

Содержание

1	Введение	2
2	Основная часть	3
2.1	Протокол Диффи-Хеллмана	3
2.2	Эфемерные ключи	4
2.3	Протокол Диффи-Хеллмана на эллиптических кривых	4
2.4	X3DH	6
2.5	Описание атак на протокол Диффи-Хеллмана	9
2.6	Сеть Ethereum	10
2.7	Смарт-контракты	11
3	Практическая часть	13
3.1	BCB-X3DH	13
3.2	Реализация	14
3.3	Недостатки	16
4	Вывод	18
	Источники	19

1 Введение

Главных проблем алгоритмов симметричного шифрования является процесс передачи ключа шифрования второй стороне. Для передачи ключа необходимо использовать защищенный канал, установление которого может оказаться крайне трудной задачей.

Одним из вариантов решения проблемы обмена криптографическими ключами по незащищенному каналу стал протокол Диффи-Хеллмана (Diffie-Hellman key exchange), предложенный Уитфилдом Диффи и Мартином Хеллманом в 1976 году [1].

Статья Сатоши Накамото [2], описывающая децентрализованные цифровые финансы (Bitcoin), стала отправной точкой для новых возможностей в интернете. Выпущенная вслед в 2014 году статья Виталика Бутерина [3] расширяет идею биткойна от “электронных денег” до “платформы для создания децентрализованных онлайн сервисов”.

Целью данной работы является реализация алгоритма X3DH [4] на блокчейне Ethereum с использованием смарт-контрактов.

Задачи работы:

1. Рассмотрение принципа выработки общего секрета;
2. Алгоритмическая сложность атаки;
3. Изучение модификации алгоритма — X3DH;
4. Рассмотрение принципов работы сети Ethereum;
5. Устройство смарт-контрактов;
6. Рассмотрение статьи VCB-X3DH;
7. Практическая реализация алгоритма X3DH на локальной сети Ethereum.

2 Основная часть

2.1 Протокол Диффи-Хеллмана

Протокол выработки общего секрета был предложен в 1976 году Уитфилдом Диффи и Мартином Хеллманом. Особенность протокола заключается в том, что две стороны (Алиса и Боб) могут установить общий секрет, используя открытый незащищенный канал связи.

Описание протокола [1]:

1. Алиса и Боб договариваются о выборе большого простого числа p и генератора g .
2. Алиса выбирает $\alpha \in \mathbb{Z}$ такое, что $1 < \alpha \leq p - 1$. Затем Алиса вычисляет $A = g^\alpha \pmod{p}$. Значение α является закрытым ключом; значение A является открытым ключом.
3. Боб выбирает $\beta \in \mathbb{Z}$ такое, что $1 < \beta \leq p - 1$. Затем Боб вычисляет $B = g^\beta \pmod{p}$. Значение β является закрытым ключом; значение B является открытым ключом.
4. Алиса получает открытый ключ Боба B и вычисляет

$$S = B^\alpha = (g^\beta)^\alpha = g^{\beta\alpha} \pmod{p}.$$

5. Боб получает открытый ключ Алисы A и вычисляет

$$S = A^\beta = (g^\alpha)^\beta = g^{\alpha\beta} \pmod{p}.$$

6. Алиса и Боб получили общий секретный ключ

$$S = g^{\alpha\beta} = g^{\beta\alpha} \pmod{p}.$$

Протокол Диффи-Хеллмана решает одну из главных проблем в криптографии — безопасный обмен ключами. Сложность, связанная с симметричными

алгоритмами шифрования, заключается в том, чтобы каждая из сторон имела ключ, который будет использоваться для шифрования и расшифрования. Для этого необходимо осуществить передачу чувствительной информации (ключа) по незащищенному каналу. Рассматриваемый протокол использует предположение о том, что по известным g и g^x вычисление x является сложной задачей. Подробнее алгоритмическая сложность атаки будет рассмотрена в пункте 2.5.

2.2 Эфемерные ключи

Важно заметить, что стандартный алгоритм Диффи-Хеллмана не обеспечивает “упреждающую секретность” (“forward secrecy”). Под упреждающей секретностью понимается свойство протокола согласования ключа, которое не позволит злоумышленнику расшифровать предыдущие сообщения, зная долговременный закрытый ключ.

На практике, если Ева (злоумышленник) получит закрытый ключ Алисы α , то она сможет вычислить общий секрет Алисы и Боба $S = g^{\beta\alpha}$, а значит сможет расшифровать их сообщения.

Использование эфемерного ключа в протоколе подразумевает то, что для каждой сессии будут генерироваться новые параметры α и g^α . Тем самым Ева, заполучив закрытый ключ Алисы, будет иметь доступ к сообщениям текущей сессии, но не к остальным.

2.3 Протокол Диффи-Хеллмана на эллиптических кривых

Одной из модификаций протокола Диффи-Хеллмана является протокол Диффи-Хеллмана на эллиптических кривых.

Эллиптической кривой E над полем F называется множество точек (x, y) , координаты которых принадлежат полю и удовлетворяют уравнению вида

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in F.$$

Если характеристика поля $p > 3$, то уравнение имеет вид

$$y^2 = x^3 + ax + b.$$

На практике используются кривые над конечными полями F_p , где p — простое число. Таким образом, кривая описывается тремя параметрами: a , b и p . Например, кривая *secp256k1* имеет следующий вид:

$$y^2 \equiv x^3 + 7 \pmod{2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 2^0}.$$

Основной операцией в рамках эллиптических кривых является сложение двух точек, принадлежащих кривой. Результат операции сложения — точка на кривой.

$$P = Q + G \in E \Leftrightarrow Q, G \in E.$$

Координаты точки $P = (x_P, y_P)$ определяются по следующим формулам:

$$\lambda \equiv \frac{y_G - y_Q}{x_G - x_Q} \pmod{p} \quad (1)$$

$$x_P \equiv \lambda^2 - x_Q - x_G \pmod{p} \quad (2)$$

$$y_P \equiv \lambda \cdot (x_Q - x_P) - y_Q \pmod{p}, \quad (3)$$

где $Q = (x_Q, y_Q)$, $G = (x_G, y_G)$.

Операция умножения точки G на скаляр k определяется как последовательное сложение точки G k раз.

$$P = k \cdot G = \underbrace{G + \dots + G}_k.$$

Опишем протокол Диффи-Хеллмана на эллиптических кривых:

1. Алиса и Боб договариваются о выборе эллиптической кривой E и точки $G \in E$ — генератор.
2. Алиса выбирает $\alpha \in \mathbb{Z}$ такое, что $1 < \alpha \leq p - 1$. Затем Алиса вычисляет $A \equiv \alpha \cdot G \pmod{p}$. Значение α является закрытым ключом; значение A является открытым ключом.

3. Боб выбирает $\beta \in \mathbb{Z}$ такое, что $1 < \beta \leq p - 1$. Затем Боб вычисляет $B \equiv \beta \cdot G \pmod{p}$. Значение β является закрытым ключом; значение B является открытым ключом.

4. Алиса получает открытый ключ Боба B и вычисляет

$$S \equiv \alpha \cdot B \equiv \alpha \cdot (\beta \cdot G) \equiv (\alpha \cdot \beta) \cdot G \pmod{p}.$$

5. Боб получает открытый ключ Алисы A и вычисляет

$$S \equiv \beta \cdot A \equiv \beta \cdot (\alpha \cdot G) \equiv (\beta \cdot \alpha) \cdot G \pmod{p}.$$

6. Алиса и Боб получили общий секретный ключ

$$S \equiv (\alpha \cdot \beta) \cdot G \equiv (\beta \cdot \alpha) \cdot G \pmod{p}.$$

Стойкость достигается за счет использования проблемы дискретного логарифмирования в эллиптических кривых: зная P и G , найти такое k , что $P = k \cdot G$.

2.4 X3DH

Одной из проблем протокола Диффи-Хеллмана является невозможность выработки ключа, когда одна из сторон не находится в сети (отсутствие асинхронности). Решение данной проблемы было предложено разработчиками мессенджера *Signal* в 2016 году — Extended Triple Diffie-Hellman (X3DH).

Идея алгоритма заключается в том, чтобы позволить Бобу загрузить некоторую информацию о себе на сервер, а Алисе — запросить информацию о Бобе с сервера. После получения этой информации Алиса вырабатывает секретный ключ и отправляет Бобу изначальное сообщение, зашифрованное секретным ключом.

Каждая из сторон вырабатывает следующие ключи, представленные в таблице 1. Каждому публичному ключу соответствует закрытый ключ. Для генерации ключей используется эллиптическая кривая *Curve25519* или *Curve448*.

Таблица 1: Ключи протокола X3DH

Название	Описание
IK_A	Идентифицирующий ключ Алисы
EK_A	Эфемерный ключ Алисы
IK_B	Идентифицирующий ключ Боба
SPK_B	Подписанный преключ Боба
OPK_B	Одноразовый ключ Боба

Идентифицирующие ключи IK_α являются постоянными и загружаются на сервер единожды. Ключ SPK меняется периодически, и необходим для того, чтобы подтвердить на стороне Алисы, что полученная информация о Бобе принадлежит действительно ему. Ключи OPK загружаются набором из n штук и используются один раз. Новый ключ EK генерируется при каждом запуске протокола.

Протокол реализуется в три этапа:

1. Боб загружает набор ключей $(IK_B, SPK_B, Sig, OPK_B)$ на сервер.
2. Алиса получает набор Боба и проверяет, что

$$Sig = \text{signature}(IK_B, SPK_B).$$

После успешной проверки подписи, Алиса генерирует секретный ключ и отправляет изначальное сообщение Бобу.

3. Боб получает сообщение Алисы и расшифровывает сообщение.

Рассмотрим принцип выработки общего секрета. Для этого определим функцию DH:

$$\text{DH}(A, B) = a \cdot B = b \cdot A = (a \cdot b) \cdot G,$$

где (a, A) — пара закрытый/открытый ключ Алисы;

(b, B) — пара закрытый/открытый ключ Боба;

G — генератор, который определяется спецификацией выбранной эллиптической кривой.

После запроса набора ключей Боба Алиса имеет следующие ключи:

$$ik_A, ek_A, IK_B, SPK_B, OPK_B,$$

где ik_A, ek_A — соответствующие закрытые ключи для IK_A, EK_A . Алиса вычисляет следующие значения:

$$\begin{aligned} DH_1 &= \text{DH}(IK_A, SPK_B) = ik_A \cdot SPK_B \\ DH_2 &= \text{DH}(EK_A, IK_B) = ek_A \cdot IK_B \\ DH_3 &= \text{DH}(EK_A, SPK_B) = ek_A \cdot SPK_B \\ DH_4 &= \text{DH}(EK_A, OPK_B) = ek_A \cdot OPK_B. \end{aligned}$$

Для выработки общего секретного ключа используется функция KDF:

$$SK = \text{KDF}(DH_1 \parallel DH_2 \parallel DH_3 \parallel DH_4).$$

На стороне Боба имеются следующие ключи:

$$IK_A, EK_A, ik_B, spk_B, oprk_B,$$

где $ik_B, spk_B, oprk_B$ — соответствующие закрытые ключи для IK_B, SPK_B, OPK_B . Боб вычисляет следующие значения

$$\begin{aligned} DH_1 &= \text{DH}(SPK_B, IK_A) = spk_B \cdot IK_A \\ DH_2 &= \text{DH}(IK_B, EK_A) = ik_B \cdot EK_A \\ DH_3 &= \text{DH}(SPK_B, EK_A) = spk_B \cdot EK_A \\ DH_4 &= \text{DH}(OPK_B, EK_A) = oprk_B \cdot EK_A \\ SK &= \text{KDF}(DH_1 \parallel DH_2 \parallel DH_3 \parallel DH_4). \end{aligned}$$

Таким образом, Алиса и Боб вырабатывают общий секретный ключ по незащищенному каналу.

Генерацию общего секрета можно представить в виде рисунка 2.1[4].

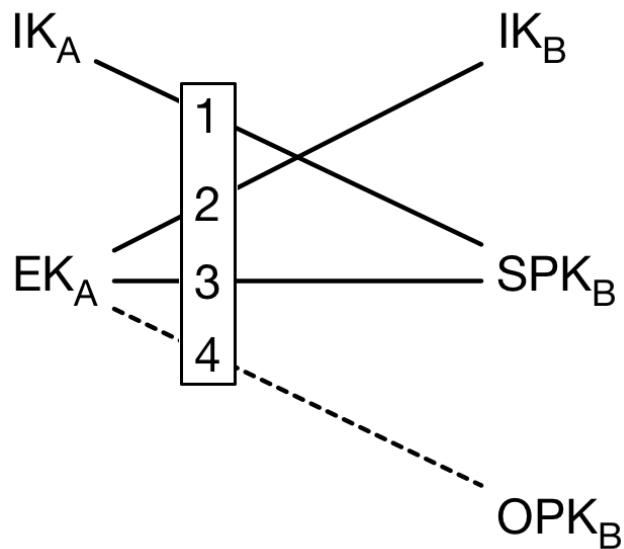


Рис. 2.1: Генерация секрета

2.5 Описание атак на протокол Диффи-Хеллмана

Стойкость протокола основывается на *проблеме Диффи-Хеллмана*: для заданного элемента g и значений g^x и g^y определить g^{xy} . На практике это означает, что Ева может читать канал Алисы и Боба и может увидеть их публичные ключи и генератор, о котором Алиса и Боб договорились.

Проблема Диффи-Хеллмана сводится к решению проблемы нахождения дискретного логарифма: по заданному g и g^x найти x . Проблема дискретного логарифмирования не имеет эффективного алгоритма решения. Поэтому для большого модуля p , его примитивного корня g и значения a нахождения x такого, что

$$g^x \equiv a \pmod{p}$$

является вычислительно сложной задачей. Для её решения необходимо перебирать x пока заданное сравнение не окажется истинным.

Однако, проблема дискретного логарифмирования остается сложной только для чисел имеющих размер больше 2048 бит. Атака *logjam* использует возможность серверов поддерживать 512 битный Диффи-Хеллман, перебор которого возможен за разумное время.

Атака *человек посередине (Man In The Middle)* возможна при использовании протокола Диффи-Хеллман за счет того, что этот протокол не подразумевает

взаимной аутентификации сторон. В данном случае Алиса и Боб будут обмениваться зашифрованными сообщениями, не подозревая, что на самом деле сообщения проходят через Еву.

Другой атакой, которой подвержен протокол Диффи-Хеллмана, является *отказ в обслуживании*. Атакующий может наводнить сеть поддельными запросами на обмен ключами, тем самым перегружая систему и предотвращая легитимные обмены.

В качестве варианта решения последних двух проблем в практической части данной работы рассматривается реализация алгоритма X3DH с помощью технологии блокчейн.

2.6 Сеть Ethereum

Ethereum — это децентрализованная блокчейн сеть, реализующая технологию смарт-контрактов. В отличие от блокчейна Bitcoin, блокчейн Ethereum является программируемым, что позволяет реализовывать на нем децентрализованные приложения.

Технология блокчейн была предложена Сатоши Накамото в 2009 году. Блокчейн представляет собой множество блоков, содержащих транзакции, которые имеют указатель на предыдущий блок. Революционным в статье Сатоши является предложенный механизм *proof-of-work*, использование которого исключает необходимость существования централизованного контролирующего органа. В PoW валидатор пытается найти хэш-значение от заголовка блока с заданным числом лидирующих нулей. После нахождения такого хэш-значения валидатор получает вознаграждение.

В отличие от Bitcoin, Ethereum использует механизм *proof-of-stake*. В PoS валидатор очередного блока выбирается случайно из числа тех, кто заложил 32 ЕТН, чтобы стать валидатором. Чем больше ЕТН было заложено, тем выше шанс стать валидатором. PoS не предполагает использование вычислительных ресурсов для нахождения хэш-значения, а поэтому является более энергоэффективным.

2.7 Смарт-контракты

Смарт-контракт — это программа, которая запускается на блокчейне Ethereum. Он представляет собой код и данные, которые имеют определенный адрес в сети. Смарт-контракт не контролируется никаким пользователем сети — он функционирует по той логике, которая была заложена в него разработчиком.

Выполнение смарт-контракта инициируется транзакцией. После валидации транзакции код смарт-контракта выполняется на виртуальной машине Ethereum (Ethereum Virtual Machine). Структура EVM представлена на рисунке 2.2. Осо-

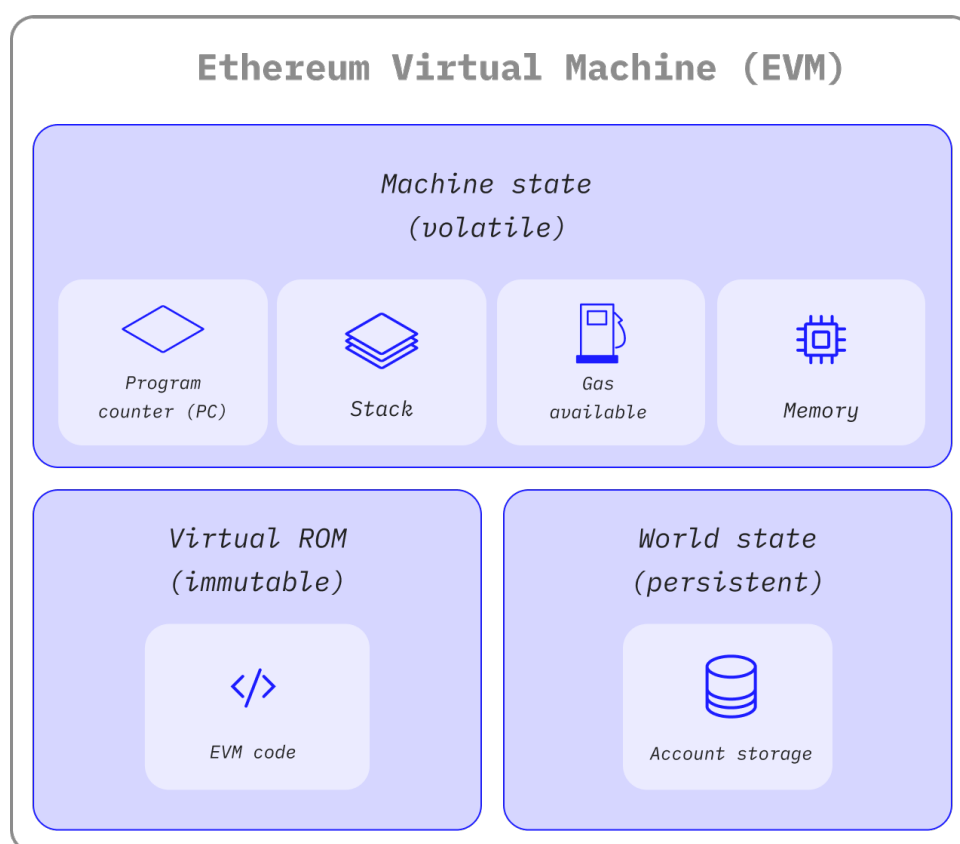


Рис. 2.2: Структура виртуальной машины Ethereum

бенность EVM заключается в том, что эта виртуальная машина выполняется на всех устройствах, которые выступают в качестве узла сети. Благодаря своей распределенной структуре, EVM обеспечивает безопасность всей сети. Кроме того, EVM гарантирует единообразное выполнение смарт-контрактов, что позволяет избежать разночтений и обеспечить надежность работы блокчейна.

Для написания смарт-контрактов используется высокоуровневый язык *Solidity*. Код, написанный на языке Solidity, транслируется в байт-код в соответствие со

спецификацией EVM [5].

Для того, чтобы разместить смарт-контракт в сети Ethereum, необходимо осуществить плату называемую *gas fee*. Газ необходим для обеспечения безопасности сети. Если злоумышленник решит наводнить сеть спам запросами, то за каждую транзакцию ему потребуется осуществить оплату.

Одним из главных недостатков, но одновременно и преимуществом, смарт-контрактов является их неизменяемость. Очевидность недостатка заключается в возможных ошибках, которые может совершить разработчик при написании кода. В 2016 году организация *The DAO* потеряла треть своих накоплений (53 миллиона долларов США) из-за ошибки в коде смарт-контракта. Однако никакой злоумышленник не сможет повлиять на корректно написанный смарт-контракт.

3 Практическая часть

Практическая часть данной работы опирается на описание алгоритма X3DH, предложенного в [4], а также на работу [6], в которой предложена реализация алгоритма X3DH с помощью технологии блокчейн.

Использование технологии блокчейн для алгоритма X3DH устраняет возможность атаки отказа в обслуживании. Для того, чтобы осуществить указанную атаку в стандартной реализации алгоритма, достаточно атаковать единственный сервер, который хранит ключи пользователей. Чтобы осуществить ту же атаку в сети Ethereum, необходимо обладать огромными ресурсами, чтобы захватить сеть, или требуется повлиять на десятки тысяч узлов сети, чтобы вывести их из строя. Сеть Ethereum не имеет единой точки отказа (single point of failure).

Авторы работы [6] видят преимущество в использовании блокчейн технологии для *интернета вещей (Interne of Things)*. Вычислительную нагрузку, связанную с выработкой ключей, можно переложить на плечи EVM, что позволяет устройствам всех мощностей осуществлять обмен ключами шифрования.

3.1 BCB-X3DH

Алгоритм BCB-X3DH использует сеть блокчейн для хранения информации о пользователях. В данном случае Боб загружает набор своих ключей как смарт-контракт. Алиса запрашивает у смарт-контракта ключи Боба; вырабатывает общий секрет. После чего она загружает информацию о себе и текст зашифрованный общим ключом как смарт-контракт. Боб запрашивает информацию об Алисе и вырабатывает общий секрет. Этапы алгоритма представлены на рисунке 3.1.

Отметим, что в данной работе генерация ключей выполняется на устройстве пользователя, а не в сети Ethereum.

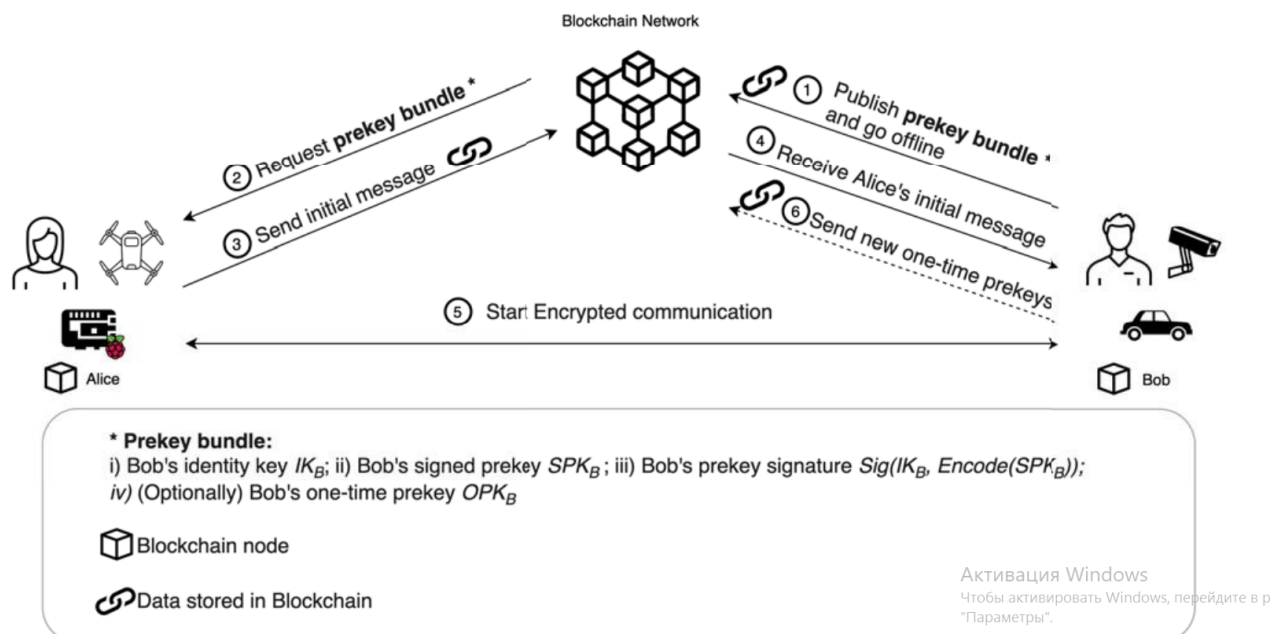


Рис. 3.1: Этапы алгоритма VCB-X3DH

3.2 Реализация

Для написания смарт-контрактов использовался язык Solidity версии 0.8.24¹. Локальная сеть Ethereum была развернута с помощью инструмента Hardhat версии 2.22.3². Логика обмена секретом была написана на языке TypeScript.

На стороне Боба выполняются следующие этапы:

1. Сгенерировать необходимые пары ключей.
2. Создать смарт-контракт и передать в конструктор открытые ключи и подпись.
3. Ожидание смарт-контракта от Алисы.
4. Выработать общий секрет.

На стороне Алисы выполняются следующие этапы:

1. Сгенерировать необходимые пары ключей.

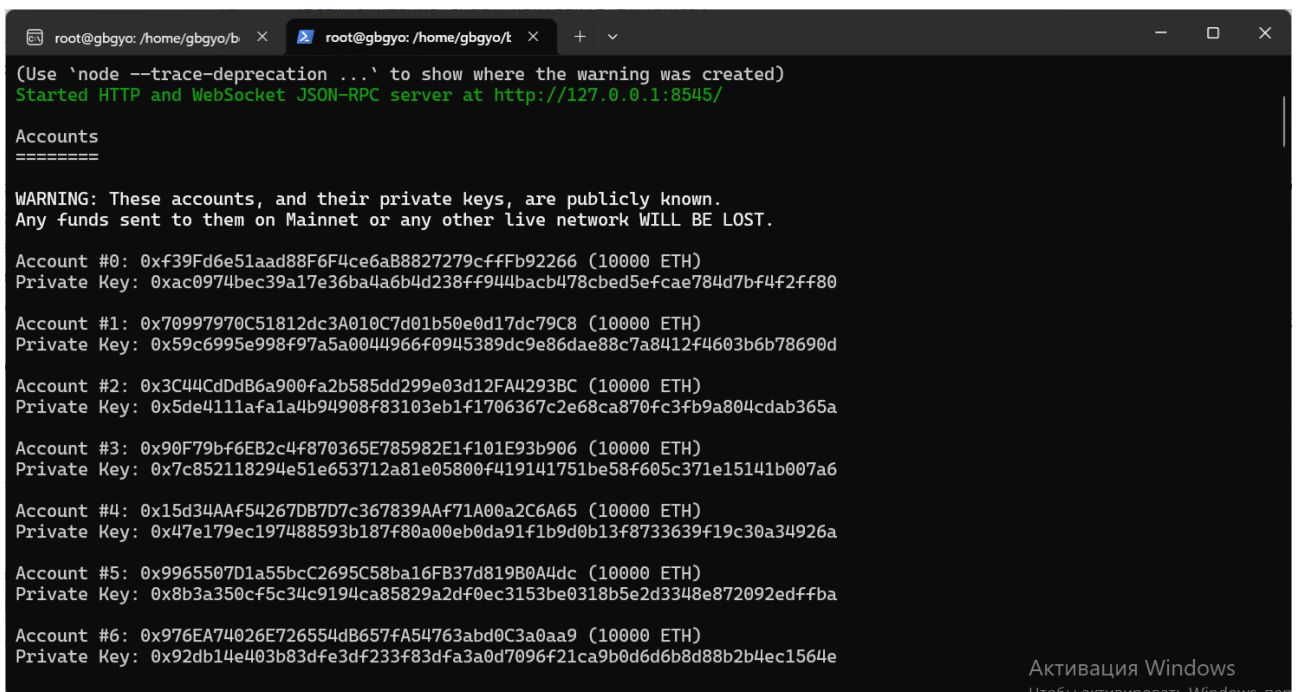
¹<https://github.com/ethereum/solidity/releases/tag/v0.8.24>

²<https://hardhat.org/>

2. Создать смарт-контракт и передать в конструктор открытые ключи и адрес смарт-контракта Боба.
3. Получить открытые ключи Боба.
4. Выработать общий секрет.

Для генерации ключей использовалась библиотека *noble-curve*³.

Чтобы протестировать обмен ключами, необходимо развернуть локальную сеть. Для этого используется инструмент Hardhat. Развернутая сеть имеет 20 адресов с 10000 ETH (рисунок 3.2).



```

root@gbgyo: /home/gbgyo/b x root@gbgyo: /home/gbgyo/t x + v
(Use `node --trace-deprecation ...` to show where the warning was created)
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
=====

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0xf39Fd6e51aad88F6F4ce6aB8827279cFfFb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3: 0x90F79b66f6EB2c4f870365E785982E1f101E93b906 (10000 ETH)
Private Key: 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6

Account #4: 0x15d34AAf54267DB7D7c36783AAf71A00a2C6A65 (10000 ETH)
Private Key: 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a

Account #5: 0x9965507D1a55bcC2695C58ba16FB37d819B0A4dc (10000 ETH)
Private Key: 0x8b3a350cf5c34c9194ca85829a2df0ec3153be0318b5e2d3348e872092edffba

Account #6: 0x976EA74026E726554dB657F54763abd0C3a0aa9 (10000 ETH)
Private Key: 0x92db14e403b83dfe3d3f233f83dfa3a0d7096f21ca9b0d6d6b8d88b2b4ec1564e

Активация Windows
Чтобы активировать Windows, перейдите на сайт windows.com
  
```

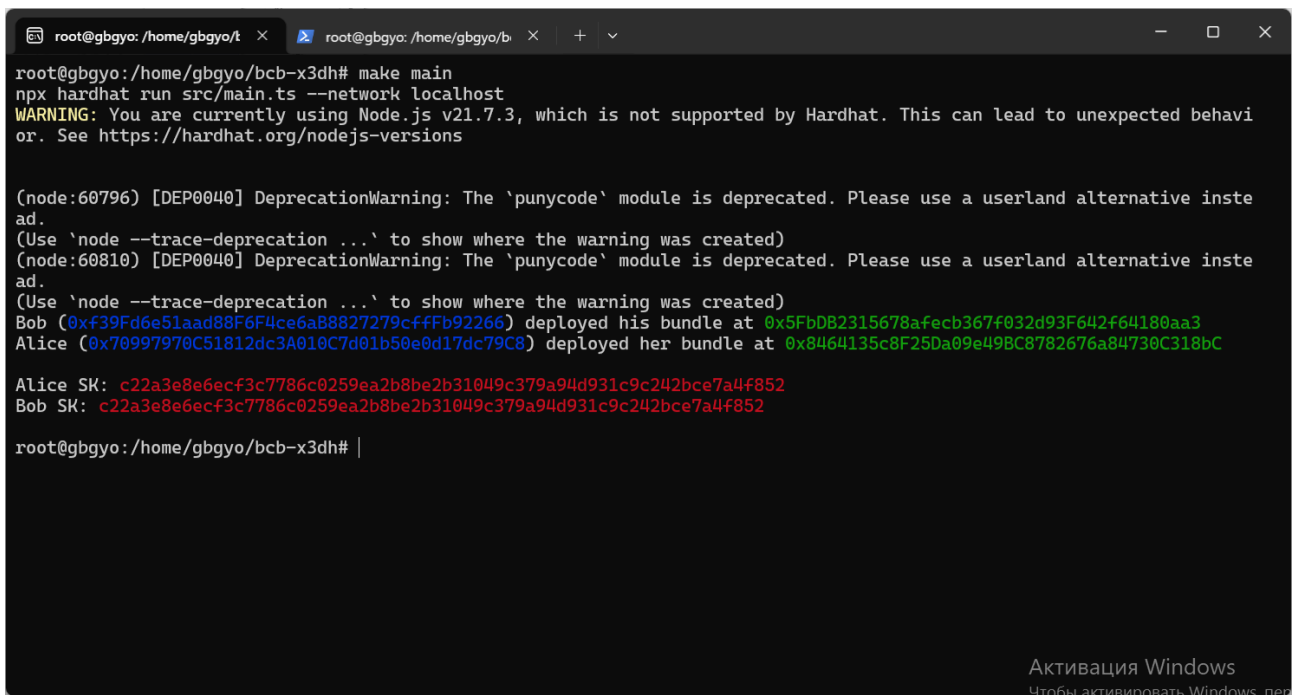
Рис. 3.2: Локальная сеть Ethereum

Обмен секретом будет осуществляться между первым (0xf39Fd6...) и вторым (0x709979...) адресами. Результат обмена представлен на рисунке 3.3

Инструмент Ethernal⁴ позволяет исследовать локальные сети: просматривать аккаунты, транзакции, токены и общую статистику. На рисунке 3.4 можно видеть, что с адреса Боба была создана транзакция на создание контракта (0xe86219...). Аналогичную транзакцию сделала Алиса (0x92276с...). Последняя транзакция (0x002d14...) направлена от Алисы к смарт-контракту Боба. В этой транзакции Алиса запрашивает ключи Боба.

³<https://github.com/paulmillr/noble-curves>

⁴<https://tryethernal.com/>



```
root@gbgyo: /home/gbgyo/bcb-x3dh# make main
npx hardhat run src/main.ts --network localhost
WARNING: You are currently using Node.js v21.7.3, which is not supported by Hardhat. This can lead to unexpected behavior. See https://hardhat.org/nodejs-versions

(node:60796) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:60810) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Bob (0xf39Fd6e51aad88F6F4ce6a88827279cFfFb92266) deployed his bundle at 0x5FbDB2315678afecb367f032d93F642f64180aa3
Alice (0x70997970C51812dc3A010C7d01b50e0d17dc79C8) deployed her bundle at 0x8464135c8F25Da09e49BC8782676a84730C318bC

Alice SK: c22a3e8e6ecf3c7786c0259ea2b8be2b31049c379a94d931c9c242bce7a4f852
Bob SK: c22a3e8e6ecf3c7786c0259ea2b8be2b31049c379a94d931c9c242bce7a4f852

root@gbgyo: /home/gbgyo/bcb-x3dh# |
```

Рис. 3.3: Обмен секретным ключом

Важно отметить следующую особенность Ethereum: транзакции на чтение не требуют валидации. Из-за особенности реализации алгоритма, всякий раз когда Алиса обращается к смарт-контракту Боба, она изменяет его состояние. Чтобы зафиксировать эти изменения, транзакция должна быть валидирована. Однако обращение Боба к контракту Алисы не изменяет состояние смарт-контракта последней. Поэтому транзакция на запрос Бобом смарт-контракта Алисы не отображается в Ethernal.

Из рисунка 3.4 можно видеть, что Боб потратил примерно 0.0087 ЕТН, что на момент написания работы равняется 26.78 долларам США. Алиса суммарно (на размещение своего контракта и доступ к контракту Боба) потратила примерно 0.00073 ЕТН или 2.25 доллара США.

3.3 Недостатки

Стоит отметить, что реализация алгоритма X3DN в сети Ethereum имеет ряд недостатков, обусловленных особенностью используемой технологии.

Первый недостаток — стоимость. Для того, чтобы разместить информацию о себе в сеть, необходимо совершить транзакцию. Для того, чтобы Алиса получила ключи Боба, необходимо совершить транзакцию. Для добавления транзак-

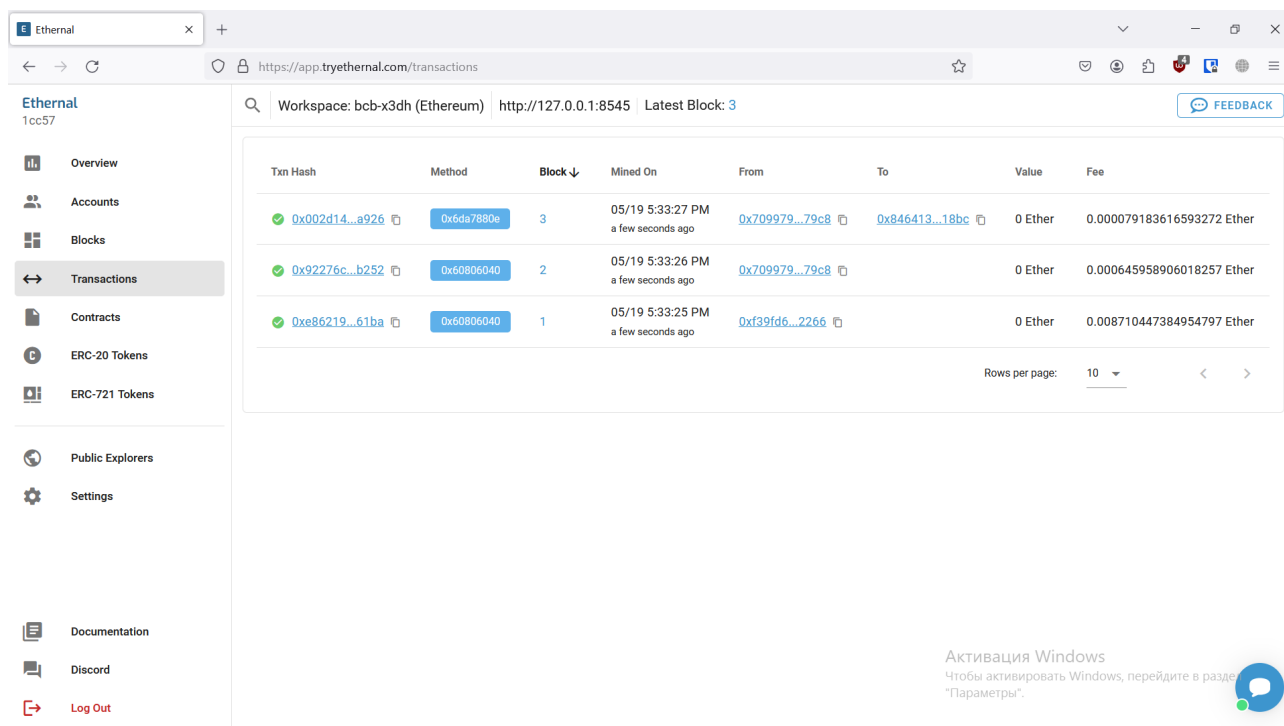


Рис. 3.4: Транзакции Боба и Алисы

ции в блок необходим газ, цена на который может варьироваться в зависимости от загруженности сети.

Второй недостаток — скорость. Каждая транзакция требует валидации перед тем, как она станет видна в сети. Так же как и предыдущий недостаток, скорость валидации зависит от загруженности сети.

4 Вывод

Алгоритм Диффи-Хеллмана по-прежнему остается важным компонентом безопасности в современном Интернете. Основываясь на проблеме нахождения дискретного логарифма, он обеспечивает защиту от подслушивания канала злоумышленником.

Применение новых технологий, таких как блокчейн, позволяет сделать алгоритм обмена ключами более отказоустойчивым. Различные улучшения алгоритма допускают его использование в современных мессенджерах.

Отметим, что предложенная в данной работе реализация алгоритма ВСВ-ХЗДН, не является оптимальной. Для того, чтобы сделать обмен ключами в блокчейне быстрее и дешевле, необходимо оптимизировать код смарт-контрактов. Более того, изменений стоит ждать и от разработчиков сети Ethereum. За счет различных изменений (например, сети второго уровня (Layer 2)) транзакции станут дешевле.

Список литературы

1. *Diffie W., Hellman M.* New directions in cryptography // IEEE Transactions on Information Theory. — 1976. — Т. 22, № 6. — С. 644—654.
2. *Nakamoto S.* Bitcoin: A Peer-to-Peer Electronic Cash System. — 2009.
3. *Buterin V.* Ethereum: A next-generation smart contract and decentralized application platform. — 2014.
4. *Marlinspike M., Perrin T.* The X3DH Key Agreement Protocol. — 2016.
5. Ethereum: A secure decentralised generalised transaction ledger / G. Wood [и др.] // Ethereum project yellow paper. — 2014. — Т. 151, № 2014. — С. 1—32.
6. BCB-X3DH: a Blockchain Based Improved Version of the Extended Triple Diffie-Hellman Protocol / A. Ruggeri [и др.]. — 2020.