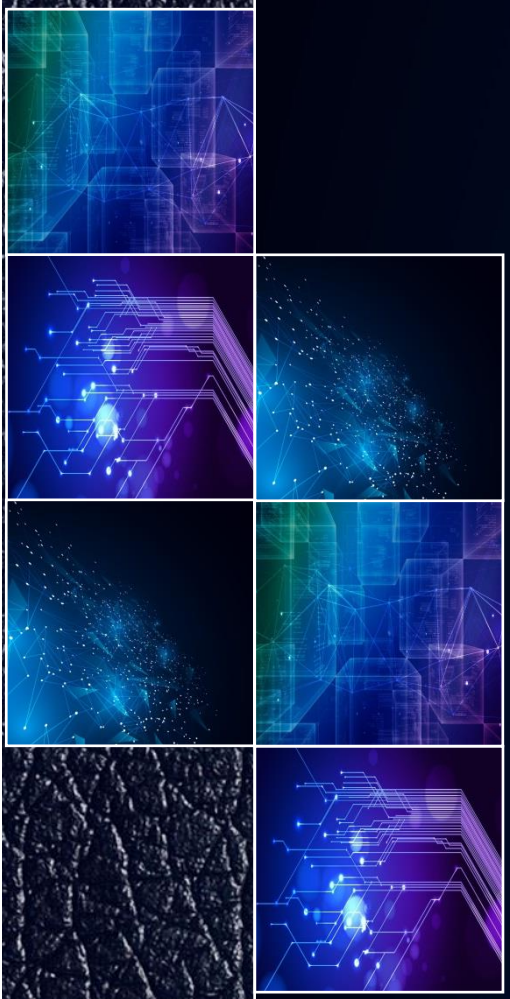
2024

Желтая бумага SLS ICO

Версия 1.1

Документ " Yellow Paper " предоставляет собой всесторонний технический обзор токена SLS, его базовой технологии, протоколов и алгоритмов. Он служит техническим руководством для разработчиков и специалистов, позволяя понять внутренние механизмы экосистемы SLS.



Компания SlonPlus

19.07.2024

1. Введение

1.1 Цель

Токен SLS предназначен для облегчения транзакций в экосистеме SlonPlus, обеспечивая безопасное и эффективное средство обмена ценностями. Этот Yellow Paper описывает техническую архитектуру и операционные механизмы токена SLS.

1.2 Объем

Этот документ охватывает математические модели, алгоритмы шифрования, консенсусные протоколы и другие технические спецификации, необходимые для токена SLS.

2. Технические спецификации токена SLS

Токен SLS - это цифровой актив, построенный на блокчейне TRON и соответствующий стандарту TRC-20. Этот документ предоставляет подробные технические спецификации токена SLS, включая его архитектуру, функциональные возможности и меры безопасности.

2.1. Общая информация

- **Название токена:** Shares SlonPlus
- **Символ токена:** SLS+
- **Десятичные знаки:** 18
- **Общее количество токенов:** 560,000,000 SLS+
- **Платформа блокчейна:** TRON
- **Стандарт токена:** TRC-20

2.2 Подробности контракта токена

2.2.1 Адрес контракта

- **Адрес контракта:** Будет предоставлен после развертывания

2.2.2 Версия компилятора

- **Версия Solidity:** ^0.8.20

2.2.3 Ключевые переменные

- **name:** string - Название токена.
- **symbol:** string - Символ токена.
- **decimals:** uint8 - Количество десятичных знаков, используемых токеном.
- **_totalSupply:** uint - Общее количество токенов в обращении.
- **TokenPrice:** uint256 - Цена одного токена в wei (18 десятичных знаков).
- **owner:** address - Адрес владельца контракта.
- **newOwner:** address - Адрес нового владельца при передаче прав собственности.
- **paused:** bool - Состояние контракта, указывающее, приостановлен ли он.

2.2.4 Отображения

- **balances:** mapping(address => uint) - Отображает адреса на их соответствующие балансы токенов.
- **allowed:** mapping(address => mapping(address => uint)) - Отображает адреса на их разрешения на расходование токенов от имени других.

2.3. Основные функции

2.3.1 Общее количество токенов

- **Функция:** totalSupply()
- **Описание:** Возвращает общее количество токенов в обращении, за исключением тех, которые находятся на нулевом адресе.
- **Тип возврата:** uint

2.3.2 Баланс

- **Функция:** balanceOf(address tokenOwner)
- **Описание:** Возвращает баланс токенов указанного адреса.
- **Параметры:**
  - tokenOwner: address - Адрес, баланс которого нужно получить.
- **Тип возврата:** uint

2.3.3 Перевод

- **Функция:** transfer(address to, uint tokens)
- **Описание:** Переводит токены с адреса отправителя на адрес получателя.
- **Параметры:**
  - to: address - Адрес получателя.
  - tokens: uint - Количество токенов для перевода.
- **Тип возврата:** bool

2.3.4 Одобрение

- **Функция:** approve(address spender, uint tokens)
- **Описание:** Одобряет указанному адресу перевод токенов от имени владельца.
- **Параметры:**
  - spender: address - Адрес, которому разрешено тратить токены.
  - tokens: uint - Количество токенов для одобрения.
- **Тип возврата:** bool

2.3.5 Перевод с

- **Функция:** transferFrom(address from, address to, uint tokens)
- **Описание:** Переводит токены от имени владельца.
- **Параметры:**
  - from: address - Адрес, с которого переводятся токены.
  - to: address - Адрес получателя.
  - tokens: uint - Количество токенов для перевода.
- **Тип возврата:** bool

2.3.6 Разрешение

- **Функция:** allowance(address tokenOwner, address spender)

- **Описание:** Возвращает оставшееся количество токенов, одобренных для перевода.
- **Параметры:**
  - tokenOwner: address - Владельцы токенов.
  - spender: address - Адрес, которому разрешено тратить токены.
- **Тип возврата:** uint

2.3.7 Покупка токенов

- **Функция:** buyTokens()
- **Описание:** Позволяет пользователям покупать токены, отправляя эфир на контракт.
- **Тип возврата:** void

2.3.8 Продажа токенов

- **Функция:** sellTokens(uint256 amount)
- **Описание:** Позволяет пользователям продавать токены и получать эфир в обмен.
- **Параметры:**
  - amount: uint256 - Количество токенов для продажи.
- **Тип возврата:** void

2.3.9 Приостановка

- **Функция:** pause()
- **Описание:** Приостанавливает контракт, отключая функции покупки и продажи.
- **Тип возврата:** void

2.3.10 Возобновление

- **Функция:** unpause()
- **Описание:** Возобновляет работу контракта, включая функции покупки и продажи.
- **Тип возврата:** void

2.3.11 Передача прав собственности

- **Функция:** transferOwnership(address _newOwner)
- **Описание:** Инициирует передачу прав собственности новому владельцу.
- **Параметры:**
  - _newOwner: address - Адрес нового владельца.
- **Тип возврата:** void

2.3.12 Принятие прав собственности

- **Функция:** acceptOwnership()
- **Описание:** Новый владелец принимает права собственности на контракт.
- **Тип возврата:** void

2.4 Меры безопасности

2.4.1 Модификатор только для владельца

- **Модификатор:** onlyOwner
- **Описание:** Обеспечивает, чтобы определенные функции могли вызываться только владельцем контракта.

2.4.2 Безопасная математика

- **Библиотека:** SafeMath
- **Описание:** Обеспечивает безопасные арифметические операции для предотвращения переполнения и недополнения.

2.4.3 Защита от повторного входа

- **Паттерн:** Checks-effects-interactions
- **Описание:** Защищает критические функции от атак повторного входа.

2.5 События

2.5.1 Событие перевода

- **Событие:** Transfer(address indexed from, address indexed to, uint tokens)
- **Описание:** Вызывается при переводе токенов.

2.5.2 Событие одобрения

- **Событие:** Approval(address indexed tokenOwner, address indexed spender, uint tokens)
- **Описание:** Вызывается при одобрении владельцем токенов для переводов другим адресам.

2.6 Операционные механизмы

2.6.1 Цена токена

- **Цена токена:** 1 USD в wei (18 десятичных знаков)

2.6.2 Покупка токенов

- Пользователи отправляют эфир на контракт для покупки токенов.
- Количество купленных токенов рассчитывается на основе текущей цены токена.
- Токены переводятся с баланса контракта на баланс пользователя.

2.6.3 Продажа токенов

- Пользователи могут продавать свои токены обратно на контракт.
- Эквивалентная сумма эфира за проданные токены рассчитывается и отправляется пользователю.
- Токены переводятся с баланса пользователя на баланс контракта.

2.6.4 Приостановка и возобновление

- Владелец контракта может приостановить контракт, чтобы отключить покупку и продажу токенов.
- Владелец контракта может возобновить контракт, чтобы снова включить покупку и продажу токенов.

Токен SLS использует стандарт TRC-20 на блокчейне TRON для обеспечения безопасности, эффективности и удобства использования. Его архитектура включает надежные меры

безопасности, операционные протоколы и функциональные возможности для обеспечения плавного взаимодействия в экосистеме TRON.

2.7 Смарт-контракт

Токен SLS реализован как совместимый с TRC-20 смарт-контракт на блокчейне TRON. Смарт-контракт управляет созданием токенов, их передачей и другими операциями.

```
pragma solidity ^0.8.20;

interface TRC20 {
   function totalSupply() external view returns (uint);
   function balanceOf(address tokenOwner) external view returns (uint balance);
   function allowance(address tokenOwner, address spender) external view returns (uint remaining);
   function transfer(address to, uint tokens) external returns (bool success);
   function approve(address spender, uint tokens) external returns (bool success);
   function transferFrom(address from, address to, uint tokens) external returns (bool success);

   event Transfer(address indexed from, address indexed to, uint tokens);
   event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}

contract SharesSlonPlus is TRC20 {
   string public name = "Shares SlonPlus";
   string public symbol = "SLS+";
   uint8 public decimals = 18;
   uint public _totalSupply = 560000000 * 10**uint(decimals);
   uint256 public TokenPrice = 1e18; // 1 USD в wei (18 десятичных знаков)
   address public owner;
   address public newOwner;
   bool public paused = false;

   mapping(address => uint) balances;
   mapping(address => mapping(address => uint)) allowed;

   modifier onlyOwner() {
      require(msg.sender == owner, "Only the owner can call this function");
      _;
   }

   constructor () {
      owner = msg.sender;
      balances[owner] = _totalSupply;
      emit Transfer(address(0), owner, _totalSupply);
   }

   function transferOwnership(address _newOwner) public onlyOwner {
      require(_newOwner != address(0), "New owner cannot be zero address");
      newOwner = _newOwner;
   }

   function acceptOwnership() public {
      require(msg.sender == newOwner, "Only the new owner can accept ownership");
      owner = newOwner;
      newOwner = address(0);
   }

   function totalSupply() public view override returns (uint) {
      return _totalSupply - balances[address(0)];
   }

   function balanceOf(address tokenOwner) public view override returns (uint balance) {
      return balances[tokenOwner];
   }

   function transfer(address to, uint tokens) public override returns (bool success) {
      require(to != address(0), "Cannot transfer to the zero address");
      require(balances[msg.sender] >= tokens, "Insufficient balance");
      balances[msg.sender] -= tokens;
      balances[to] += tokens;
      emit Transfer(msg.sender, to, tokens);
      return true;
   }
```

```
    function approve(address spender, uint tokens) public override returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }

    function transferFrom(address from, address to, uint tokens) public override returns (bool success) {
        require(to != address(0), "Cannot transfer to the zero address");
        require(balances[from] >= tokens, "Insufficient balance");
        require(allowed[from][msg.sender] >= tokens, "Insufficient allowance");
        balances[from] -= tokens;
        allowed[from][msg.sender] -= tokens;
        balances[to] += tokens;
        emit Transfer(from, to, tokens);
        return true;
    }

    function allowance(address tokenOwner, address spender) public view override returns (uint remaining) {
        return allowed[tokenOwner][spender];
    }

    function buyTokens() public payable {
        require(!paused, "Contract is paused");
        uint256 amount = msg.value / TokenPrice;
        require(amount > 0, "You need to send some TRX");
        require(amount <= balances[owner], "Insufficient balance in the contract");
        balances[owner] -= amount;
        balances[msg.sender] += amount;
        emit Transfer(owner, msg.sender, amount);
    }

    function sellTokens(uint256 amount) public {
        require(!paused, "Contract is paused");
        require(amount > 0, "You need to sell at least one token");
        require(amount <= balances[msg.sender], "Insufficient balance");
        uint256 trxAmount = amount * TokenPrice;
        balances[msg.sender] -= amount;
        balances[owner] += amount;

        // Использование call вместо transfer
        (bool success, ) = msg.sender.call{value: trxAmount}("");
        require(success, "Transfer failed.");

        emit Transfer(msg.sender, owner, amount);
    }

    function pause() public onlyOwner {
        paused = true;
    }

    function unpause() public onlyOwner {
        paused = false;
    }

    // Функция для обработки поступления TRX на контракт
    receive() external payable {
        // Здесь можно добавить пользовательскую логику для получения TRX
    }
}
```

3. Техническая архитектура и операционные механизмы токена SLS

3.1 Техническая архитектура

3.1.1 Платформа блокчейн

Токен SLS реализован на блокчейне TRON, который обладает высокой пропускной способностью, низкими транзакционными издержками и мощными возможностями для

смарт-контрактов. Выбор TRON обеспечивает эффективные и масштабируемые операции для токена SLS.

3.1.2 Смарт-контракт

Смарт-контракт токена SLS соответствует стандарту TRC-20. Он управляет выпуском токенов, их передачей и другими функциями, необходимыми для работы токена.

3.1.3 Ключевые компоненты

- **Детали токена:**
  - **Имя:** Shares SlonPlus
  - **Символ:** SLS+
  - **Десятичные знаки:** 18
  - **Общий объем эмиссии:** 560,000,000 SLS+
- **Переменные контракта:**
  - TokenPrice: Цена токена установлена на уровне 1 USD в wei.
  - owner: Адрес владельца контракта.
  - newOwner: Адрес нового владельца для передачи прав собственности.
  - paused: Булева переменная, указывающая, приостановлен ли контракт.

3.1.4 Структуры данных

- **Отображения:**
  - balances: Отслеживает баланс токенов SLS для каждого адреса.
  - allowed: Отслеживает разрешенные суммы для передачи токенов от имени других адресов.

3.1.5 Модификаторы

- **onlyOwner:** Обеспечивает вызов определенных функций только владельцем контракта.

3.1.6 Функции

- **Основные функции:**
  - totalSupply(): Возвращает общий объем эмиссии токенов.
  - balanceOf(address tokenOwner): Возвращает баланс токенов для данного адреса.
  - allowance(address tokenOwner, address spender): Возвращает оставшиеся токены, которые разрешено потратить с данного адреса.
  - transfer(address to, uint tokens): Переводит токены на указанный адрес.
  - approve(address spender, uint tokens): Разрешает указанному адресу потратить определенное количество токенов.
  - transferFrom(address from, address to, uint tokens): Переводит токены с одного адреса на другой с использованием разрешения.
- **Функции управления правами собственности:**
  - transferOwnership(address _newOwner): Инициирует передачу прав собственности новому владельцу.
  - acceptOwnership(): Принимает права собственности новым владельцем.
- **Функции продажи токенов:**
  - buyTokens(): Позволяет пользователям покупать токены, отправляя TRX на контракт.
  - sellTokens(uint256 amount): Позволяет пользователям продавать токены в обмен на TRX.
- **Функции управления контрактом:**
  - pause(): Приостанавливает операции контракта.
  - unpause(): Возобновляет операции контракта.

- **Функция обратного вызова:**
  - receive(): Обрабатывает получение TRX, отправленных на контракт.

3.2 Операционные механизмы

3.2.1 Распределение токенов

Общий объем эмиссии токенов SLS предварительно создан и распределен владельцу контракта. Владелец может распределять токены через различные механизмы, такие как продажи, награды или другие методы.

3.2.2 Ценообразование токенов

Цена токена SLS установлена на уровне 1 USD в wei. Этот механизм ценообразования обеспечивает стабильную оценку токена и упрощает процесс покупки и продажи для пользователей.

3.2.3 Покупка токенов

Пользователи могут покупать токены SLS, отправляя TRX на контракт. Контракт рассчитывает количество токенов, которые будут выданы на основе отправленной суммы TRX и цены токена. Баланс пользователя обновляется, и генерируется событие Transfer.

3.2.4 Продажа токенов

Пользователи могут продавать свои токены SLS обратно на контракт в обмен на TRX. Контракт рассчитывает количество TRX, которое будет отправлено на основе цены токена. Баланс пользователя обновляется, и сумма TRX передается с использованием метода call для обеспечения безопасности против атак повторного входа.

3.2.5 Передача токенов

Пользователи могут переводить токены на другие адреса. Контракт проверяет, что у отправителя достаточно токенов и что получатель не является нулевым адресом. Балансы обновляются, и генерируется событие Transfer.

3.2.6 Разрешение и transferFrom

Пользователи могут разрешить другим адресам тратить токены от их имени, используя функцию approve. Функция transferFrom позволяет распорядителю переводить токены с адреса владельца на другой адрес в пределах установленного лимита.

3.2.7 Приостановка контракта

Владелец контракта может приостанавливать и возобновлять операции контракта с помощью функций pause и unpause. При приостановке критические функции, такие как покупка, продажа и передача токенов, отключаются для защиты от потенциальных угроз или уязвимостей.

3.2.8 Передача прав собственности

Контракт поддерживает безопасную передачу прав собственности через двухэтапный процесс. Текущий владелец инициирует передачу, вызывая функцию transferOwnership с указанием нового владельца. Новый владелец должен принять передачу, вызвав функцию acceptOwnership.

3.2.9 Меры безопасности

- **Защита от повторного входа:** Контракт использует метод call для передачи TRX, что снижает риск атак повторного входа.
- **Проверка нулевого адреса:** Передачи и изменения прав собственности проверяются для предотвращения операций с нулевым адресом.
- **Экстренная остановка:** Механизм приостановки предоставляет возможность экстренной остановки для прекращения операций контракта в случае необходимости.

4. Математическая модель токена SLS

Математическая модель токена SLS описывает алгоритмы и формулы, используемые для управления эмиссией, распределением, ценой и операциями с токенами. Это описание включает в себя основные математические концепции и методы, применяемые в смарт-контракте, а также обоснование выбора данных моделей.

4.1. Эмиссия и распределение токенов

4.1.1 Общая эмиссия

Объем эмиссии токенов SLS определен как фиксированная величина на этапе создания контракта. Это число отражается в переменной _totalSupply.

Total Supply=560,000,000×1018 SLS\text{Total Supply} = 560,000,000 \times 10^{18} \, \text{SLS}Total Supply=560,000,000×1018SLS

где 101810^{18}1018 - это десятичные знаки токена.

4.1.2 Начальное распределение

Все токены на этапе создания контракта распределяются на адрес владельца контракта:

Initial Balance of Owner=Total Supply\text{Initial Balance of Owner} = \text{Total Supply}Initial Balance of Owner=Total Supply

4.2. Ценообразование токенов

4.2.1 Цена токена

Цена одного токена SLS установлена в эквиваленте 1 USD в wei. В контексте TRON, 1 wei равен 10−1810^{-18}10−18 TRX.

Token Price=1 USD×1018 wei\text{Token Price} = 1 \, \text{USD} \times 10^{18} \, \text{wei}Token Price=1USD×1018wei

4.2.2 Конвертация TRX в токены

Количество токенов, которые пользователь может приобрести за отправленные TRX, рассчитывается следующим образом:

Amount of Tokens=Amount of TRXToken Price\text{Amount of Tokens} = \frac{\text{Amount of TRX}}{\text{Token Price}}Amount of Tokens=Token PriceAmount of TRX

где Amount of TRX\text{Amount of TRX}Amount of TRX - сумма отправленных TRX.

4.2.3 Конвертация токенов в TRX

Количество TRX, которое пользователь получает при продаже токенов, рассчитывается следующим образом:

Amount of TRX=Amount of Tokens×Token Price\text{Amount of TRX} = \text{Amount of Tokens} \times \text{Token Price}Amount of TRX=Amount of Tokens×Token Price

где Amount of Tokens\text{Amount of Tokens}Amount of Tokens - количество продаваемых токенов.

4.3. Операции с токенами

4.3.1 Баланс токенов

Баланс токенов для любого адреса рассчитывается как сумма всех поступлений на этот адрес минус все отправленные с него токены.

Balance(a)=∑Received(a)−∑Sent(a)\text{Balance}(a) = \sum \text{Received}(a) - \sum \text{Sent}(a)Balance(a)=∑Received(a)−∑Sent(a)

где aaa - адрес пользователя.

4.3.2 Передача токенов

При передаче токенов от одного адреса к другому баланс каждого из участников обновляется следующим образом:

Balance(sender)=Balance(sender)−Amount\text{Balance}(sender) = \text{Balance}(sender) - \text{Amount}Balance(sender)=Balance(sender)−Amount
Balance(receiver)=Balance(receiver)+Amount\text{Balance}(receiver) = \text{Balance}(receiver) + \text{Amount}Balance(receiver)=Balance(receiver)+Amount

где Amount\text{Amount}Amount - количество переданных токенов.

4.3.3 Разрешение и transferFrom

Разрешение на передачу токенов от имени другого адреса осуществляется через функцию approve. Разрешенная сумма записывается в таблицу allowed.

allowed[owner][spender]=Amount\text{allowed}[owner][spender] = \text{Amount}allowed[owner][spender]=Amount

Передача токенов через transferFrom уменьшает разрешенную сумму и баланс отправителя, увеличивая баланс получателя:

allowed[owner][spender]=allowed[owner][spender]−Amount\text{allowed}[owner][spender] = \text{allowed}[owner][spender] - \text{Amount}allowed[owner][spender]=allowed[owner][spender]−Amount
Balance(owner)=Balance(owner)−Amount\text{Balance}(owner) = \text{Balance}(owner) - \text{Amount}Balance(owner)=Balance(owner)−Amount
Balance(receiver)=Balance(receiver)+Amount\text{Balance}(receiver) = \text{Balance}(receiver) + \text{Amount}Balance(receiver)=Balance(receiver)+Amount

4.4. Механизмы управления

4.4.1 Приостановка и возобновление контракта

Приостановка операций контракта осуществляется изменением состояния переменной paused. Если контракт приостановлен, основные функции операций с токенами (такие как покупка, продажа и передача) недоступны.

paused=true\text{paused} = \text{true}paused=true

Возобновление операций изменяет значение переменной paused на false.

paused=false\text{paused} = \text{false}paused=false

4.4.2 Передача прав собственности

Передача прав собственности осуществляется в два этапа: инициирование передачи и принятие прав. На первом этапе текущий владелец назначает нового владельца, а на втором этапе новый владелец принимает права.

newOwner=new owner address\text{newOwner} = \text{new owner address}newOwner=new owner address owner=newOwner\text{owner} = \text{newOwner}owner=newOwner newOwner=address(0)\text{newOwner} = \text{address(0)}newOwner=address(0)

Математическая модель токена SLS детально описывает алгоритмы и методы, используемые для управления токеном, обеспечивая надежные и прозрачные операции. Данная модель основывается на стандарте TRC-20, что обеспечивает совместимость и стабильность в рамках блокчейна TRON.

5. Алгоритмы шифрования токена SLS

Токен SLS использует криптографические алгоритмы для обеспечения безопасности, целостности и конфиденциальности транзакций в блокчейне TRON. Этот документ содержит подробное описание алгоритмов шифрования и механизмов безопасности, используемых токеном SLS.

5.1. Криптография с открытым ключом

5.1.1 Криптография на эллиптических кривых (ECC)

TRON использует криптографию на эллиптических кривых (ECC) для криптографии с открытым ключом. ECC обеспечивает высокий уровень безопасности при относительно небольших ключах, что делает его эффективным и безопасным.

- **Алгоритм:** SECP256k1
- **Размер ключа:** 256 бит

Генерация ключей:

1. **Приватный ключ:** Случайно сгенерированное число размером 256 бит.
2. **Публичный ключ:** Получен из приватного ключа с использованием алгоритма ECC.

Публичный ключ=Приватный ключ×G\text{Публичный ключ} = \text{Приватный ключ} \times GПубличный ключ=Приватный ключ×G

где GGG — это точка генератора на кривой.

5.1.2 Цифровые подписи

Цифровые подписи используются для проверки подлинности и целостности транзакций. TRON использует алгоритм цифровой подписи на эллиптических кривых (ECDSA).

- **Алгоритм:** ECDSA
- **Кривая:** SECP256k1

Генерация подписи:

1. **Хеширование сообщения:** Сообщение хешируется с использованием алгоритма SHA-256.
2. **Создание подписи:** Хеш подписывается с использованием приватного ключа.

Подпись=(r,s)\text{Подпись} = (\text{r}, \text{s})Подпись=(r,s)

где r\text{r}r и s\text{s}s получены из приватного ключа и хеша сообщения.

Проверка подписи:

1. **Вычисление хеша:** Полученное сообщение хешируется с использованием SHA-256.
2. **Проверка подписи:** Публичный ключ и подпись используются для проверки хеша.

Проверка(Публичный ключ,Хеш сообщения,(r,s))=true/false\text{Проверка}(\text{Публичный ключ}, \text{Хеш сообщения}, (\text{r}, \text{s})) = \text{true/false}Проверка(Публичный ключ,Хеш сообщения,(r,s))=true/false

5.2. Хеш-функции

5.2.1 SHA-256

Алгоритм Secure Hash Algorithm 256 (SHA-256) используется для создания криптографических хешей данных. Он обеспечивает целостность данных, создавая хеш фиксированного размера (256 бит) из произвольного количества входных данных.

Свойства:

- **Размер выхода:** 256 бит
- **Устойчивость к коллизиям:** Очень высокая
- **Устойчивость к прообразу:** Очень высокая

Вычисление хеша:

Хеш=SHA-256(данные)\text{Хеш} = \text{SHA-256}(\text{данные})Хеш=SHA-256(данные)

5.2.2 KECCAK-256

KECCAK-256, вариант алгоритма SHA-3, используется в некоторых операциях TRON. Он создает хеш размером 256 бит и известен своей безопасностью и эффективностью.

Вычисление хеша:

Хеш=KECCAK-256(данные)\text{Хеш} = \text{KECCAK-256}(\text{данные})Хеш=KECCAK-256(данные)

5.3. Защищенная связь

5.3.1 Transport Layer Security (TLS)

TLS обеспечивает защищенную связь между клиентами и серверами, взаимодействующими с блокчейном TRON. Он обеспечивает шифрование, целостность данных и аутентификацию.

- **Версия:** TLS 1.2 или выше
- **Алгоритмы шифрования:** AES-256, ChaCha20
- **Обмен ключами:** ECDHE (Эфемерный Диффи-Хеллман на эллиптических кривых)
- **Аутентификация:** X.509 сертификаты

Процесс рукопожатия TLS:

1. **Приветствие клиента:** Клиент предлагает метод шифрования.
2. **Приветствие сервера:** Сервер соглашается на метод шифрования.
3. **Обмен сертификатами:** Сервер отправляет свой сертификат клиенту.
4. **Обмен ключами:** Клиент и сервер обмениваются ключами.
5. **Завершение рукопожатия:** Обе стороны подтверждают рукопожатие и начинают зашифрованную связь.

5.4. Безопасность операций с токенами

5.4.1 Шифрование транзакций

Все транзакции в блокчейне TRON зашифрованы с использованием приватного ключа отправителя. Детали транзакции хешируются с использованием SHA-256, и хеш подписывается приватным ключом для создания цифровой подписи.

Структура транзакции:

1. **Адрес отправителя:** Публичный ключ отправителя.
2. **Адрес получателя:** Публичный ключ получателя.
3. **Количество:** Количество токенов для передачи.
4. **Подпись:** Цифровая подпись хеша транзакции.

5.4.2 Безопасность смарт-контрактов

Смарт-контракт токена SLS включает несколько механизмов безопасности для предотвращения распространенных уязвимостей, таких как повторный вход, переполнение/недополнение целых чисел и несанкционированный доступ.

- **Защита от повторного входа:** Предотвращает повторные вызовы.
- **Библиотека SafeMath:** Обеспечивает безопасные арифметические операции.
- **Контроль доступа:** Только владелец контракта может выполнять критические операции, такие как приостановка или возобновление контракта.

Токен SLS использует надежные криптографические алгоритмы для обеспечения безопасности и целостности транзакций в блокчейне TRON. Криптография с открытым ключом, защищенные хеш-функции и транспортный уровень безопасности совместно обеспечивают комплексную систему безопасности для токена SLS и его операций.

6. Описание протоколов токена SLS

Токен SLS, развернутый на блокчейне TRON, использует ряд протоколов для обеспечения надежной работы, безопасности и взаимодействия с пользователями и другими смарт-контрактами. В этом документе представлены основные протоколы, используемые токеном SLS.

6.1. Протокол TRC-20

6.1.1 Описание

TRC-20 — это стандартный протокол для создания и выпуска токенов на блокчейне TRON. Этот стандарт обеспечивает совместимость с различными смарт-контрактами и dApp (децентрализованными приложениями) в экосистеме TRON.

6.1.2 Основные функции TRC-20

- **totalSupply:** Возвращает общее количество выпущенных токенов.
- **balanceOf:** Возвращает баланс токенов указанного адреса.
- **transfer:** Перемещает указанное количество токенов от отправителя к получателю.
- **approve:** Позволяет владельцу токенов разрешить другому адресу тратить токены от его имени.
- **transferFrom:** Перемещает токены от одного адреса к другому, используя разрешение, предоставленное функцией approve.
- **allowance:** Возвращает оставшееся количество токенов, которые были разрешены для передачи определенному адресу.

6.1.3 События TRC-20

- **Transfer:** Записывает перемещение токенов от одного адреса к другому.
- **Approval:** Записывает одобрение на перемещение токенов от имени владельца.

6.2. Протоколы безопасности

6.2.1 Контроль доступа

- **Модификатор onlyOwner:** Обеспечивает, что критические функции могут вызываться только владельцем контракта.
- **transferOwnership:** Позволяет владельцу передать права собственности другому адресу.
- **acceptOwnership:** Позволяет новому владельцу принять права собственности.

6.2.2 Защита от переполнения и недополнения

Использование библиотеки SafeMath обеспечивает безопасные арифметические операции, предотвращая переполнение и недополнение целых чисел.

6.2.3 Защита от повторного входа

Критические функции защищены от атак повторного входа с использованием шаблона "проверка-эффект-взаимодействие".

6.3. Протоколы паузы и возобновления

6.3.1 Пауза

- **pause:** Приостанавливает все критические функции контракта, такие как покупка и продажа токенов. Может быть вызвана только владельцем.
- **unpause:** Возобновляет работу всех приостановленных функций. Может быть вызвана только владельцем.

6.3.2 Применение

Пауза используется для предотвращения операций в случае обнаружения уязвимостей или при необходимости выполнения обновлений и технического обслуживания.

6.4. Протоколы торговли токенами

6.4.1 Покупка токенов

- **buyTokens:** Позволяет пользователям покупать токены, отправляя эфир на контракт. Количество приобретаемых токенов рассчитывается на основе установленной цены токена.

6.4.2 Продажа токенов

- **sellTokens:** Позволяет пользователям продавать токены и получать эфир в обмен. Количество отправляемого эфира рассчитывается на основе установленной цены токена.

6.5. Протоколы взаимодействия с пользователем

6.5.1 Отправка токенов

- **transfer:** Пользователи могут отправлять токены другим пользователям, указывая адрес получателя и количество токенов.

6.5.2 Одобрение и передача

- **approve:** Пользователи могут разрешить третьим лицам тратить токены от их имени.
- **transferFrom:** Третьи лица могут передавать токены от имени пользователя при наличии соответствующего разрешения.

6.5.3 Получение информации

- **balanceOf:** Пользователи могут узнать количество токенов на любом адресе.
- **allowance:** Пользователи могут узнать количество токенов, разрешенных для передачи определенному адресу.

Токен SLS реализован на блокчейне TRON и использует стандарты и протоколы TRC-20 для обеспечения совместимости, безопасности и простоты использования. Контракты токена включают в себя расширенные протоколы безопасности, защиты от атак, а также функции для торговли и управления токенами, что обеспечивает его надежность и удобство для пользователей.

7. Будущие Улучшения

7.1 Обновляемые контракты

Планы по введению механизмов обновляемых контрактов, чтобы позволить будущие улучшения и функции без нарушения существующей экосистемы.

7.2 Дополнительные аудиты безопасности

Регулярные аудиты безопасности и тестирование на проникновение будут проводиться для обеспечения безопасности контракта и его соответствия последним практикам безопасности.

8. Заключение

Yellow Paper предоставляет подробное техническое описание токена SLS, его архитектуры и операционных механизмов. Соблюдая строгие технические стандарты и внедряя надежные меры безопасности, токен SLS стремится предоставить безопасную и эффективную платформу для обмена ценностями в экосистеме SlonPlus.

Этот документ является всесторонним руководством для разработчиков и специалистов, желающих понять технические основы токена SLS. Для получения дополнительных деталей и

обновлений, пожалуйста, обратитесь к официальной документации и репозиториям SlonPlus.