется с помощью точек останова (инструкции int3), внедряемых в исполнимый код ядра. Это позволяет устанавливать kprobes в любом месте любой функции, если оно известно. Аналогично, kretprobes реализуются через подмену адреса возврата на стеке и позволяют перехватить возврат из любой функции (за исключением тех, которые управление в принципе не возвращают).

Недостатки kprobes:

* Техническая сложность. Kprobes — это только способ установить точку останова в любом места ядра. Для получения аргументов функции или значений локальных переменных надо знать, в каких регистрах или где на стеке они лежат, и самостоятельно их оттуда извлекать. Для блокировки вызова функции необходимо вручную модифицировать состояние процесса так, чтобы процессор подумал, что он уже вернул управление из функции.
* Ограничения kprobes. Kprobes реализуется через подмену адреса возврата на стеке. Соответственно, необходимо где-то хранить оригинальный адрес, чтобы вернуться туда после обработки kprobes. Адреса хранятся в буфере фиксированного размера. В случае его переполнения, когда в системе выполняется слишком много одновременных вызовов перехваченной функции, kprobes будет пропускать срабатывания.
* Отключенное вытеснение. Для синхронизации все обработчики выполняются с отключенным вытеснением (preemption). Это накладывает определённые ограничения на обработчики: в них нельзя ждать — выделять много памяти, заниматься вводом-выводом, спать в таймерах и семафорах, и так далее.

Из-за перечисленных выше недостатков в данной работе нецелесообразно использовать этот вариант перехвата.

**Сплайсинг**

Сплайсинг - это способ перехвата функций, заключающийся в замене инструкций в начале функции на безусловный переход, ведущий в нужный обработчик. Оригинальные инструкции переносятся в другое место и исполняются перед переходом обратно в перехваченную функцию. С помощью двух переходов вшивается (splice in) дополнительный код в функцию, поэтому такой подход называется сплайсингом.

Используя сплайсинг можно добиться тех же результатов, что и при применении kprobes, но без дополнительных расходов и с полным контролем ситуации.

Преимущества сплайсинга:

* Минимальные требования к ядру. Сплайсинг не требует особенных опций в ядре и работает в начале любой функции. Нужно только знать её адрес.
* Минимальные вычислительные расходы. Два безусловных перехода — это все действия, которые надо выполнить перехваченному коду, чтобы передать управление обработчику и обратно.

Однако, этот подход является технически сложным. При переписывании машинного кода придётся решить следующие проблемы:

* синхронизация установки и снятия перехвата, к примеру, на случай, если функцию вызовут прямо в процессе замены её инструкций;
* обход защиты на модификацию регионов памяти с кодом;
* инвалидация кешей процессора после замены инструкций;
* дизассемблирование заменяемых инструкций, чтобы скопировать их целыми;
* проверка на отсутствие переходов внутрь заменяемого куска кода;
* проверка на возможность переместить заменяемый кусок кода в другое место.

**Ftrace**

Ftrace [33] — это фреймворк для трассирования ядра на уровне функций, обладающий интерфейсом для пользовательских программ. Ftrace позволяет отслеживать частоту и длительность вызовов функций, отображать графы вызовов, фильтровать интересующие функции по шаблонам.

Реализуется ftrace на основе ключей компилятора -pg и -mfentry, которые вставляют в начало каждой функции вызов специальной трассировочной функции mcount() или \_\_fentry\_\_(). Вызывать ftrace из каждой функции — это слишком ресурсоемко, поэтому для популярных архитектур доступна оптимизация в виде динамического ftrace, при использовании которого ядро знает расположение всех вызовов mcount() или \_\_fentry\_\_() и на ранних этапах загрузки заменяет их машинный код на nop — специальную ничего не делающую инструкцию. При включении трассирования в нужные функции вызовы ftrace добавляются обратно. Таким образом, если ftrace не используется, то его влияние на систему минимально.

Для перехвата функции требуется минимум кода: необходимо знать лишь имя перехватываемой функции, адрес функции-обёртки, которая будет вызываться вместо перехваченной функции, и указатель на место, куда следует записать адрес перехватываемой функции.

Рассмотрим преимущества ftrace:

* Проработанная система API и простой код. Вся установка перехвата требует пары вызовов функций, заполнение двух полей в структуре.
* Перехват любой функции по имени.
* Перехват совместим с трассировкой, то есть с ядра можно снимать полезные показатели производительности.

Рассмотрим недостатки ftrace:

* Требования к конфигурации ядра. Для успешного выполнения перехвата функций с помощью ftrace ядро должно предоставлять целый ряд возможностей:
  + - список символов kallsyms для поиска функций по имени
    - фреймворк ftrace в целом для выполнения трассировки
    - опции ftrace, критически важные для перехвата
* Функции оборачиваются целиком, так же как и при сплайсинге. Однако, если сплайсинг технически возможно выполнить в любом месте функции, то ftrace срабатывает исключительно при входе.

Недостатки, связанные с требованиям к конфигурации ядра, решаемы на этапе установки. Недостатки с полным оборачиванием функций усложняет процесс разработки, но критически не повышает сложность. Поэтому в данной работе для перехвата функций будем использовать ftrace, так как это простой в реализации и при этом эффективный способ перехвата любых функций в ядре Linux.

### Полный процесс перехвата функции

На рисунке 7 рассмотрим детально схему работы процесса перехвата на примере перехвата системного вызова  execve(), вызванного пользователем.

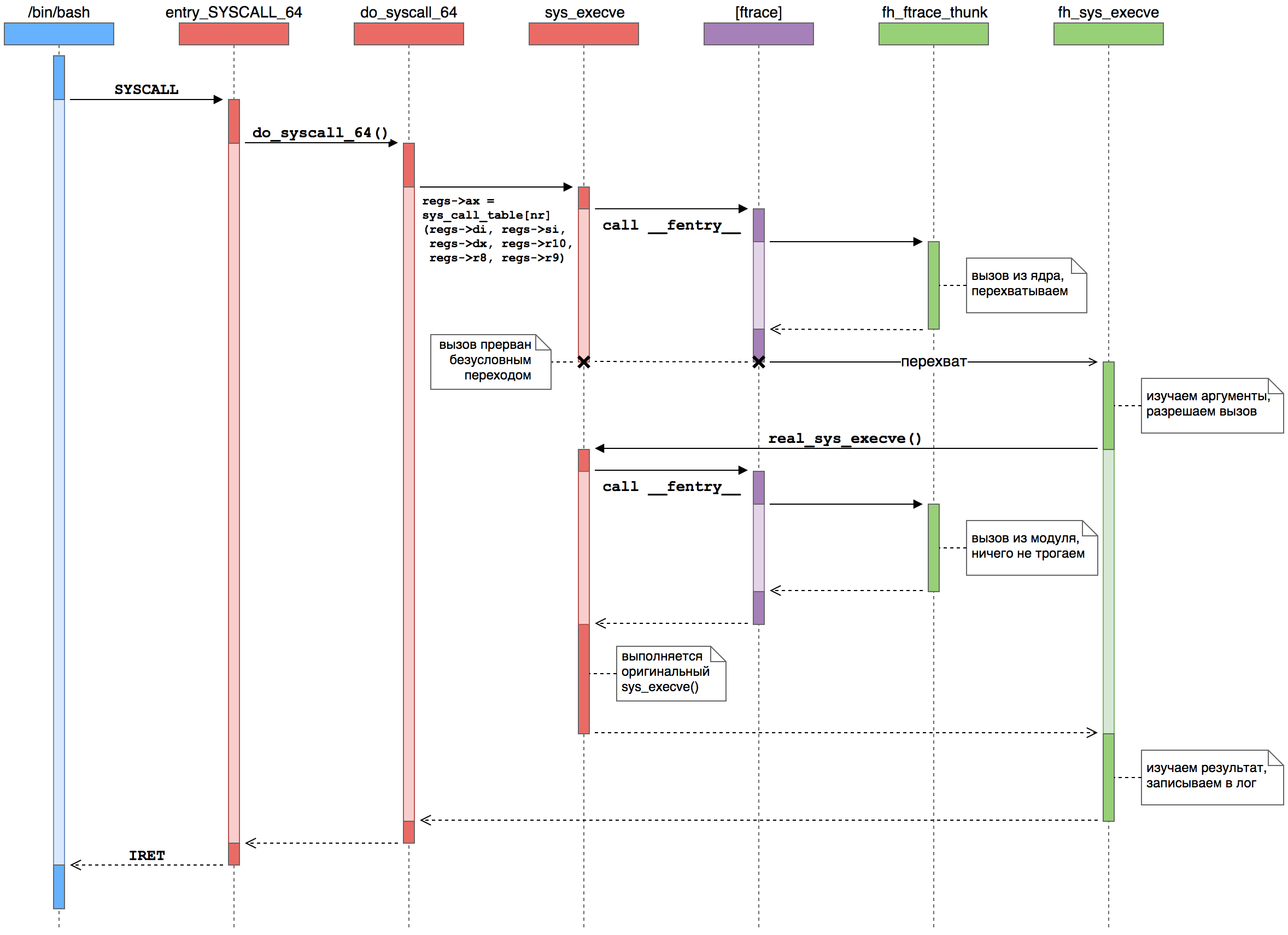


Рисунок 7. Перехват функции execve() с помощью фреймворка ftrace

Пользовательский процесс (голубой) выполняет системный вызов в ядро (красное), где фреймворк ftrace (фиолетовый) вызывает функции из подставляемого модуля (зелёного).

1. Пользовательский процесс с помощью SYSCALL выполняет переход в режим ядра, и управление передаётся низкоуровневому обработчику системных вызовов — entry\_SYSCALL\_64(). Он отвечает за все системные вызовы 64-битных программ на 64-битных ядрах.
2. Управление переходит к конкретному обработчику. Ядро выполняет низкоуровневый код, [реализованный на ассемблере](https://elixir.bootlin.com/linux/v4.16/source/arch/x86/entry/entry_64.S" \l "L206), и передаёт управление высокоуровневой функции do\_syscall\_64(), [написанной на Си](https://elixir.bootlin.com/linux/v4.16/source/arch/x86/entry/common.c" \l "L269). Эта функция в свою очередь обращается к таблице обработчиков системных вызовов sys\_call\_table и вызывает оттуда конкретный обработчик по номеру системного вызова — в данном случае это будет функция sys\_execve().
3. Вызывается фреймворк ftrace, который реализует функцию \_\_fentry\_\_().
4. Ftrace вызывает callback функцию. В процессе работы ftrace вызывает все зарегистрированные трассировочные callback функции.
5. Callback функция выполняет перехват. Эта функция смотрит на значение parent\_ip, указывающее внутрь do\_syscall\_64() - так как именно эта функция вызвала обработчик sys\_execve() - и принимает решение выполнит перехват, обновляя значение регистра %rip в структуре pt\_regs.
6. Ftrace восстанавливает регистры. Следуя флагу FTRACE\_SAVE\_REGS, ftrace сохраняет состояние регистров в структуре pt\_regs перед вызовом обработчиков. При завершении обработки ftrace восстанавливает регистры из этой структуры. Рассматриваемый обработчик изменяет регистр %rip — указатель на следующую исполняемую инструкцию — что в итоге приводит к передаче управления по новому адресу.
7. Управление получает функция-обёртка. Из-за безусловного перехода активация функции sys\_execve() прерывается. Вместо неё управление получает подменяющая функция fh\_sys\_execve(). При этом всё остальное состояние процессора и памяти остаётся без изменений, поэтому данная функция получает все аргументы оригинального обработчика и при завершении вернёт управление в функцию do\_syscall\_64().
8. Обёртка вызывает оригинальную функцию. Функция fh\_sys\_execve() может проанализировать аргументы и контекст системного вызова и запретить или разрешить процессу его выполнение. В случае запрета функция возвращает код ошибки. Иначе же ей следует вызвать оригинальный обработчик — sys\_execve() вызывается повторно, через указатель real\_sys\_execve, который был сохранён при настройке перехвата.
9. Управление получает callback функция. Как и при первом вызове sys\_execve(), управление опять проходит через ftrace и передаётся в callback функцию.
10. Callback функция ничего не делает. Потому что в этот раз функция sys\_execve() вызывается функцией fh\_sys\_execve(), а не ядром из do\_syscall\_64(). Поэтому callback функция не модифицирует регистры и выполнение функции sys\_execve() продолжается как обычно.
11. Управление возвращается обёртке. Во второй раз обработчик системного вызова sys\_execve() вернёт управление в функцию fh\_sys\_execve(). К этому моменту новый процесс уже почти запущен.
12. Управление возвращается ядру. Функция fh\_sys\_execve() завершается и управление переходит в do\_syscall\_64(), которая считает, что системный вызов был завершён как обычно.
13. Управление возвращается в пользовательский процесс. Ядро выполняет инструкцию IRET (или SYSRET, но для execve() — всегда IRET), устанавливая регистры для нового пользовательского процесса и переводя центральный процессор в режим исполнения пользовательского кода. Системный вызов (и запуск нового процесса) завершён.

## ==/\*TO DO\*/==

1. Организация данных и команд на аппаратном уровне: Описать где хранится firmware, где хранятся ключи и промежуточные данные
2. Демон: обеспечение контролируемого доступа к защищаемым ключам подписи со стороны авторизованного пользователя в соответствии с политиками доступа, установленными администратором.
3. Анализ семантики транзакций?
4. Посчитать требуемую память для ключевых пар, массива имен ключей, хранимых логов
5. Посчитать быстродействие и скорость выполнения операций/
6. Обосновать почему вехний порог разовой генерации равен 100 узлов
7. Объяснитm минимальный размер транзацкий
8. Описать алогоритм работы прграммы и обосновать его
9. Объяснить почему везде используют ECDSA (чтобы сохранить последовательность и совместимость с существующим исходным кодом биткойна)
10. Не отчислиться и не умереть от недосыпа и перегруза

==/\*TO DO\*/==

# ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

==/\*TO DO\*/==

# ЗАКЛЮЧЕНИЕ

ПАК управления криптовалютными активами предназначен для безопасного хранения ключей подписи и совершения криптовалютных операций (формирование, подпись и трансляция в сеть криптовалютных транзакций).

Аппаратной основой для ПАК управления криптовалютными активами был выбран микрокомпьютер m-trust с имплементированным микроконтроллером at91sam3u, который будет выполнять роль РКБ.

В качестве операционной системы была выбрана Linux Ubuntu 16.04.

Был составлен перечень готового ПО, необходимого для управления криптовалютными активами:

* криптовалютные кошельки
  + - для Bitcoin: Electrum, Bitcoin Core;
    - для Etherium: Etherium Wallet (Mist);
    - для остальных криптовалют (мультивалютный): Exodus;
* децентрализованная (DEX) биржа, устанавливаемая на компьютер: Waves DEX;
* Браузеры: Mozilla Firefox, Google Chrome.

==/\*TO DO\*/==

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Bitcoin in a nutshell — Mining [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://habr.com/ru/post/320178/>
2. A Next-Generation Smart Contract and Decentralized Application Platform [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://github.com/Ethereum/wiki/wiki/White-Paper>
3. ERC-20 Token Standard [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://eips.ethereum.org/EIPS/eip-20>
4. Токен [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://ru.wikipedia.org/wiki/Токен_(криптовалюта)>
5. Cold storage [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: <https://en.bitcoinwiki.org/wiki/Cold_storage>, свободный
6. Обзор криптовалютных онлайн-обменников [Электронный ресурс] /. — Электрон. журн. — Режим доступа: [https://miningbitcoinguide.com/trading/market/obmenniki-kriptovalyut#\_\_\_2019](https://miningbitcoinguide.com/trading/market/obmenniki-kriptovalyut" \l "___2019)
7. ТОП-15 лучших криптобирж на 2019 год [Электронный ресурс] /. — Электрон. журн. — Режим доступа: [https://mining-cryptocurrency.ru/rejting-luchshie-birzhi-kriptovalyut-2017/#-15\_\_2019](https://mining-cryptocurrency.ru/rejting-luchshie-birzhi-kriptovalyut-2017/" \l "-15__2019)
8. bip-0032 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: <https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>, свободный
9. Топ децентрализованных криптовалютных бирж (DEX) [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://happycoin.club/top-8-detsentralizovannyih-kriptovalyutnyih-birzh-dex/>
10. «0x» protocol: Detailed review on the project [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://medium.com/paradigm-fund/0x-protocol-detailed-review-on-the-project-5913f363805e>
11. ТОП-10 лучших ботов для автоматической торговли криптовалютами в 2019 году [Электронный ресурс] /. — Электрон. журн. — Режим доступа: [https://mining-cryptocurrency.ru/bot-dlya-torgovli-kriptovalyutoj-na-birzhe/#i-6](https://mining-cryptocurrency.ru/bot-dlya-torgovli-kriptovalyutoj-na-birzhe/" \l "i-6)
12. Доверительное управление на биржах криптовалют через API [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://bitexpert.io/news/du-via-api/>
13. ТОП-10: лучшие пулы для майнинга [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://bisnesideya.ru/coins/top-10-luchshie-puly-dlya-majninga.html>
14. Что такое майнинг пул и как он работает [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://investor100.ru/chto-takoe-majning-pul-i-kak-on-rabotaet/>
15. Разработка холодного мультивалютного кошелька на платформе МКТ: НИР / Шарапов Р.А. - Москва: Московский физико-технический институт, 2018. - 64 с.
16. В.А. Конявский, Основы понимания феномена электронного обмена информацией / В.А. Гадасин. - Минск: 2004.
17. Компьютер с «вирусным иммунитетом» [Электронный ресурс] /. — Режим доступа: <http://www.okbsapr.ru/konyavskiy_2015_2.html>, свободный
18. Встраиваем бэкдор в Bitcoin (ECDSA) [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://habr.com/ru/post/248419/>
19. Hardware Wallet Vulnerabilities [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://blog.gridplus.io/hardware-wallet-vulnerabilities-f20688361b88>
20. Конявский В.А. Доверенные информационные технологии. - М.: Ленанд, 2019.
21. m-TrusT // URL: http://www.trustedcloudcomputers.ru/m-trust/ (дата обращения: 18.11.19).
22. bip-0039 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki, свободный
23. bip-0043 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0043.mediawiki, свободный
24. bip-0044 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki, свободный
25. Electrum // GitHub URL: https://github.com/spesmilo/electrum/tree/master/electrum (дата обращения: 18.11.19).
26. IDex // GitHub URL: https://github.com/IDex (дата обращения: 18.11.19).
27. Bitcoin // GitHub URL: https://github.com/bitcoin/bitcoin (дата обращения: 18.11.19).
28. Mist // GitHub URL: https://github.com/Ethereum/Mist/releases (дата обращения: 18.11.19).
29. Exodus // GitHub URL: https://github.com/ExodusMovement (дата обращения: 18.11.19).
30. Waves DEX // GitHub URL: https://github.com/wavesplatform (дата обращения: 18.11.19).
31. Etherdelta // GitHub URL: https://github.com/etherdelta/smart\_contract (дата обращения: 18.11.19).
32. Перехват функций в ядре Linux с помощью ftrace // Habr URL: https://habr.com/ru/post/413241/ (дата обращения: 18.11.19).
33. Трассировка ядра с ftrace [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://habr.com/ru/company/selectel/blog/280322/