# Лабораторная работа № 2

Цель работы: получить знания о внутреннем устройстве EVM, получить навыки программирования на языке Solidity, проектирования и создания смарт-контрактов.

Результат: написанный и размещенный в Ropsten смарт-контракт.

Дата представления результатов: 12.11.2021

## Задание

1. Спроектировать и написать смарт-контракт реализующий логику игры “Камень-ножницы-бумага”
2. Реализовать ролевую модель и использовать modifier (<https://habr.com/ru/post/572004/>)
3. Использовать схему commit-reveal для фиксации выбора игрока и обосновать почему она нужна
4. Добавить события в функции смарт-контракта
5. Разместить в блокчейне Ethereum-Ropsten
6. Опубликовать исходный код в github

Выполнение

Схема алгоритма тут

https://miro.com/app/board/o9J\_llMbqyY=/?invite\_link\_id=913984598901

Как сгенерировать commitment:

*pip3 install web3*

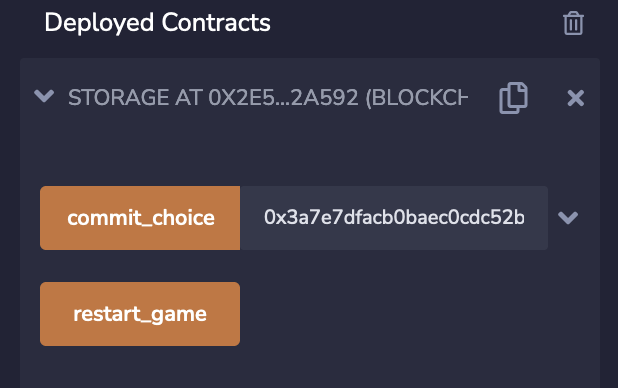
*Import web3*

*web3.Web3.solidityKeccak(['string', 'string'], ['paper', 'password12345'])*

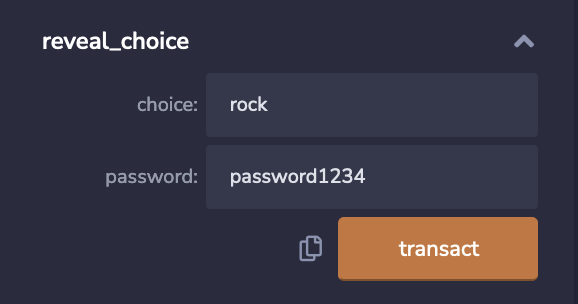
*HexBytes('0xb0c1f97a3a2b7f321dcef007069b5aa7b7d8f608e59fac217519f6fd4450076c')*

В коде выше есть  *['paper', 'password12345'] -* тут paper – это выбор пользователя, а password12345 – секрет, чтобы по хэшу нельзя было восстановить выбор (HMAC).

Дальше в сам смарт-контракт можно передавать только полученный хэш, как в данном случае *0xb0c1f97a3a2b7f321dcef007069b5aa7b7d8f608e59fac217519f6fd4450076c:*

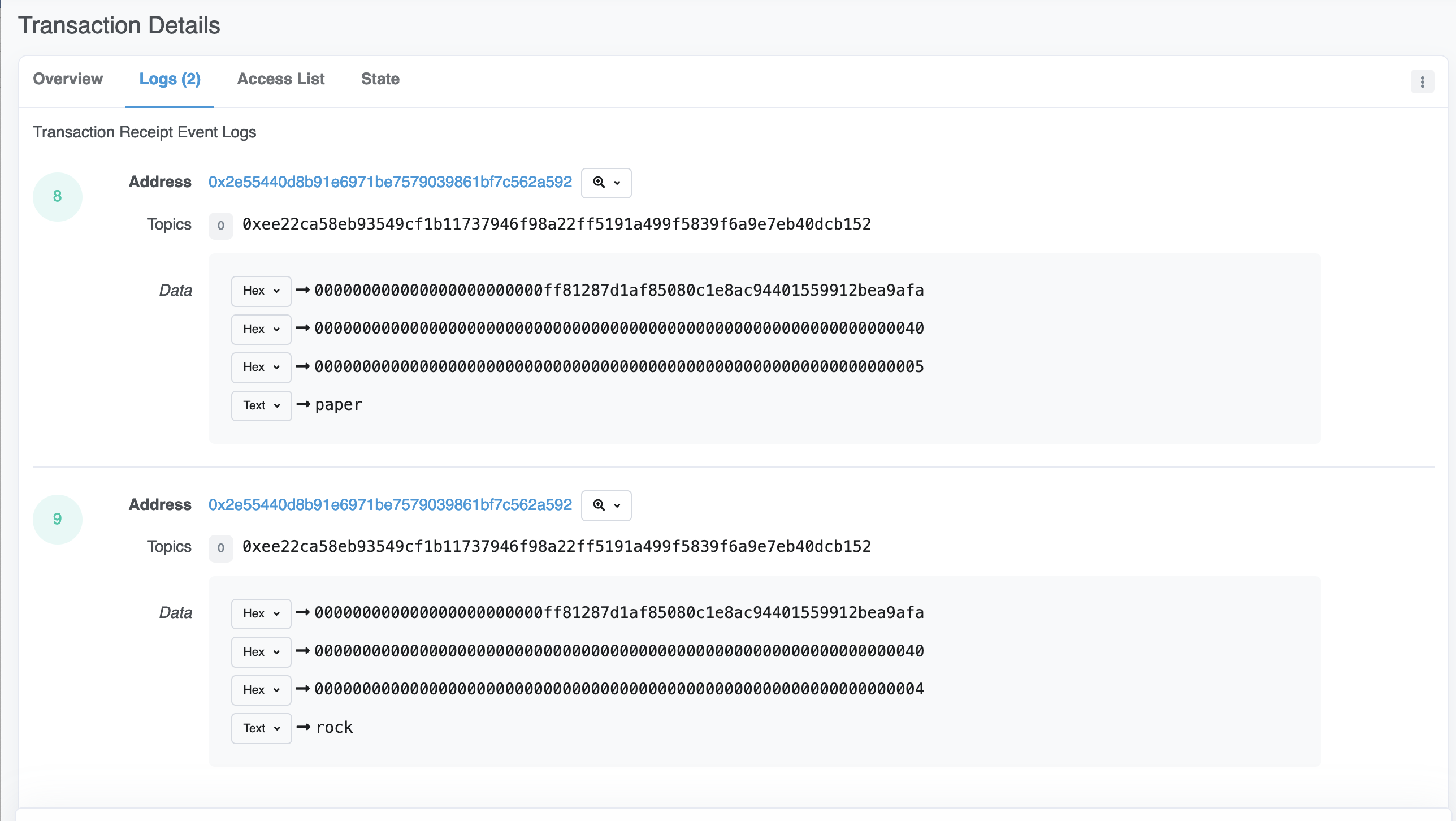
**

Затем выбор и секрет можно передать в reveal\_choice



Пользователи не видят значения друг друга, поскольку изначально в смарт-контракт записываются хэши от секрета и выбора.

Затем после старта игры (когда оба выбора уже рассекречены) рассылаются уведомления о том, кто какой выбор сделал:



Код контракта находится в репозитории

<https://github.com/Parkhomets/crypto_solidity>

## Контрольные вопросы

1. В чём особенность смарт-контрактов без конструктора?

Они будут одинаковы, поскольку не принимают никаких переменных во время деплоя, чтобы задать внутреннее состояние.

1. Чем отличается тип смарт-контракта library от типа contract?

Lib – это библиотека. Библиотека в Solidity - это тип смарт-контракта, содержащий многократно используемый код. После развертывания в блокчейне (развёртывается только один раз) ему присваивается определённый адрес, а его свойства / методы могут многократно использоваться другими смарт-контрактами в сети Ethereum.

<https://habr.com/ru/post/572894/>

1. Какие типы памяти существуют в EVM?

Переменные объявляются как данные хранилища, памяти или вызова, чтобы явно указать местоположение данных.

**storage** - это переменная состояния (хранилище в блокчейне)

**memory** - переменная находится в памяти и существует во время вызова функции

**calldata** - специальное расположение данных, содержащее аргументы функции, доступное только для внешних функций

1. Зачем нужен ABI?

Это стандартный способ взаимодействия с контрактами в экосистеме Ethereum, как за пределами блокчейна, так и для взаимодействия между контрактами.

1. Зачем нужны вставки assembly в смарт-контакт?

assembly позволяет считывать целые слова (256 бит) из таких типов данных, как строка и байты, за одну операцию. Solidity-stringutils использует это для очень быстрого сравнения строк путем вычитания 32-байтовых фрагментов двух сравниваемых строк. Без assembly это придется делать это байт за байтом.

С помощью inline assembly можно передать строку и хэшировать только тот бит, который интересует.

Solidity не поддерживает получение возвращаемых значений от внешних функций, которые возвращают типы переменной длины, такие как динамические массивы, байты или строки, но если вы знаете ожидаемую длину, вы можете вызвать их с помощью inline assembly.

1. Зачем нужен тип msg, tx, block?

Специальные типы данных, свойственные блокчейну (msg – сообщение, tx – транзакция, block – блок). С ними можно работать – получать данные об отправителе, о содержимом, проверять хэш и тд.

1. Как можно задать случайное значение в смарт-контракт?

<https://blog.chain.link/random-number-generation-solidity/>

Но это больше похоже на возвращение результата хэш-функции в зависимости от передаваемого пользователем значения

## Ссылки

1. [Contracts — Solidity 0.8.9 documentation](https://docs.soliditylang.org/en/latest/contracts.html)
2. [Remix](http://remix.ethereum.org/)
3. [OpenZeppelin/awesome-openzeppelin: Blockchain educational resources curated by the OpenZeppelin team](https://github.com/OpenZeppelin/awesome-openzeppelin)
4. [Ethereum Virtual Machine Opcodes](https://www.ethervm.io/)
5. [Software Engineering Techniques - Ethereum Smart Contract Best Practices](https://consensys.github.io/smart-contract-best-practices/software_engineering/)
6. [Creating Upgradeable Contracts From Solidity](https://docs.openzeppelin.com/upgrades/2.8/creating-upgradeable-from-solidity)

Дополнительные ссылки:

<https://habr.com/ru/post/572004/>

<https://ethereum.org/ru/developers/tutorials/logging-events-smart-contracts/>

<https://medium.com/swlh/exploring-commit-reveal-schemes-on-ethereum-c4ff5a777db8>

<https://betterprogramming.pub/learn-solidity-events-2801d6a99a92>

https://web3py.readthedocs.io/en/stable/web3.main.html