МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ (ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

Кафедра защиты информации

**НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА**

По теме: «Разработка холодного мультивалютного кошелька на платформе МКТ»

Выполнил работу:

Шарапов Роман Андреевич,

студент ФРТК 419 группы

Научный руководитель:

Конявский В.А.

Москва – 2018

# 

# Оглавление

[Оглавление 1](#_Toc30152)

[Определения, обозначения и сокращения  2](#_Toc10127)

[Введение 6](#_Toc30294)

[Анализ существующих систем и технологий 8](#_Toc25887)

[Краткий обзор блокчейн технологии 8](#_Toc12566)

[Криптовалютные кошельки 9](#_Toc18214)

[Офлайн подпись транзакций 14](#_Toc3660)

[Hierarchical Deterministic (HD) Wallet 15](#_Toc30541)

[Bitcoin Improvement Proposal (BIP) 21](#_Toc12700)

[Фрагментированные ключи и мультиподпись 25](#_Toc20648)

[U2f (Universal Second Factor) 27](#_Toc16333)

[МКТ и его архитектура   31](#_Toc10996)

[Проектирование системы 38](#_Toc10092)

[Требования, предъявляемые к системе 39](#_Toc300)

[Использование особенностей МКТ для устранения известных уязвимостей 42](#_Toc11172)

[Анализ рынка 52](#_Toc21889)

[Сценарий работы системы 57](#_Toc10959)

[Заключение 60](#_Toc1575)

[Список литературы 62](#_Toc9862)

# Определения, обозначения и сокращения

**Хэш-функции -** это функции, предназначенные для «сжатия» произвольного сообщения или набора данных, записанных, как правило, в двоичном алфавите, в некоторую битовую комбинацию фиксированной длины, называемую сверткой [1].

**Криптографической хэш-функцией** - называетсятакая хэш-функция, которая удовлетворяет следующему ряду критериев [1]:

* *Необратимость* или *стойкость к восстановлению прообраза*: При известном хэше m вычислительно невозможно найти исходное сообщение X за приемлемое время.
* Стойкость к *коллизиям первого рода*: Для заданного сообщения М должно быть вычислительно невозможно подобрать такое сообщение N, при котором хэши этих сообщений совпадали.
* Стойкость к *коллизиям второго рода*: должно быть вычислительно невозможно подобрать пару сообщений (M,M'), имеющих одинаковый хэш.
* Искомая функция должна обладать так называемым эффектом лавины, когда малейшее изменение входных данных значительно преобразует выходное значение.

**Nonce –** это одноразовый код, выбранный случайным или псевдослучайным образом, который используется для безопасной передачи основного пароля, предотвращая атаку повторного воспроизведения.

**Пиринговая (P2P) сеть -** это сеть, в которой все узлы равноправны и равномощны, каждый узел является и клиентом и сервером.[2]

**Имитовставка**(или код аутентификации сообщения) - это набор слов, который добавляется к сообщению для обеспечения целостности, подлинности и аутентификации источника сообщения.

**Генератор псевдослучайных чисел (ГПС)** - алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно равномерному).

**Криптографически стойкий генератор псевдослучайных чисел -** ГПС, который удовлетворяет следующим требованиям:

1. Он должны проходить [статистические тесты на случайность](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D1%85_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9" \o "Тестирование псевдослучайных последовательностей);
2. Он должен сохранять непредсказуемость, даже если часть его исходного или текущего состояния становится известна [криптоаналитику](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D0%BA" \o "Криптоаналитик).

**Криптографическая система с открытым ключом** - система асимметричного шифрования или электронной подписи (ЭП), при которой 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 закрытый ключ.

**Blockchain** (блокчейн, цепь из блоков) – это последовательный набор блоков (или же, в более общем случае, ориентированный граф), каждый следующий блок в котором включает в качестве хэшируемой информации значение хэш-функции от предыдущего блока.

**Майнинг** - процесс решения вычислительной задачи, поставленной протоколом доказательства работы (PoW).

**Мастерноды** - узлы, которые хранят полную версию блокчейна.

**Клиент** - это аппаратный или программный компонент вычислительной системы, управляемый пользователем.

**Узел сети** (англ. node) - устройство, соединённое с другими устройствами как часть компьютерной сети.

**Лайтнод** - хранит только чанки информации, необходимые для работы, за частую лайтнодами являются клиенты.

**Криптовалютный кошелек** -

1. С математической составляющей блокчейна «криптовалютный кошелек» - это пара ключей ( открытый и закрытый)
2. С программной точки зрения «Криптовалютный кошелек» - это программа для хранения ключей пользователя и взаимодействия с блокчейном (просмотр, совершение транзакций).

**Мультивалютный кошелек** - это криптовалютный кошелек для хранения пользовательских ключей разных криптовалют и взаимодействия с их блокчейнами (просмотр, совершение транзакций).

**Горячий криптовалютный кошелек** - это криптовалютный кошелек, имеющий постоянное соединение с интернетом и блокчейном.

**Холодный криптовалютный кошелек** - это криптовалютный кошелек, не имеющий постоянное соединение с интернетом и блокчейном, и с него не совершаются транзакции.

**Блокчейн транзакция** - это подписанный раздел данных, который транслируется в сеть и записываются в блоки. В случае криптовалют транзакции содержат информацию о платежах. Например: [Ivanov->5btc, Petrov->2btc,sidorov->1btc].

**U2F** (Universal 2nd Factor) — это открытый, бездрайверный протокол, использующий для двухфакторной аутентификации аппаратные устройства (например USB накопители или смарт карты), которые хранят ключи и выполняют необходимые криптографические операции.[3]

**ERC-20** (Ethereum Request for Comments) — это официальный протокол для внесения предложений по улучшению сети Ethereum (20 – уникальный идентификационный номер предложения). Протокол установил технические спецификации для токенов, выпускаемых на блокчейне Ethereum, которые известны как токены стандарта ERC-20 и фактически являются смарт-контрактами на блокчейне Ethereum.[4]

# Введение

При использовании горячих кошельков из-за постоянного контакта с сетью, повышается риск компрометации ключей, поэтому для более безопасного хранения криптовалюты используют холодные криптовалютные кошельки.

Холодный кошелек не имеет постоянного соединения с интернетом и блокчейном, и как следствие менее подверженный атакам злоумышленников. Однако, рано или поздно появляется необходимость произвести транзакцию с применением закрытых ключей, хранящихся в холодном кошельке, при этом возникают следующие проблемы: во - первых, нет гарантии, что операция проводится на доверенном компьютере, во-вторых, для выполнения транзакции кошельку необходимо обновить блокчейн, то есть за продолжительное время ему необходимо скачать большой объем информации, и в данных условиях снова повышается риск компрометации ключей, так как за время закачки необходимой информации в систему может скачаться и начать исполняться вредоносный код.

Таким образом возникает противоречие между потенциальными уязвимостями при хранении ключей и проведении транзакций из горячих и холодных кошельков и потребностями общества в безопасном хранении ключей и безопасном проведении транзакций.

Данное противоречие можно разрешить, если в качестве доверенной исполняемой среды использовать облачный микрокомпьютер с динамически изменяемой архитектурой MKT, который позволяет размещать ПО в памяти с физически устанавливаемым доступом read-only[5], что исключает ее искажение и обеспечивает неизменность среды функционирования, а также позволяет нам разделить функции на опасную и безопасную группу, и каждую группу исполнять в своей среде, тем самым повышая надежность системы и снижая риск компроментации.

Исследование такого варианта будет проведено в данной научно исследовательской работе.

**Цель работы:** Спроектировать систему холодного хранения различных криптовалют на платформе доверенного облачного микрокомпьютера с динамически изменяемой архитектурой MKT.

**В рамках данной работы необходимо выполнить следующие *задачи:***

* Исследование существующих подходов к реализации подобных систем, изучение принципов работы аналогов, их достоинств и недостатков
* Формулирование необходимых критериев системы, диктуемых поставленной задачей;
* Эскизное проектирование системы

**Объект исследования**: холодный мультивалютный кошелёк.

**Предмет исследования**: реализация холодного мультивалютного кошелька на платформе МКТ.

# Анализ существующих систем и технологий

В первую очередь изучим существующие технологии и решения на рынке, чтобы при проектировании собственной системы использовать удачные идеи, и избежать известных ошибок.

## Краткий обзор блокчейн технологии

Рассмотрим механизм работы блокчейна, блокчейн транзакций, а также разновидности кошельков.

В блокчейн имплементирована программа с внутренним хранилищем в которой хранится реестр монет, который представляет собой массив ассоциаций аккаунтов с суммами. Полный доступ к данному реестру имеет только эта программа, она же предоставляет всем желающим возможности просматривать суммы на аккаунтах и переводить сумм между аккаунтами, 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 закрытого ключа из классической пары шифрования с открытым ключом. 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[6] .

В большинстве криптовалют действует система, 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).

Распространённой практикой является то, что люди, нежелающие подвергать себя лишнему риску компрометации ключей, посылают сдачу на свой другой только что созданный аккаунт. В такой схеме аккаунты (открытые ключи) являются одноразовыми, и после первой же произведенной транзакции больше не используются.

В данной работе при проектировании системы необходимо учесть возможность использования одноразовых ключевых пар для снижения риска компрометации ключей и потери средств.

## 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 передаче в эту саму программу информации “о направлении передачи монет”, с сигнатурой для идентификации пользователя как владельца. В конечном итоге программа должна уметь просканировать реестр блокчейна, прочитать там баланс пользователя, затем изменить его и баланс получателя транзакции.

Рассмотрим классификацию кошельков:

* По подключению к интернету:
  + Горячие - это криптовалютные кошельки, имеющие постоянное соединение с интернетом и блокчейном.
  + Холодный кошелек - это криптовалютные кошельки, не имеющие постоянное соединение с интернетом и блокчейном, при этом, с них не совершаются транзакции. [7]
* По количеству поддерживаемых монет:
  + Одновалютные – кошельки поддерживающие операции только с одной криптовалютой.
  + Мультивалютные – кошельки поддерживающие операции с несколькими криптовалютами. Современные мультивалютные кошельки поддерживают от 10 монет и выше, а также токены стандарта ERC-20.
* Классификация по месту размещения кошелька и блокчейна:
  + Онлайн кошельки – горячий кошелёк, запускаемый через обычный браузер. В таком кошельке закрытые ключи хранятся либо на стороннем сервере в сети в зашифрованном виде, либо передаются с компьютера пользователя при подключении к сервису. Данные кошельки - самый не безопасный, и подверженный риску вариант. Однако он очень хорошо подходит для оперативных расходов и малых сумм, за счет своей быстроты и доступности.
  + «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, то есть данный кошелек также является горячим. Зачастую в таких кошельках есть возможность управления ключами пользователя, (то есть хранить ключи в устройстве, распечатать ключи и удалить ключи) но так как ключи все равно посылаются в сеть на сторонние сервера, то данный вариант считается не безопасным.
  + «Толстые» локальные кошельки – кошельки, устанавливаемые на компьютере пользователя, которые общаются только со своим, заранее скаченным и постоянно обновляемым блокчейном. Этот тип кошельков также является горячим
  + Бумажные кошельки – представляют собой ключевую пару в виде QR кодов или текстовых строк, распечатанных на бумажном носителе. Программный кошелёк (или специализированный сервис) генерирует пару ключей и трансформирует их QR коды. Затем закрытый ключ распечатывается и удаляется с цифрового носителя, а открытый ключ размещается в публичном доступе для приема средств. Это наиболее безопасный вариант (с точки зрения компьютерной безопасности) для хранения и накопления больших сумм криптовалюты. Как только возникает необходимость произвести операцию с этого адреса (то есть использовать закрытый ключ), то повышается вероятность взлома, поэтому необходимо создать новый аналогичный бумажный кошелек и перевести сдачу с операции на этот второй адрес.
  + Аппаратные кошельки – кошельки на аппаратных отчуждаемых носителях (обычно флеш накопителях или смарт картах), ведущие взаимодействие с блокчейном через локальный кошелек или же через сайт. Такие кошельки, хранят закрытый ключ внутри себя и получить к нему доступ через сеть будет нельзя (на выходе прибора выдается только уже сформированная транзакция). Такие девайсы часто защищены дополнительными уровнями защиты (сканер отпечатков, пин код, пароль и тд) и не позволяют записывать в себя новую информацию, то есть работают в режиме read only. Однако, если при проведении транзакции пользователь использует не доверенный компьютер, то, хоть риск кражи закрытого ключа минимален, есть вероятность того, что злоумышленник может отправить в устройство на подпись фальшивую транзакцию (например заменить адрес получателя на посторонний), и затем эта транзакция будет послана в блокчейн.

Все типы горячих кошельков являются 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 ключа снижается, но остается, так как злоумышленники, применяя аналитические методы и вредоносное программное обеспечение, все равно могут получить доступ к закрытым ключам.

Подытожим информацию по двум подразделам и распишем ключевую информацию от которой мы будем отталкивать при построении собственной системы:

* Криптовалюта как деньги — это не более чем счетчик, записанный в реестре валют блокчейн программы.
* Криптовалюта хранится не в кошельке, а в реестре валюты в блокчейне,
* У кого есть закрытый ключ от аккаунта, на котором лежат деньги, тот может переводить деньги в независимости от мнения оригинального владельца пары ключей.
* В случае кражи ключей, и вывода денег, злоумышленника вычислить крайне трудно, практически невозможно
* Хранить криптовалюту в безопасности означает хранить в безопасности закрытые ключи,
* Неважно откуда или с какого типа кошелька идет перевод денег, важно, чтобы программа получила из блокчейна корректную идентификационную информацию.
* «Холодный кошелек» означает, что кошелек не требует постоянного подключения к интернету. А горячий, соответственно, требует подключения к сети, что делает его более уязвимым для атак.
* Пока не проведен ни один платеж с холодного кошелька (ни разу не использован приватный ключ) — холодный криптокошелек будет столь же надежен, как и аппаратный или бумажный.
* Проверять получение платежей на свой адрес можно на сторонних сервисах контроля блокчейна, по запросу на открытый ключ пользователя.
* После проведения первой транзакции с холодного кошелька, кошелек перестает быть холодным в чистом виде (из пассивного режима на короткий период переходит в активный), из-за возрастает риск компроментации ключа. Чем больше хранимая сумма - тем больше риск.
* Послетранзакционный риск можно нивелировать, если использовать одноразовые ключевые пары, и после каждой транзакции переводить сдачу на новый адрес.
* Снижение и ликвидация риска компроментации ключа во время соединения с интернетом - это и есть одна из задач этой работы.

Согласно цели НИР - в данной работе мы будем рассматривать реализацию локального холодного кошелька, поддерживающий и «тонкий» и «толстый» режимы работы, при этом сохраняющий безопасность ключей как после транзакции, так и во время совершения транзакции.

## Офлайн подпись транзакций

Есть модификация холодного хранения, которая применяется для уменьшения риска компрометации в момент совершения транзакции:

* Система состоит из двух компьютеров, один из них всегда отключен от интернета, второй - подключен к сети
* На первом компьютере с помощью кошелька создаем пару ключей, импортируем открытый ключ на второй компьютер. После чего на этот адрес 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 отсылается в сеть.

Ошибочно считается, что этот метод является абсолютно безопасным, потому что Интернет и закрытые ключи никогда не контактируют. Однако при импорте транзакции на носитель, даже при всех мерах безопасности, все равно остается вероятность загрузки исполняемого вредоносного кода и на съемный носитель и на сам компьютер, что может привести к подмене данных в транзакции, краже или повреждению закрытого ключа.

При проектировке системы в данной исследовательской работе необходимо учесть эту уязвимость и не допустить ее в итоговом решении.

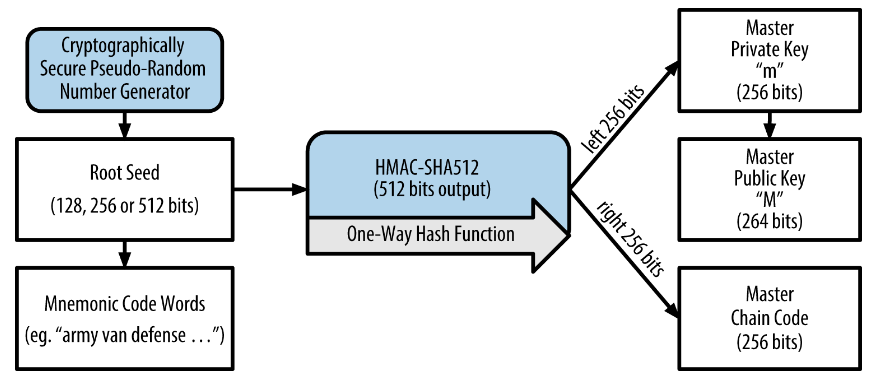
## Hierarchical Deterministic (HD) Wallet

Рассмотрим технологию, позволяющую удобным образом реализовать одноразовые ключевые пары.

Deterministic wallet — это кошелек, в котором все используемые личные ключи были порождены 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. Стандартизированный подход к кодированию основного секрета расписан в протоколах семейства BIP (Bitcoin Improvement Protocol). [8]

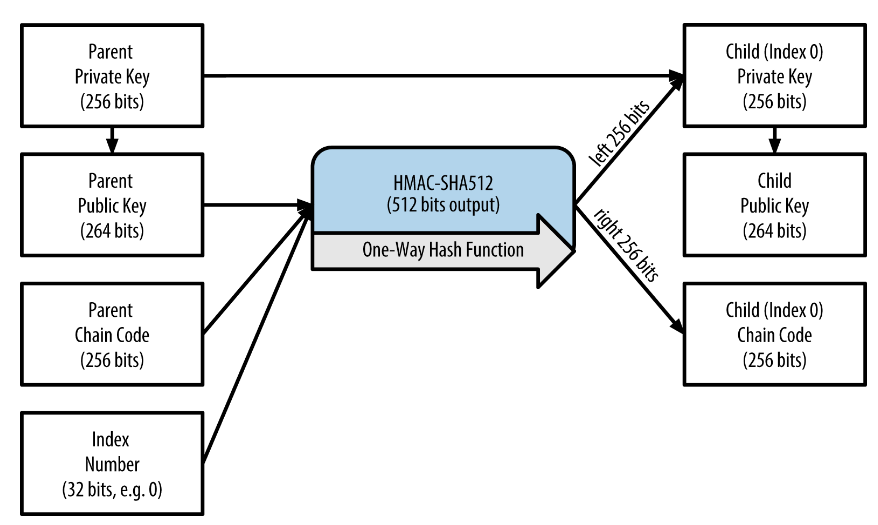
Детерминистические кошельки бывают двух типов:

* Простой детерминистический кошелек: Основной секрет здесь конкатенируется с индексом дочернего ключа, который мы хотим получить, после чего конкатенированные данные хешируются, с помощью хеш-функции SHA-256. Кошелек имеет некоторый seed, из которого напрямую генерируется огромное множество личных ключей. Их количество может быть ограничено только размерностью индекса, который конкатенируется к секрету перед хешированием. Обычно это 4 байта, то есть пространство возможных вариантов допускает 232, а это около 4 миллиардов уникальных ключей.
* Иерархически детерминистический кошелек (hierarchical deterministic wallets, HD wallets): На каждом уровне иерархии узел порождения имеет три объекта: личный ключ (private key), открытый ключ (public key) и код цепочки (chain code), который используется для порождения следующего уровня иерархии. Рассмотрим более детально схему генерации ключей по стандарту BIP32 (Hierarchical Deterministic Wallets), в котором определены принципы работы этих кошельков: [8]
  + Из master seed (или root seed, как его еще называют) рассчитывается нулевой уровень иерархии — пара master keys и chain code: Для создания этого закрытого мастер-ключа “зерно” обрабатывается с помощью хэш-функции HMAC-SHA512. Результат (512-битное число) - делится на 2 части по 256 бит в каждой: собственно закрытый мастер-ключ (он же родительский закрытый ключ) и “код цепи”.



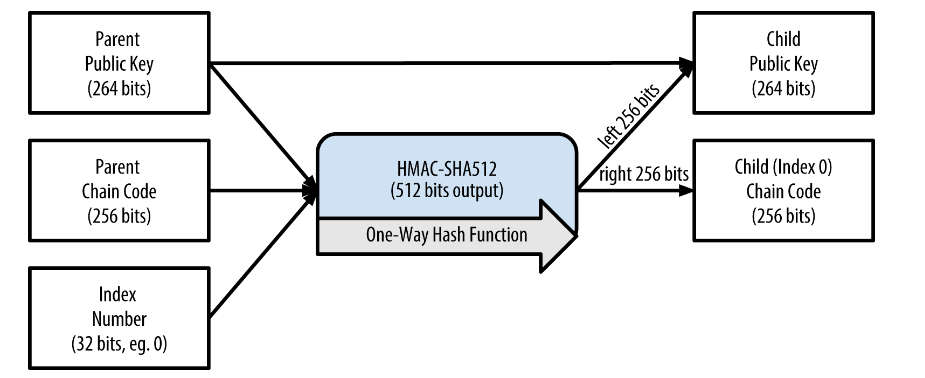
(Рис.1) Извлечение пары мастер ключей из seed-а

* + Формируется новый уровень иерархии, который используется для генерации аккаунтов: ключи получаются из комбинации расширенного ключа (сочетание ключа и “кода цепи”) и 32-битного индекса (порядкового номера) с помощью HMAC-SHA512. Результат разделяется на дочерние закрытый ключ и код цепи. Затем процедуру можно повторить с полученным дочерним ключом, использовав его в качестве “родителя” нового поколения ключей.



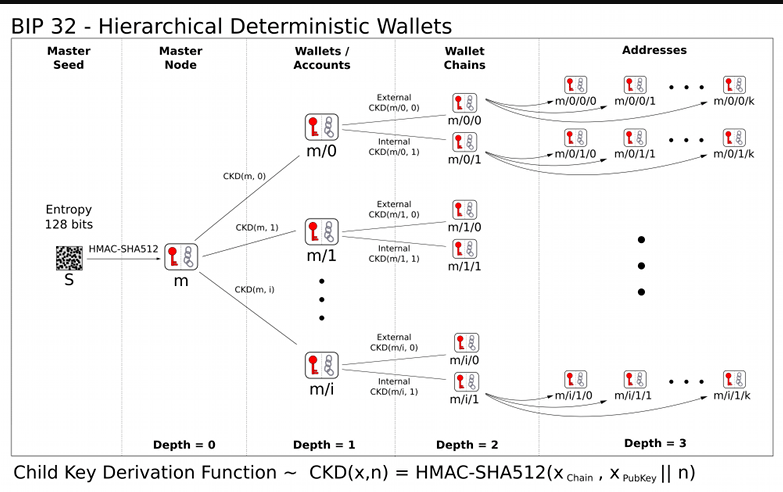
(Рис.2) Извлечение дочерних ключевых пар из родительских ключевых пар

* + Следующий уровень иерархии определяет разные цепочки генерирования ключей. Чаще всего используются цепочки с индексами 0 и 1. Цепочка с индексом 0 будет генерировать конечные ключи для формирования адреса для входящих платежей, а цепочка с индексом 1 будет генерировать кошельки, на которые будут приходить «сдача». Это нужно, чтобы кошелек на программном уровне отличал отправленные извне платежи от сдачи, рассчитывал изменения баланса каждой транзакции и составлял наглядный список с историей всех платежей.
  + Помимо “традиционного” способа генерации - из приватных ключей - дочерние публичные ключи в HD-кошельке можно получить из родительского публичного ключа, используя процедуру, аналогичной приведенной выше. Только вместо расширенного закрытого ключа применяется расширенный открытый. Это позволяет создавать неограниченное число публичных ключей, даже не имея доступа к приватному ключу.



(Рис.3) Извлечение дочерних открытых ключей из родительских расширенных открытых ключей

Полную картину иерархии ключей можно увидеть на рисунке ниже



(Рис.4) Полная иерархия ключей в технологии HD Wallet

Стоит отметить, что получение из родительского открытого ключа дочернего личного невозможно. То есть в данных системах можно выполнить следующие операции по развертыванию ключа:

* + Master seed --> master key
  + Private parent key --> private child key
  + Private parent key --> public child key
  + public parent key --> public child key

Если на каком либо сервисе пользователь с очень высокой частотой совершает платежи, при этом использует метод одноразовых адресов, то есть каждый раз заново создает новую пару ключей, то это создает определенные неудобства в использовании и путаницу для пользователя, а также забивают базу данных сервиса лишней информацией.

В подобной ситуации от неудобств помогает избавиться **расширенный открытый ключ** (extended public key, xPubKey). Пользователь может дать возможность стороннему сервису вместо себя генерировать такие адреса, которые будут известны сервису, но закрытые ключи будут только у пользователя. Сервис может генерировать какое угодно количество адресов (открытых ключей) по стандарту BIP32 без ведома пользователя и отправлять на них средства, а пользователь может развернуть закрытые ключи и получить доступ к любому из этих адресов.[8] Для этого нужно:

* Сгенерировать новый аккаунт на втором уровне иерархии ключей
* Рассчитать для него открытый ключ и chain code для текущего уровня
* Передать сервису открытый ключ и chain code, в стандарте кодирования Base58Check Encoding (система счисления по основанию 58).
* Конкатенировать данные (открытый ключ, chain code и контрольная сумма) в единый пакет
* Кодировать пакет в Base58, добавить в начало строку “xpub”.
* На выходе получаем закодированный открытый расширенный ключ: *xPubKey = Base58(version || PublicKey || ChainCode || CheckSum)*

Рассмотрим следующий подход - **hardened derivation**. Это метод, не позволяющий рассчитывать дочерние открытые ключи из соответствующего родительского открытого ключа.[8] От обычного порождения отличается тем, что в обычном в качестве сообщения функции HMAC используется конкатенация сериализованной точки на эллиптической кривой, в качестве родительского открытого ключа, а в hardened derivation мы используем сериализацию родительского личного ключа. Кроме того, индекс в обычном порождении напрямую сериализуется в 32 бита, а в hardened derivation он несколько преобразуется: к нему добавляется константное значение 231, что устанавливает старший бит в 1 (становится легко отличать типы деривации). Соответственно, пространство вариантов возможных ключей одинаково как для нормального порождения, так и для hardened derivation и равно 231.

Таким образом, имея родительский открытый ключ и hardened derivation, невозможно рассчитать дочерние открытые ключи. Если злоумышленник получит родительский открытый ключ, то он не сможет вычислить дочерние ключи. Следовательно, не сможет вычислить адреса и связь их с полученным родительским ключом. В случае normal derivation, то есть в обычном, такой функцией можно пользоваться и прослеживать взаимосвязь адресов между собой. Структура извлечений ключей:

* Normal derivation: HMAC (CCpar, serp(point(kpar)) || ser32(i)) , i = index
* Hardened dervivation: HMAC (CCpar, ser256(kpar) || ser32(i)) , i = index+231

Рассмотрим механизм хранения и идентификации ключей внутри системы.

На каждом уровне иерархии есть определенный индекс, который определяет аспекты порождения ключей. Путь от Мастер-ключа до конечного ключа может записываться через слэши в виде индексов. Если речь идет о личном ключе то запись начинается с маленькой “m”, а если речь о порождении открытого ключа, то с большой “M”. Если индекс обозначен апострофом, то следует понимать, что речь идет о hardened derivation, без апострофа — normal derivation. К примеру:

*Private keys: m/4’/6/1*

*Public keys: M/4’/6/1*

Таким образом у каждого ключа появляется собственный путь ,который одновременно может служить именем этого ключа в кошельке. Такая идентификация активно применяется в стандартах BIP.

Подведем итог раздела: технология HD Wallet позволяет оперировать очень большим (до 4 млрд) количеством аккаунтов в разных блокчейн системах, сохраняя анонимность платежей, и при этом от пользователя требуется всего лишь знать один мастер ключ (или мнемоническую фразу, при использовании стандарта BIP39). Решение является очень удобным для пользователя и эффективным с точки зрения эффективности и безопасности, кроме того, оно обеспечивает мультивалютность кошелька, поэтому HD wallet целесообразно использовать при проектировании системы в рамках данной работы.

## Bitcoin Improvement Proposal (BIP)

Рассмотрим основные протоколы из семейства BIP, активно применяемые при проектировании криптовалютных кошельков:

* **BIP32** (Hierarchical Deterministic Wallets) [9]

BIP32 был первым протоколом, где и были определены иерархические ключи. Он использует такой путь, где нулевым уровнем иерархии является мастер-ключ. Далее следуют индексы аккаунтов, которые обозначают одного и того же пользователя, после чего идут цепочки, где могут быть цепочки адресов, которые публикуются вовне для принятия входящих платежей, а с индексом 1 будут создаваться те цепочки, на которые сам пользователь отправляет себе платежи (сдача). Конечный индекс будет использоваться для порождения тех ключей, из которых будут рассчитываться адреса. Для того чтобы по стандарту BIP32 рассчитать самый первый ключ с индексом 0, мы будем иметь m, 0 с порождением hardened, chain — 0, индекс — 0 (m/0’/0/0). Так мы получим путь для первого иерархически порожденного ключа. Пример:

*m / accounts’ / chains / addresses <===> m /0’/0/0*

* **BIP39** (Mnemonic code for generating deterministic keys) [10]

При развертывании ключей часто применяется протокол BIP39 - кодирование основного секрета в мнемоническую фразу - набор обычных слов, который легко запомнить.

При вводе и выводе фразы есть возможность проверить контрольную сумму, то есть с большой вероятностью выявить ошибку, если такая имеется. Это работает следующим образом: есть основной секрет (Entropy) — это данные, из которых разворачиваются все личные ключи кошелька, при этом секрет может иметь разную длину. На каждые 32 бита Entropy приходится 1 бит контрольной суммы, то есть Checksum по формуле рассчитывается, как длина Entropy в битах, разделенная на 32, то есть: checksum = entropy/32, words = (entropy + checksum)/11.

Entropy конкатенируется с контрольной суммой, которая рассчитывается, как двойной хеш SHA-256 (SHA-2 на длине 256 бит), после чего отрезается необходимое количество битов. Конкатенированные данные переводятся в другую систему счисления: из двоичной в систему счисления по основанию 2048 (211). И если сложить длину битов Entropy и контрольной суммы, то получится число, кратное 11-ти. Таким образом, мы получаем количество слов в выходной мнемонической фразе.

Фактически данные “нарезаются” частями по 11 бит. Есть словарь, состоящий из 2048 (211) слов, к которым применены определенные требования. По умолчанию язык словаря английский, но может использоваться любой. Слова не должны превышать определенную длину (обычно предел до 7 символов). Все они должны быть закодированы в UTF-8 с определенной нормализацией всех символов.

Уникальность каждого слова является обязательной по первым четырем символам, которые однозначно определяют слово в словаре, а остальные символы используются, чтобы завершить это слово до удобной формы для чтения, запоминания и т. д. Таким образом каждый фрагмент данных, состоящий из 11-ти бит, получает однозначное соответствие в виде слова из словаря. Если Entropy секрета составляет 256 бит, то данные для кодирования составят 264 бита, а мнемоническая фраза будет содержать 24 слова. Это основной подход к кодированию секрета кошельков в BIP39, который применяется на практике чаще всего, и при этом хорошо подходит для создания резервных копий.

* **BIP43** (Purpose Field for Deterministic Wallets) [11] и **BIP44** (Multi-Account Hierarchy for Deterministic Wallets).[12]

BIP43 предполагает запись в первый уровень иерархии номера улучшения, которое предлагает новый путь порождения (m/bip\_number’/\*) таким образом, появился BIP44, который использовал особенность предыдущего предложения, то есть для первого уровня иерархии записывается индекс 44, а предложил следующие улучшения: в индексе второго уровня иерархии записывать определенное значение, которое будет соответствовать типу монеты, которую мы используем для данного кошелька. Теперь в одном кошельке могут разворачиваться и использоваться ключи для разных валют.

К примеру:

* + Общая структура: *m / 44' / currency\_number / account / chain / address*
  + Для Биткоина путь будет выглядеть, как *“m/44’/0’/0’/0/0”*
  + Для Bitcoin testnet — *“m/44’/1’/0’/0/0”*
  + Ддля Litecoin — *“m/44’/2’/0’/0/0”*
  + Для Dash — *“m/44’/5’/0’/0/0”*
  + Для Ethereum — *“m/44’/60’/0’/0/0”*
  + И так далее

Этот протокол является существенно важным, при создании мультивалютного кошелька.

* **BIP45** (Structure for Deterministic P2SH Multisignature Wallets) [13]

Протокол определяет правила порождения ключей в случае их использования в multisignature кошельках (подробнее эта технология рассмотрена в следующем разделе) и формирования адресов таких кошельков. Он включает в себя предложение BIP43 и указывает на первом уровне иерархии индекс 45, на втором же уровне иерархии он требует указания подписанта (cosigner). Например, есть правило мультиподписи 3-из-5. Следовательно есть 5 подписантов, но чтобы потратить монеты, нужно как минимум 3 подписи. Таким образом, каждый из подписантов будет иметь HD кошелек со своим мастер-сидом, а в своем пути будет указывать свой порядковый номер. Он может быть рассчитан, как индекс при сортировке ключей, порожденных на первом уровне иерархии каждого пользователя. Допустим, на первом уровне произошло порождение ключей у каждого пользователя, они обмениваются друг с другом, сортируют и узнают, у кого какой индекс для второго уровня иерархии. Это нужно, чтобы в дальнейшем исключить необходимость взаимодействовать подобным образом, а сразу правильно генерировать адреса и знать свой порядковый номер. То есть можно единожды обменяться расширенным открытым ключом, чтобы потом самостоятельно, независимо от других участников группы, формировать multisignature адреса и принимать на них платежи.

Общая структура: *m / 45' / cosigner / chain / assress*

В данном разделе были перечислены стандарты и протоколы, необходимые для имплементации технологии hd wallet в проектируемый в данной работе мультивалютный кошелек.

## Фрагментированные ключи и мультиподпись

Если хранить только одну копию ключа - то велик риск потери ключа, а если хранить несколько копий одного и того же секрета, то повышается риск компрометации. Для решения этой проблемы применяется пороговое разделение секрета.

Идея пороговой схемы (n, N) - разделения общего секрета состоит в следующем:

* Каждый из доверенных пользователей получает долю секрета
* Любые m ( n <= m <= N ) легальных пользователей могу восстановить секрет при предъявлении своих долей
* Любые m ( n > m) легальных пользователей не могу восстановить секрет даже при наличии неограниченных вычислительных ресурсов

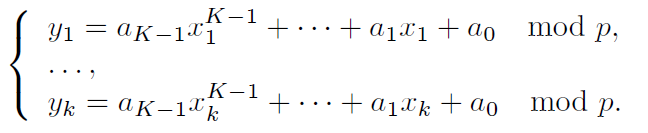
При разделении закрытых ключей от криптовалютного кошелька чаще всего используют схему Шамира,[14] которая основывается на восстановлении всех коэффициентов полинома лагранжа P(x) =an-1xn-1+…+a1x+a0 степени (n-1), для чего требуется n координат различных точек, принадлежащих кривой палинома, при этом все операции проходят в конечном поле GF(p).

Для разделения секрета S между N сторонами таким образом, чтобы любые n сторон и больше могли восстановить секрет, доверенный центр выполняет следующие операции:

* выбирает несекретное большое простое число p: p>S;
* в качестве свободного члена секретного многочлена полагает разделяемый секрет a0 =S
* выбирает остальные секретные коэффициенты многочлена a1,...,an-1 <p
* выбирает N различных аргументов xi, i < N и xi <p
* для каждого выбранного xi вычисляется соответствующий yi, секретного многочлена yi=an-1xn-1+…+a1x+S mod p
* Каждому легальному пользователю раздается доля секрета в виде (xi,yi)

Если n пользователей собираются вместе и объединяют свои доли, то получается система из n уравнений с n неизвестными, из которого реально получить все коэффициенты включая a0=S.

Если собрано меньшее количество долей то система уравнений будет иметь *p* различных равновероятных решений относительно S. При достаточно большом *p* перебор этих значений слишком долог и ресурсозатратен.



Когда речь заходит о крупных суммах, то в реальных бизнес ситуациях для проведения операции обычно необходимо согласование нескольких человек (например директора и бухгалтера). Большинство современных криптовалют и кошельков поддерживают такую опцию, позволяя опционально выбирать количество электронных подписей необходимых для подтверждения транзакций.

Мультиподпись бывает двух видов:

1. m из n - то есть пороговое разделение секрета
2. n из n - может быть как пороговым разделением секрета, так и обычной подписью всех легитимных пользователей кошелька. Например в случае алгоритма RSA:  *s= mD mod n = md1\*d2\*…\*dn mod n*

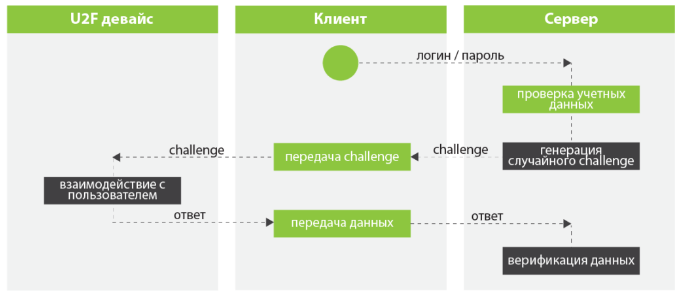
Подытожим раздел: исходя из соображений безопасности и требований реалий бизнеса проектируемый кошелек должен поддерживать возможность мультиподписи и порогового разделения секрета, механизмы которых были подробно рассмотрены выше.

## U2f (Universal Second Factor)

Распространенным вариантом аутентификации является электронная подпись с помощью отчуждаемого носителя (USB накопитель или смарт карта). Данный метод строится на протоколе U2F, основанном на challenge-response аутентификации, семействе протоколов, в которых одна сторона представляет вопрос, "вызов", а другая сторона должна предоставить действительный ответ, "отклик", для проверки подлинности. Модификация этой идеи привела к появлению аппаратных кошельков, таких как Trezor, Lenger Nano S и так далее.

Рассмотрим протокол подробнее. С точки зрения пользователя работа с протоколом достаточно тривиальна: пользователь вводит логин и пароль, вставляет в компьютер USB U2F устройство, затем проходит второй фактор (нажимает кнопку на устройстве, вводит пин-код или проходит биометрическую проверку) и успешно проходит аутентификацию.

Процесс challenge-response аутентификации выглядит следующим образом:



(Рис. 5) Схема работы challenge-response аутентификации для U2F устройства

* Пользователь вводит логин и пароль
* Сервер проверяет учетные данные и, если они верны, генерирует случайный challenge («соль» или текущие дату и время) и отправляет его клиентскому ПО
* Клиент передает полученные данные U2F-устройству, перед этим, для защиты от фишинга, добавив к challenge origin URL и [TLS channel ID](https://en.wikipedia.org/wiki/Transport_Layer_Security_Channel_ID" \t "https://tproger.ru/articles/fido-u2f/_blank)
* Устройство ожидает отклика пользователя для подтверждения дальнейших операций и затем, возвращает клиентскому ПО подписанный challenge
* Данные передаются дальше на сервер для сверки подписи.

Рассмотрим процесс работы устройсвтва [15]:

* Регистрационно - зависимая пара ключей: Чтобы не подписывать все одной парой ключей, при регистрации сервер передает вместе с challenge параметры application ID и random seed, на основе которых U2F-устройство и генерирует уникальную пару Registering Dependent Keys. За счет того, что пара ключей уникальна для каждой регистрации, становится возможным использовать совместно одно U2F-устройство для множества аккаунтов и сервисов.
* Защита от клонирования: Для того, чтобы защитить U2F-устройство от копирования, стандарт предусматривает в нем встроенный счетчик. Каждая подпись и регистрация увеличивает состояние счетчика оригинала на единицу. Состояние счетчика подписывается и возвращается зависимой стороне вместе с response. Если U2F-устройство будет клонировано, то состояние счетчика клонированного устройства будет меньше, чем состояние счетчика оригинального устройства, что вызовет ошибку во время верификации.
* Внутренне устройство: Перечислим данные изначально записанные в памяти U2F Ключа:
  + Сертификат производителя, который содержит публичный и секретный ключ для цифровой подписи ECDSA.
  + Ключи хранятся в защищенной ПЗУ памяти, и их невозможно удалить, изменить или извлечь через APDU команды

Перечислим фактический функционал U2F устройства в стандарте v.1 :

1. Регистрация системы аутентификации (СА):
   1. Устройство внутри себя генерирует пару ключей ECDSA,
   2. шифрует секретный ключ (используя свой секрет и параметр СА), тем самым получая Key Handle
   3. Отправляет пакет (Key Handle || signature || sertificate) клиенту.
2. Сохранение в СА:
   1. параметр,
   2. публичный ключ и зашифрованный секретный ключ (Key Handle).
3. Верификация воли пользователя на аутентификацию : необходимость дотронуться до корпуса устройства или нажать на кнопку.
4. Генерация случайных чисел с использованием своего собственного внутреннего генератора, чтобы система никогда не возвращала одни и те же ключи при регистрации даже если все остальные входные данные одинаковые;
5. Проверка зашифрованного секретного ключа ECDSA и параметра системы: U2F устройство должно уметь расшифровать KeyHandle, используя свой внутренний секрет и параметр СА, чтобы определить какому пользователю принадлежит ключ.
6. Подпись challenge:  
   Если проверка key handle прошла успешно, то устройство вернет цифровую подпись пакета(challenge || param || counter).

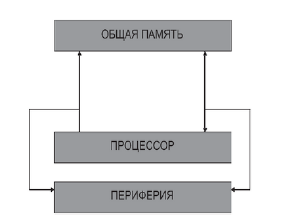
Подытожим раздел: U2F устройства позволяют безопасно хранить закрытые ключи и использовать их для подписи, кроме того они распространены на рынке и популярны. Поэтому проектируемая система по возможности должна иметь поддержку аутентификации и подписи транзакций с помощью аппаратных средств. Однако в этом случае надо будет решить проблему, обозначенную выше - если компьютер, канал связи или система в целом скомпрометированы, то есть риск того, что на вход в устройство могут подать ложную транзакцию, и устройство его все равно подпишет.

## МКТ и его архитектура

Рассмотрим доверенный облачный микрокомпьютер с динамически изменяемой архитектурой MKT-card, особенности его работы и архитектуры, основанной на новой гарвардской схеме.

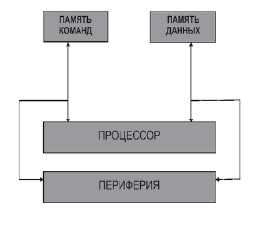
Для начала рассмотрим классические подходы при построении архитектуры компьютерных систем:

* Архитектура фон Неймана – реализована практически во всех настольных компьютерах: Команды и данные не разделяются, а  передаются по единому общему каналу



(Рис.6) Архитектура фон неймана

* Гарвардская архитектура – реализована практически во всех планшетных компьютерах и телефонах: потоки команд и данных параллельны и  независимы, что требует более сложной организации процессора, но обеспечивает более высокое быстродействие



(Рис.6) Гарвардская архитектура

Классические архитектуры уязвимы, так как в них есть возможность изменения последовательности команд и данных независимо от того, размещены они в одной памяти или разделены. Как следствие – возможность для несанкционированного вмешательства в логику работы с помощью вредоносного программного обеспечения

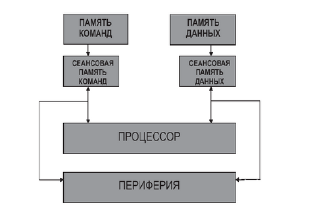
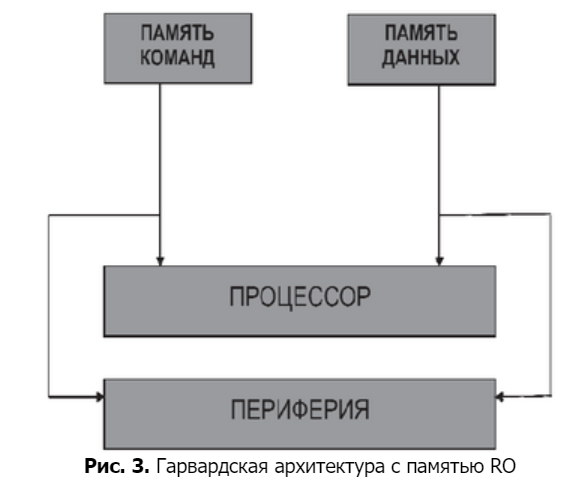
На использовании перехвата управления основаны большинство современных хакерских атак. Схема атаки обычно выглядит так[5]:

1. подготовка атаки:
   1. внедряется и размещается в оперативной памяти вредоносное ПО (ВрПО);
   2. внедряется и размещается в оперативной памяти вредоносный обработчик прерываний;
   3. записывается в долговременную память ВрПО и обработчик прерываний;
2. инициирование атаки: с помощью любого доступного механизма вызывается прерывание, например, с помощью DDOS-атаки;
3. Атака:
   1. внедренный ранее обработчик прерываний срабатывает и передает управление ВрПО;
   2. ВрПО выполняет свою функцию, например, реализует разрушающее программное воздействие (РПВ).

Запись в долговременную память можно заблокировать если использовать механизмы контроля целостности данных, причем проверка целостности должна идти до загрузки операционной системы. Атаку можно заблокировать с помощью механизмов контроля запуска задач (процессов, потоков). Это очень эффективные, но дорогостоящие механизмы, которые должны быть реализованы аппаратно, а не программно

Чтобы предотвратить эти атаки, необходимо добавить неизменяемую память, разделить потоки команд и данных, исполнить контрольные процедуры в доверенной среде до запуска ОС. В гарвардской архитектуре уже есть разделение потоков команд и данных, поэтому если сделать память неизменяемой и разрешить движение данных и команд только по направлению к процессору (рис 7), то ВРПО не будет фиксироваться и исполняться на компьютере, однако в этом случае не сможет исполняться и часть обычных программ, которым необходима запись данных в память для нормальной работы.

Эти противоречия разрешает новая гарвардская архитектура – модификация гарвардской архитектуры, с использованием неизменяемой памяти (что избавляет от необходимости использовать сложные механизмы контроля целостности программ и данных до запуска ОС) и блоков сеансовой памяти (для возможности исполнения программ, для большей части из которых необходима возможность записи).[5]



(Рис.8) Новая гарвардская архитектура

(Рис.7) Гарвардская архитектура

с памятью read only

Таким образом новая архитектура обладает динамической изменяемостью, что обеспечивает защищенность от вирусов и неизменность операционной системы, при этом не ограничивая возможности применения всего программного обеспечения, написанного для них.

Новая гарвардская архитектура является одним из ключевых решений в доверенном облачном микрокомпьютере с динамически изменяемой архитектурой MKT. Эта динамическая изменяемость обеспечивается использованием док-станции с отчуждаемым компьютером.



(Рис.9) Новая гарвардская архитектура реализованная в МКТ

 Рассмотрим еще одно применение этой схемы на практике. Есть целая ветка семейства компьютеров на базе новой гарвардской архитектуре MKT - компьютеры MKTrusT, которые позволяют работать в одном из двух режимов — защищенном (read only) или незащищенном (read and write).[16] Работа в этих режимах производится в разных ОС, загружающихся из разных, физически разделенных, разделов памяти (то есть взаимовлияние ОС исключено технологически, как исключен и их одновременный запуск). Переключение режимов работы производится с помощью физического переключателя, расположенного на корпусе устройства, то есть необходимый режим выбирает пользователь, и не может выбрать хакер, так как невозможно программное воздействие на выбор режима.

При первоначальной подготовке такого компьютера к работе, операционную систему необходимо записать в защищенный банк памяти, то есть в этот момент память должна быть доступна не только не чтение, но и на запись (RW), что можно сделать физическим переключателем, размещенным внутри корпуса устройства. Доступ к этому переключателю должен быть только у специалистов на аттестованном производстве

Пользователю в процессе работы может понадобиться выйти в интернет, и в таком случае ОС может перестать быть доверенной. Поэтому выход в интеренет осуществляется только со второй, незащищенной, ОС. У пользователя есть возможность самому выбрать, в какой из операционных систем ему работать. Эта возможность обеспечивается физическим переключателем снаружи корпуса устройства.

Если предполагается, что данные, обрабатываемые в защищенном режиме, подвержены постоянному санкционированному изменению, то пытаться обеспечивать целостность этих данных бессмысленно. Но возможно ее контролировать. Для этого в режиме локальной обработки информации должно быть установлено СЗИ, оснащенное собственной подсистемой контроля целостности. Далее, очевидно, что в общем случае к данным на ПК необходимо разграничивать доступ, а значит, это СЗИ должно также обладать собственными подсистемами идентификации/ аутентификации и разграничения доступа.

Еще раз перечислим ключевые особенности компьютеров семейства МКТ:

* Возможность реализации процедур взаимной аутентификации док-станции и отчуждаемого компьютера
* размещение ПО в памяти с физически устанавливаемым доступом read-only, что исключает его искажение и обеспечивает неизменность среды функционирования
* встроенные средства разграничения доступа
* встроенные средства защищенного терминального доступа
* встроенные средства «проброса» токенов и других периферийных устройств на  удаленный рабочий стол
* среда функционирования криптографии
* высокий уровень «вирусного иммунитета»
* возможность создания и поддержки доверенной среды
* Можно использовать без доработок все  ранее разработанное ПО
* Отсутствует необходимость использовать сложные механизмы контроля целостности программ и данных до старта ОС
* Контрольные процедуры можно исполнять под управлением проверенной и неизменяемой ОС
* МКТ - это полноценный ПК, который должен иметь две ОС.
  + Одна из них должна быть размещена в защищенной от несанкционированной модификации памяти
  + Вторая ОС должна запускаться из доступной для модификации памяти и позволять пользователю выполнять установку/изменение/удаление пользовательского ПО. В этой ОС должно быть установлено СЗИ, позволяющее, как минимум, выполнять процедуры разграничения доступа и контроля целостности.

Перечисленные выше ключевые особенности МКТ позволяют решить ряд проблем при работе с криптовалютными кошельками:

1. Офлайн подпись транзакций: если в качестве офлайн компьютера использовать мкт в защищенном 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.
2. Использование U2F для подписи транзакций: используя защищенную ОС МКТ в качестве доверенной среды, мы можем быть уверены, что на входе и выходе устройства будет именно та транзакция, которая нам нужна
3. Хранение ключей: если устанавливать кошелек в защищенную ОС МКТ, то при создании аккаунта в этом кошельке, можно выбрать опцию сохранения закрытых ключей в постоянную память, и не бояться, что эти ключи будут скомпрометированы.

Кроме того особенности архитектуры МКТ позволяют часть функционала кошелька исполнять в защищенной среде, и часть - в не защищенной. Как именно разбить функции на эти части, и что это даст - будет подробно рассмотрено в следующих разделах.

Теперь, когда все имеющие технологиии изучены, проблемы выявлены, и выявлены приблизительные способы их решения - перейдем непосредственно к проектированию мультивалютного холодного кошелька на базе на платформе МКТ.

# Проектирование системы

Рассмотрим подробно реализацию мультивалютного кошелька на базе МКТ:

* Распишем требования к проектируемой системе: ее составляющие и необходимый функционал
* Распишем, как именно использование МКТ позволяет избавиться от известных проблем и уязвимостей
* Распишем систему в целом

## Требования, предъявляемые к системе

В предыдущих разделах были исследованы имеющиеся на рынке решения и продукты, изучены известные недостатки и уязвимости, а также положительные аспекты рассмотренных технологий. В конце каждого раздела был сформулирован вывод, в котором были выделены методы и технологии, которые целесообразно и которые не целесообразно применять в рамках данной работы. На основе этих данных сформулируем все необходимые требования к проектируемой системе:

* **Безопасность** 
  + Поддержка двухфакторной аутентификации с помощью U2F
  + Возможность хранить ключи в защищенной ОС постоянной памяти в зашифрованном на пользовательском пароле виде для подписи транзакций
  + Учет государственных стандартов криптографии (описанных в документах, обозначенных на схеме ниже), при разработке своего программного обеспечения



(Рис. 10) Схема государственных стандартов криптографии в Российской Федерации

* **Работа с ключами**
  + Поддержка технологии HD wallet, то есть поддержка стандарта BIP32
  + Поддержка кошельков с мультиподписью, то есть поддержка стандарта BIP45
  + Поддержка развертывания ключей из мнемонической фразы, то есть поддержка стандарта BIP39
  + Поддержка фрагментации ключа, то есть порогового разделения секрета
* **Блокчейн составляющая и мультивалютность**
  + Поддержка мультивалютности (от 2 видов монет и выше), то есть поддержка стандарта BIP44
  + Возможность хранения токенов стандарта ERC20
  + Возможность просматривать блокчейн для каждой из криптовалют
  + Возможность совершать транзакции для каждой из криптовалют
* **Отказоустойчивость и Доступность**
  + Возможность создавать резервные копии ключей
  + Возможность восстановления ключей из мнемонической фразы, то есть поддержка стандарта BIP39
* **Экономическая составляющая:**
  + Отсутствие монетизации
  + Отсутствие привязки к фиатным деньгам
  + Отсутствие комиссий кошелька за операции просмотра и создания транзакций
  + Отсутствие внутреннего токена кошелька
  + Отсутствие возможности конвертации валюты
* **Имплементация в МКТ**
  + Проведение потенциально опасных операций и исполнение потенциально уязвимых функций только, когда МКТ находится в безопасном read only режиме
* **Фукнциональная составляющая кошелька**
  + Возможность пользоваться функционалом кошелька (чтение блокчейна; формирование, подпись и отправление транзакций) без создания аккаунта в кошельке
  + Возможность создать аккаунт в кошельке, в котором будут храниться все ключевые пары пользователя и история транзакций
  + При создании аккаунта должна быть возможность выбрать:
    - Создать стандартный кошелек
    - Создать кошелек с двухфакторной аутентификацией
    - Создать кошелек с мультиподписью
  + При работе с ключевым хранилищем должна быть возможность выбрать:
    - Создать новый seed и новую мнемоническую фразу
    - Использовать уже существующий seed или мнемоническую фразу
    - Использовать уже созданный мастер ключ
    - Использовать U2F устройство
  + Возможность сформировать транзакцию, но не подписывать ее
  + Возможность экспорта транзакций на съемный носитель
  + Возможность импорта транзакций с съемного носителя

Спроектируем систему, удовлетворяющую сформулированным требованиям.

## Использование особенностей МКТ для устранения известных уязвимостей

МКТ позволяет работать с двумя ОС - защищенной и не защищенной. Чтобы эффективно использовать эту опцию, необходимо понять - какую часть функционала кошелька необходимо исполнять в каждой из ОС. Для этого введем два класса операций:

1. **Операции, которые необходимо проводить в защищенной (RO) ОС**
2. **Операции, которые необходимо проводить в не защищенной (RW) ОС**

К первому классу отнесем все функции, где так или иначе задействованы закрытые ключи или персональные данные пользователя, так как эти данные являются наиболее ценными и как следствие наиболее вероятными целями для хакеров, и поэтому для этих функций необходимо обеспечить максимально возможную безопасную среду исполнения.

Ко второму классу отнесем те функции, для исполнения которых необходима запись данных в постоянную память (так как в этом классе применяется read and write ОС), а также необходим интернет (соединение с которым повышает вероятность атаки на систему, а поэтому не желательно оперировать ценными данными в этой ОС).

Проведем следующий анализ функционала:

* опишем каждую функцию;
* укажем - нужен ли интернет для ее корректной работы;
* Укажем - есть ли потенциальные уязвимости (если применять функцию на обычном компьютере) и какие;
* на основе предыдущих трех пунктов решим - к какому классу причислить эту функцию.

Таблица 1

Сравнительный анализ функционала кошелька

| Название функции | Описание | Требуется ли интернет? | Есть ли потенциальные уязвимости? | Класс функции |
| --- | --- | --- | --- | --- |
| 1. Генерация ключей, мастер ключа или мнемонической фразы | 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 и по протоколу BIP39 этот seed будет представлен в виде мнемонической фразы. | Да | Если среда и приложение доверенные - то нет. | Класс 1 |
| 1. Развертывание ключей из мастер ключа или мнемонической фразы | Пользователь вводит ключ или мнемоническую фразу, а кошелек на основе введеных данных выдает список всех ключевых пар пользователя, после чего ими можно пользоваться. | Нет | Если среда и приложение доверенные - то нет. | Класс 1 |
| 1. Создание нового аккаунта в кошельке | После генерации и развертывания ключей, будет создан файл вашего акканута в кошельке, в этом файле будет записано, что за данным пользователем закреплены соответствующие открытые ключи, и этот же файл будет хранить историю операций. Создавать такой файл или нет- пользователь может выбрать при установке кошелька . | Нет | Хранить историю операций и открытые ключи в постоянной памяти компьютера - это удобно, но менее безопасно. Поэтому хранение подобных данных необходимо разрешат только в защищенном режиме работы ОС, так как если среда недоверенная, то при компрометации такого файла злоумышленник может узнать, например, что определенный набор открытых ключей принадлежит одному человеку. | Класс 1 |
| 1. Чтение баланса из скачанного блокчейна | Идет чтение из блокчейна, расположенного в постоянной памяти. По завершению операции мы получаем баланс адреса на момент последнего обновления блокчейна. | Нет | Данную операцию необходимо проводить только в незащищенной ОС, так как блокчейн обновляется через интернет. Если исключить вероятность исполнения вредоносного кода, то сама операция чтения - не является уязвимой | Класс 2 |
| 1. Скачивание обновление блокчейна | Обновление блокчейна, для последующего совершения транзакции или чтения баланса аккаунта | Да | Из интернета в течении продолжительного времени качается большой объем данных (в случае, если блокчейн не обновлялся достаточно давно), поэтому есть вероятность закачки вредоносного кода. Данную операцию необходимо выполнять в незащищенной ОС | Класс 2 |
| 1. 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 мы получаем баланс аккаунта на момент последнего обновления блокчейна | Да | 1. За время соединения с интернетом в систему может попасть вредоносный код 2. Фишинг атака - есть вероятность чтения данных из ложных источников   Данную операцию необходимо выполнять в незащищенной ОС.  Вторая уязвимость зависит от надёжности сетевого соединения и сервера, к которому идет обращение. Чтобы избежать подобной атаки проще всего проверять баланс используя сторонний компьютер, если с его помощью зайти на официальный сайт данного блокчейна и по открытом ключу проверить свой баланс | Класс 2 |
| 1. Создание транзакции | Пользователем водятся 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 | Нет | Есть возможность того, что злоумышленник может подменить данные в транзакции на ложные. Учитывая что перед этим этапом, был этап обновления блокчейна - то в системе вполне мог оказаться вредоносный код.  Данную операцию необходимо выполнять в незащищенной ОС. | Класс 2 |
| 1. Подпись транзакции через интерфейс кошелька | Чтобы подписать транзакцию, пользователю необходимо ввести закрытый ключ 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 | нет | 1. Подмена транзакции на ложные данные 2. Компрометация закрытого ключа   Данную операция необходимо проводить только в защищенной ОС | Класс 1 |
| 1. Трансляция транзакции в блокчейн сеть | Подписанная транзакция посылается в сеть, на ближайший доступный узел. | да | 1. Злоумышленник может послать в сеть подписанную в предыдущем пункте транзакцию с ложными данными 2. Остальные атаки зависят от надежности канала связи и узлов, на которые идет распространение информации   Данную операцию необходимо выполнять в незащищенной ОС. | Класс 2 |
| 1. Аутентификация через U2F | В случае если был создан кошелек с двухфакторной аутентификацией, о для входа в свой аккаунт пользователь должен вставить в компьютер U2F устройство | Нет | Если среда и приложение доверенные - то нет. | Класс 1 |
| 1. Подпись транзакции через U2F | 1. На вход в U2F устройство посылается сформированная и не подписанная транзакция 2. U2F устройство посылает в компьютер подписанную транзакцию | Нет | На вход устройству может послаться транзакция с фальшивыми данными или с вредоносным кодом. | Класс 1 |
| 1. Хранение ключей | Закрытые ключи хранятся на жестком диске в зашифрованном виде | нет | Ключи лучше не хранить в постоянной памяти, но если возникает такая необходимость- то делать это можно только в защищенной ОС. | Класс 1 |

По результатам анализа разобьем операции на два класса:

* **Операции, которые необходимо проводить в защищенной (RO) ОС** 
  + Генерация ключей, мастер ключа или мнемонической фразы
  + Развертывание ключей из мастер ключа или мнемонической фразы
  + Создание нового аккаунта в кошельке
  + Подпись транзакции через интерфейс кошелька
  + Подпись транзакции через U2F
  + Хранение ключей в постоянной памяти
  + Аутентификация через U2F
* **Операции, которые необходимо проводить в не защищенной (RW) ОС**
  + Чтение баланса из скачанного блокчейна
  + Скачивание и обновление блокчейна
  + Чтение баланса из блокчейна в интернете
  + Создание транзакции
  + Трансляция транзакции в блокчейн сеть

Выполнение этих операций в соответствующих ОС поможет минимизировать риски компрометации.

## Анализ рынка

Как уже было показано выше, разделение функционала кошелька на две части и исполнение каждой части в своей ОС значительно поможет минимизировать риски компрометации. Для такого разделения имеет смысл использовать уже имеющиеся на рынке программные решения, то есть уже созданные мультивалютные кошельки, поддерживающие функционал, описанный в таблице в предыдущем разделе, а МКТ позиционировать как удобное решение, предлагающее две ОС, не имеющие доступа друг к другу на аппаратном уровне, одна из которых (read only) предназначена для наиболее безопасной работы с закрытыми ключами и непосредственно подписи транзакций, а вторая (read and write) для выхода в интернет, обновления блокчейна и формирования транзакций без подписи, для дальнейшего импорта на защищенную ОС.

По сути вся итоговая концепция - это модифицированная модель проведения офлайн транзакций (описанная в разделе «Офлайн подпись транзакций»), где в роли второго компьютера выступает защищенная ОС в МКТ, в роли первого компьютера - не защищенная ОС, и в роли программного обеспечения выступает один из тех мультивалютных кошельков, которые будут рекомендованы к использованию в данной системе. Этот список кошельков будет составлен ниже, после анализа.

Считаю такое решение целесообразным, так как создание собственного программного обеспечения, его написание, тестирование, отладка, учет программных, логических и математических нюансов при проектировании - это трудозатратный, времязатратный и дорогой процесс, по результатам которого будет создан аналог уже имеющихся на рынке решений от конкурентов, не несущий в себе научной новизны.

Необходимо понять, какие мультивалютные кошельки, из уже имеющихся на рынке, подойдут лучше всего для имплементации в МКТ. Для этого проведем сравнительный анализ существующих десктопных решений по следующим критериям:

* Поддержка Linux
* Поддержка Windows
* Поддержка HD Wallet
* Поддержка мультиподписи и порогового разделения секрета
* Поддержка подписи с помощью U2F устройства
* Поддержка необходимых для данной работы протоколов семейства BIP ( 32, 39, 43, 44, 45)
* Поддержка процедуры офлайн подписи транзакций (то есть возможность сформировать транзакцию, и не подписывая, импортировать ее на съемный носитель)
* Количество поддерживаемых криптовалют
* Поддержка токенов стандарта ERC20
* Создание резервных копий
* Поддержка двухфакторной аутентификации
* Прочее (обзоры от аналитиков, наличие зафиксированных уязвимостей, случаи взлома и так далее)

## 

Таблица 2

Сравнительный анализ холодных мультивалютных кошельков

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название кошелька | Linux | Windows | HD Wallet | мультиподпись | U2F | BIP | офлайн подпись транзакций | Количество поддерживаемых криптовалют | ERC20 | Создание резервных копий | Двухфакторная аутентификация | Прочее |
| Jaxx [17] | **+** | **+** | **+** | **+** | **+** | **+** | **-** | **57** | **+** | **+** | **+** | Кошелек не поддерживает офлайн транзакции |
| Atomic Wallet [18] | **+** | **+** | **-** | **-** | **-** | **+** | **-** | **84** | **+** | **-** | **+** | Представляет собой кошелек для взаимодействия с биржей |
| Exodus [19] | **+** | **+** | **+** | **-** | **+** | **+** | **-** | **16** | **+** | **+** | **-** | Хоть это и весьма популярный кошелек, к сожалению - его функционала не хватает для целей этой работы |
| Electrum [20] | **+** | **+** | **+** | **+** | **+** | **+** | **+** | **Серия из 5 одновалютных кошельков** | **+** | **+** | **+** | Один из самый старых и проработанных «тонких» кошельков, который требует всего 100МБ данных при скачивании блокчейна вместо 9ГБ |
| Coinomi [21] | **-** | **-** | **+** | **-** | **-** | **+** | **-** | **93** | **-** | **+** | **-** | Только для телефонов. Через несколько месяцев разработчики обещают выпустить декстопную версию |
| Coinbase [22] | **-** | **-** | **-** | **-** | **-** | **+** | **-** | **4** | **+** | **-** | **+** | Только для телефонов. Кошелек который модифицирован в криптобиржу |
| Green address [23] | **-** | **+** | **+** | **+** | **+** | **+** | **+** | **1** | **-** | **+** | **+** | Поддерживает только работу с биткойн |

Анализ показал, что лучше всего по функционалу подходит кошелек Electrum. Однако специфика кошельков этого семейства заключается в том, что для каждой валюты был сделан свой одновалютный кошелек, при этом исходный код кошельков отличается минимально (отличие заключается в семантике составления транзакций и алгоритмах генерации ключевых пар).

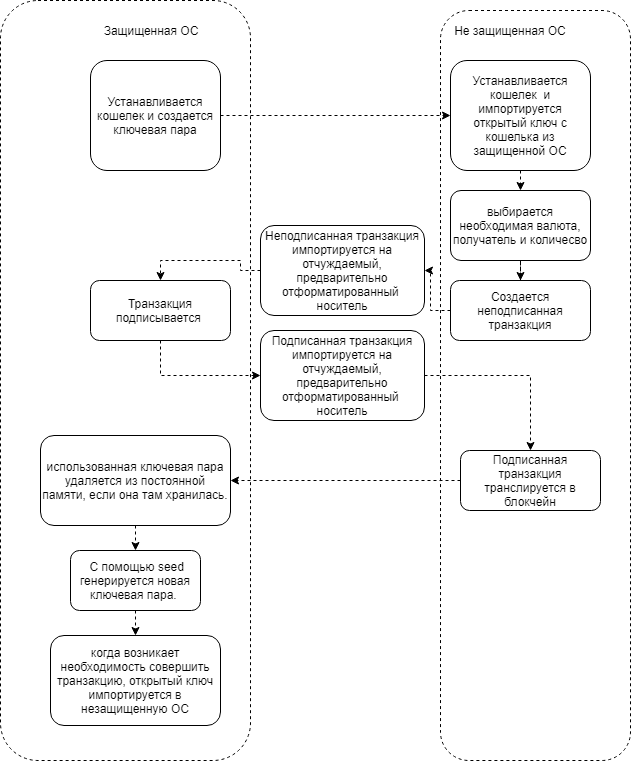
Electrum написан на Python и при этом является открытым программным обеспечением [24] - то есть программном обеспечением с открытым для модификации исходным кодом, и следовательно при разработке своего мультивалютного кошелька можно использовать и модифицировать этот исходный код. Соответственно для создания мультивалютного кошелька необходимо агрегировать семантику из уже созданных одновалютных кошельков на основе кода из семейства Electrum, а также добавить семантику тех криптовалют, которые пока не были реализованы в рамках этого семейства. Объем работ, зависит от количества валют, которые должен поддерживать проектируемый кошелек. При разработке для добавлении новой криптовалюты необходимо не просто добавить дополнительный класс в исходный код, но также учесть потенциальные уязвимости и нюансы работы каждой из блокчейн систем.

Подведем итог: в этом разделе были исследованы существующие на рынке холодные мультивалютные кошельки. Чтобы выбрать кошелек, наиболее подходящий для целей работы, был проведен сравнительный анализ, где критериями сравнения были необходимые технологии и функционал, рассмотренные подробно в предыдущих разделах. На основе анализа был выбран кошелек Electrum, являющийся проектом с открытым для модификации кодом. Кошелек уже поддерживает весь необходимый функционал, кроме возможности выбора валюты, так как каждой из кошельков этого семейства является одновалютным. Добавление этой возможности в исходный код, наращивание количества используемых криптовалют, и разделение на программном уровне функционала на два класса, описанных в предыдущем разделе - эти работы необходимо выполнить в рамках дипломной работы, следующей после данной научно исследовательской работы.

Теперь подытожим всю информацию и опишем модель работы системы на логическом уровне.

## Сценарий работы системы

Перейдем непосредственно к описанию работы системы.



(Рис. 11) Схема работы проектируемой системы

Опишем работу системы на верхне логическом уровне. Начало работы с мультивалютным кошельком выглядит следующим образом:

* Устанавливаем программное обеспечение на обе ОС в МКТ.
* Заходим в защищенную ОС и создаем кошелек:
  + **Запуск приложения, после которого программа предложит «Создать новый» или «Восстановить существующий» кошелек:** Для создания нужно просто выбрать соответствующую кнопку. Для восстановления потребуется ввести seed, мастер ключ или мнемоническую фразу, которую пользователь должен был записать при создании кошелька.
  + **Генерация мнемо-фразы:** Мнемо-фразу кошелек генерирует самостоятельно по протоколу BIP39, и пользователю рекомендуется сразу записать ее на бумажный носитель. Делается это для того, чтобы в случае утери ключа, у пользователя была возможность восстановления кошелька.
  + **Создание пароля к аккаунту в кошельке:** Если пользователь решил создать аккаунт в кошельке и хранить свои ключи и данные и историю операций в постоянной памяти, то ему необходимо придумать пароль от аккаунта, включив в него любые буквы, цифры и прочие символы. Вводить его понадобится при каждом входе в программу, и он обезопасит деньги пользователя в случае кражи или потери устройства, не позволяя стороннему лицу войти в кошелек. Как альтернативу, для аутентификации при входе в аккаунт кошелька вместо пароля моно использовать U2F устройство.
  + **Выбор необходимых криптовалют и адресов:** Добавлять и удалять выбранные криптовалюты можно будет позже в настройках. Благодаря технологии HD Wallet ля каждой криптовалюты кошелек может сгенерировать любое количество адресов, что и происходит по умолчанию при возвращении сдачи: остаток денег после транзакции возвращается на новый адрес.
  + **Создание резервной копии:** Программа должна предлагать создать резервную копию ключа несколькими способами - предложить запомнить мнемоническую фразу, предложить распечатать в виде QR кода мастер ключ, или предложить сделать копию файла, в котором в зашифрованном виде лежат закрытые ключи.
  + **Выгрузка открытых ключей: импорт необходимых открытых ключей в виде QR кода для последующего применения в не защищенной ОС**
* Заходим в незащищенную ОС и импортируем созданные в предыдущем пункте открытые ключи в кошелек
* Когда возникает необходимость совершить транзакцию с одного из адресов, то система в незащищенной ОС должна:
  + Обновить блокчейн
  + Сформировать транзакцию
  + Экспортировать сформированную транзакцию на отчуждаемый носитель
* Сформированная в предыдущем пункте транзакция на отчуждаемом носителе импортируется в защищенную ОС
* В защищенной ОС транзакция подписывается одним из вариантов:
  + С помощью U2F устройства
  + С помощью закрытого ключа, хранимого в зашифрованном виде в файле в постоянной памяти
  + С помощью закрытого ключа, введенного пользователем вручную в соответствующее поле интерфейса
* После этого подписанная транзакция экспортируется через отчуждаемый носитель обратно в незащищенную ОС
* В незащищенной ОС подписанная транзакция транслируется в сеть

Таким образом мы получаем систему, в которой все операции, производимые с закрытыми ключами, проводятся в отельной защищенной операционной системе с постоянной памятью доступной только чтения. Благодаря технологии иерархически детерминированного кошелька все ключевые пары являются одноразовыми, то есть после каждой транзакции, закрытый ключ больше не используется, при чем даже если злоумышленник перехватит все использованные ключи, он не сможет вычислить остальные. Спроектированный кошелек также обладает функционалом, аналогичным конкурентам: возможность резервного копирования и восстановления, возможность хранить данные в одном аккаунт (хоть это и снижает надежность хранения данных, но повышает удобство пользования).

# Заключение

В ходе данной работы по проектированию холодного мультивалютного кошелька на платформе МКТ были рассмотрены существующие решения на рынке, изучены их преимущества и недостатки. На основе этих данных были сформулированы требования, которым должна удовлетворять система. Каждый блок этих критериев был проанализирован отдельно, и в каждом было выбрано определенное решение. Итогом этой работы является эскизный проект системы, расписанный на логическом (без углубления в код и машинно-математическую составляющую) уровне. В рамках продолжения работы над проектом дальнейшие шаги вижу в составлении ТЗ на разработку, и непосредственно разработку, отладку и тестирование проекта.

# Список литературы

1. Хеширование [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://ru.wikipedia.org/wiki/Хеширование#Криптографические\_хеш-функции
2. Файлообменные сети P2P: основные принципы, протоколы, безопасность [Электронный ресурс] /. — Электрон. журн. — Режим доступа: http://ccc.ru/magazine/depot/06\_11/read.html?0302.htm
3. Универсальная двухфакторная аутентификация [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://ru.wikipedia.org/wiki/Универсальная\_двухфакторная\_аутентификация, свободный
4. ERC-20 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://en.wikipedia.org/wiki/ERC-20, свободный
5. Конявский, В.А. Компьютер с «вирусным иммунитетом» [Электронный ресурс] / В.А. Конявский. — Электрон. текстовые дан. — Режим доступа: http://www.okbsapr.ru/konyavskiy\_2015\_2.html, свободный
6. Блокчейн [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://ru.wikipedia.org/wiki/Блокчейн, свободный
7. Cold storage [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://en.bitcoinwiki.org/wiki/Cold\_storage, свободный
8. Иерархическая генерация ключей [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://habr.com/company/distributedlab/blog/413627/, свободный
9. bip-0032 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, свободный
10. bip-0039 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki, свободный
11. bip-0043 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0043.mediawiki, свободный
12. bip-0044 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki, свободный
13. bip-0045 [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://github.com/bitcoin/bips/blob/master/bip-0045.mediawiki, свободный
14. Владимиров, С.М. Криптографические методы защиты информации /С.М. Владимиров, Э.М. Габидуллин, А.И. Колыбельников, А.С. Кшевецкий. — 2-е изд., перераб. и доп. — 2016. —
15. FIDO U2F — Универсальная Двухфакторная Аутентификация. Введение [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: https://habr.com/post/305508/, свободный
16. Работа с одного рабочего места в двух разных контурах защищенности [Электронный ресурс] /. — Электрон. текстовые дан. — Режим доступа: http://www.okbsapr.ru/sol17.html, свободный
17. Jaxx support [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://support.decentral.ca/hc/en-us/categories/202644577-Jaxx>
18. Atomic Wallet [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://docviewer.yandex.ru/view/85466828/?\*=VHTSFAAvclJO3mm%%3D%3D&page=1&lang=en
19. Exodus Common questions and support documentation [Электронный ресурс] /. — Электрон. журн. — Режим доступа: <https://support.exodus.io>
20. Electrum Documentation [Электронный ресурс] /. — Электрон. журн. — Режим доступа: http://docs.electrum.org/en/latest/index.html
21. Coinomi Knowledge base [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://coinomi.freshdesk.com/support/home
22. Coinbase FAQ [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://support.coinbase.com
23. GreenAddress FAQ [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://greenaddress.it/ru/faq.html
24. Electrum Documentation [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://github.com/spesmilo/electrum