SECAI Token (SECAI): World's first decentralised AI

1. White paper abstract

The World's first decentralised AI will see the light with the SECAI Token (SECAI). SECAI is an ERC20-based cryptocurrency tokken designed to provide a sustainable ecosystem of private, secure, surveillance resistand, efficient, and decentralized AI. This may naturally be of very high value in fascilitating private financial transactions. By integrating advanced artificial intelligence, state-of-the-art encryption, and blockchain technology, SECAI aims for unparalleled privacy protection and surveillance resistance in this Finance revolution. This white paper outlines the innovative approach of SECAI to revolutionize the AI and digital finance landscape, ensuring your privacy, safeguarding user data and seamless transactions.

AI has introduced significant challenges regarding privacy, security, and efficiency. Traditional financial systems often fall short in providing the level of security and privacy users demand, leading to data breaches, unauthorized surveillance, and inefficient processes.

SECAI's Vision

SECAI aims to address these challenges by offering a cryptocurrency that leverages AI-focused technology and blockchain to create a secure, private, and intelligent financial and privacy protection ecosystem revolution. Our mission is to empower individuals and businesses with complete control over their financial data, ensuring protection from surveillance and unauthorized access while providing efficient and seamless transaction processes.

SECAI aims to allow computer owners supply decentralized computing power incentivised by payment in SECAI tokens to AI users. This is the practical growth of philosophical privacy, were not only the ledger is decentralised, but also the computing power is decentralised and data are truly anonymous and surveillance opportunites such as phishing is minimized, and privacy maximised. This ensures that no central computer can store AI searches. This is truly a revolutionary development of the AI future.

2. Blockchain Infrastructure and Technical Specifications

2.1 Blockchain Infrastructure
SECAI operates on the Ethereum blockchain, leveraging its proven security and decentralization.

Our ERC20 token ensures compatibility with existing Ethereum-based systems, providing a reliable and scalable platform.

ERC20 Standard: Ensures compatibility with wallets and exchanges supporting Ethereum tokens.
Security: Utilizes Ethereum's robust security protocols to protect transactions.
Scalability: Designed to handle high transaction volumes with minimal latency.
Source Code: Specified in appendix A.

2.2 Technical Specifications

- Consensus Mechanism: SECAI leverages Ethereum's Proof-of-Stake (PoS) consensus for transaction validation, reducing energy consumption and enhancing scalability.
- Smart Contracts: Custom smart contracts manage transactions, implement AI algorithms, and enforce privacy protocols.
- Privacy Protocols: Zero-Knowledge Proofs (ZKP) and homomorphic encryption enable secure, private transactions and data exchanges.

2.3 Algorithms Used

- Differential Privacy: Ensures data privacy by deleting queries and keeping users anonymous, maintaining the utility of AI models while safeguarding individual privacy.

2.4 Architecture Overview

SECAI’s architecture is composed of three primary layers:

1. Blockchain Layer: Handles decentralized ledger functions, transaction validation, and smart contract execution.
2. AI Layer: Executes AI models and algorithms using secure computation techniques, such as secure multi-party computation (SMPC).
3. Application Layer: Interfaces with end-users and developers, enabling decentralized applications (dApps) to utilize SECAI's AI capabilities securely.

3. Enhanced Privacy Protection

3.1 Privacy Protection

Privacy is at the core of SECAI's design. We employ cutting-edge encryption methods to ensure that user data and transactions remain confidential, empowering users to conduct transactions without fear of surveillance or exploitation. By turning the AI model upside down, users no longer have to «trust big tech» companies. In the future, we aim for decentralized AI searches where users are not tracked for «trust big tech» profits.

End-to-End Encryption: Secure communication channels protect data integrity and confidentiality.
Anonymity Protocols: Conduct transactions without revealing personal information.
Zero-Knowledge Proofs: Verify transactions without disclosing sensitive details.
SECAI employs advanced cryptographic methods like Zero-Knowledge Proofs (ZKP) to enable secure, verifiable transactions without revealing underlying data. Data will be deleted after user queries to ensure that unlike big tech giants, we do not store user data of AI searches.

3.2 Query Deletion Mechanism

A blockchain-based proof of erasure protocol allows users to request the deletion of their queries. This mechanism is implemented through smart contracts that ensure verifiable data removal while maintaining transaction integrity.

3.3. Surveillance Protection

SECAI plan to focus on robust defenses against digital surveillance, aimig to assist that user data is shielded from unauthorized monitoring. Our decentralized architecture and advanced cryptographic techniques aims to help protect against surveillance threats.

Decentralized Data Storage: Reduces vulnerability to attacks and eliminates single points of failure.
Advanced Cryptography: Protects data from unauthorized access and surveillance.
User-Controlled Privacy Settings: Customize privacy preferences to meet individual needs.

3.4 Seamless Integration with Existing Systems

SECAI is designed to integrate seamlessly with existing financial systems, providing an easy transition for businesses and individuals looking to enhance their privacy and transaction capabilities.

API Support: Allows businesses to integrate SECAI into existing platforms easily.
Cross-Platform Compatibility: Works with various devices and operating systems.
Interoperability: Facilitates interactions with other cryptocurrencies and financial services.

4. Regulatory Compliance and Risk Analysis

4.1 KYC and AML Compliance

SECAI integrates Know Your Customer (KYC) and Anti-Money Laundering (AML) compliance. Secure AI AS which is incorporated in Norway, will only send SECAI tokens to purchasors that have supplied ID documents. This will verify user identities of Secure AI AS customers. The ERC blockchain will monitor transactions in real-time. Secure AI AS will not send SECAI tokens in jurisdictions where this is not allowed transaction, aligning with global regulatory standards.

4.2 Risk Analysis

- Legal Risks: Continuous monitoring of regulatory changes and updates to compliance protocols to mitigate potential legal exposure.
- Operational Risks: Deployment of robust cybersecurity measures to protect against data breaches, fraud, and other security threats.
- Market Risks: Diversification strategies and strategic partnerships to ensure resilience against market volatility.

5. Economic Model

SECAI adopts a sustainable economic model where a 1% transaction tax is levied. This tax finances staking rewards, ecosystem maintenance, and development costs.

5.1 Tokenomics

- Total Supply: Fixed supply of SECAI tokens to prevent inflation of 1.000.000 SECAI.

- SECAI tokens used to pay providors of computing power on a market based system.

- Sustainable Economic Model: The 1% tax model on SECAI transactions to separate wallet address supports Ecosystem Fund (50% of tax) for ecosystem sustainability, and Staking Rewards (50% of tax) providing staking rewards.

- Ecosystem Fund: A 50% portion of the transaction tax is allocated to fund ecosystem development, marketing, and legal compliance.

- Staking Rewards: A 50 % portion of the transaction tax is allocated to staking rewards to incentivize active participation and network security.

5.2 Token Utility

SECAI Token (SECAI) serves as the native currency of the SECAI ecosystem, facilitating transactions, incentivizing network participation, and powering smart contract execution.

Transaction Fees: SECAI is used to pay for network transactions and services.

Staking Rewards: Users can stake SECAI to earn rewards and participate in network governance:

Token holders that do not move their SECAI tokens between January 1. 2025 and January 1 2026 will receive 15% staking tokens from Secure AI AS segregated wallet 2025 staking reward address if token holders request this by way of email to staking@secureai.me before January 10. 2026 with wallet address of SECAI holdings documenting the SECAI tokens have not been transferred or transacted in any way in the time period above.

For the 1 year periods after January 1. 2026, 50 % of the transaction tax will be allocated to Token holders that do not move their SECAI tokens in the full calendar year annually if token holders request this by way of email to staking@secureai.me before Janury 10. the following year.

Token Distribution

To ensure a fair and equitable distribution, SECAI's token allocation is structured as follows:

Strategic Advisors: up to 0,5%. Lockup of all tokens until October 1. 2025.

Community and Ecosystem: 10% dedicated to community engagement and ecosystem growth. Intially held by Secure AI AS, but will be released to community in 2025, 2026, and 2027 with 12 month lockup.

Initial investors: 20 %

Team and Advisors: 10% for founding team members. Lockup until October 1. 2025.

Secure AI AS: Initially 59,5 % of tokens with the following lockups*):

**) Staking rewards:*
Of these 59,5% up to 15% of total supply will be reserved for staking rewards for SECAI holders that have not moved their SECAI tokens from their wallet at all in 2025 and claimed their staking rewards by email before mid january 2026.

**) Friends & Family*
Of these 59,5% up to 5 % of total supply may be offered to secure token equitable distribution to Friends & Family at fixed price of US$ 0.01 before October 15. 2024 for a Development Fund allocated for platform development and innovation.

**) Pre Sale Reserve Fund:*
10% for future opportunities and contingencies
A: Of these 59,5% up to 5 % of total supply may be offered to secure token equitable distribution in Pre Sale at fixed price of US$ 0.02 before November 15. 2024.

B: Of these 59,5% up to 5 % of total supply may be offered to secure token equitable distribution in Pre Sale at fixed price of US$ 0.03 before December 15. 2024. Special discounts for large orders may apply.

**) Lockup with maximum 3 % offered for sale per month from 2025.*
The remaining SECAI tokens held by Secure AI AS will be locked up where a maximum of 3 % of total supply may be offered to secure token equitable distribution per month starting in 2025 at prevailing market prices.

6. Roadmap for 2024 and 2025

- Q4 2024: Launch of beta version, user acquisition campaigns, and strategic partnerships.
- Q1-Q2 2025: Full platform launch, scaling operations, and expansion into new markets.
- Q3-Q4 2025: Introduction of advanced AI features, further regulatory compliance measures, and integration with financial institutions.

7. Competitive Analysis

7.1 Market Landscape

SECAI differentiates itself by combining decentralized AI with secure financial transactions, focusing on privacy and compliance. Competitors in this space include other AI-focused blockchain platforms and privacy-centric cryptocurrencies.

7.2 Unique Selling Proposition (USP)

- Privacy-Centric AI 1: The SECAI ecosystem will fund the world's first globally available AI decentralised sustainable ecosystem.
- Privacy-Centric AI 2: SECAI’s use of user search deletion and advanced cryptography ensures robust privacy protections.
- Sustainable Economic Model: The 1% tax model to separate wallet address supports ecosystem sustainability while providing staking rewards.
- Regulatory Compliance: Built-in KYC and AML compliance offers a competitive edge in

regulated markets. Tokens sent from Secure AI AS to purchasors after receipt of ID documentation as Secure AI AS sees suitable for KYC.

7.3. Market Potential and Growth Strategy

7.3.1. Target Market

SECAI targets a diverse range of markets, including individual users seeking privacy-focused financial solutions, businesses looking for secure payment systems, and industries that require advanced data protection.

Individuals: Offering enhanced privacy and security for personal financial transactions.
Businesses: Providing secure and efficient payment solutions for e-commerce and retail.
Institutions: Enabling financial institutions to integrate privacy-focused solutions into their services.

7.3.2. Growth Strategy

SECAI's growth strategy focuses on expanding its user base, building strategic partnerships, and continuously enhancing the platform's features and capabilities.

Partnerships: Collaborating with financial institutions, technology providers, and blockchain projects to expand reach and adoption.
Community Engagement: Building a strong community through social media, events, and educational content.
Continuous Development: Investing in research and development to ensure the platform remains at the forefront of innovation.

8. Strategy for User Adoption

8.1 Marketing and Community Building

- Incentive Programs: Staking rewards, referral bonuses, and community engagement campaigns to attract early adopters. Token holders that do not move their SECAI tokens between January 1. 2025 and January 1 2026 will receive 15% staking tokens from Secure AI AS segregated wallet 2025 staking reward address if token holders request this by way of email to staking@secureai.me before January 10. 2026 with wallet address of SECAI holdings documenting they have not been transferred or transacted in any way in the time period above. For the 1 year periods after January 1. 2026, 50 % of the transaction tax will be allocated to Token holders that do not move their SECAI tokens in the full calendar year annually if token holders request this by way of email to staking@secureai.me before Janury 10. the following year.
- Partnerships: Secuare AI AS will try to establish collaborations with AI and blockchain projects, financial institutions, and regulatory bodies to enhance platform credibility and reach.

9. Case Studies and Use Cases

9.1 Use Cases and Applications

9.1.1. Secure AI searches
SECAI do not store AI searches, meaning Secure AI AS does not have AI search data that criminals or state agencies can request or steal. This is of extreme value for AI search users.

9.1.2. Secure Financial Transactions

SECAI enables secure peer-to-peer transactions, allowing users to transfer funds without intermediaries. Businesses and individuals can leverage our platform for domestic and international payments, enjoying reduced fees and enhanced security.

Cost Savings: Lower transaction fees compared to traditional banking systems.
Speed: Instantaneous transfers across borders.
Security: Encrypted transactions ensure data protection.

9.1.3. Decentralized Finance (DeFi) Solutions

SECAI supports a wide range of DeFi applications, offering users opportunities to engage in lending, borrowing, and investing with enhanced privacy and security. Our platform enables seamless integration with other DeFi protocols, expanding the possibilities for financial innovation.

Lending and Borrowing: Secure, peer-to-peer lending platforms with competitive interest rates.
Decentralized Exchanges: Privacy-focused trading without third-party interference.
Yield Farming: Opportunities for users to earn rewards by providing liquidity.

9.1.4. Privacy-Focused IoT Solutions

SECAI's secure infrastructure extends to the Internet of Things (IoT), protecting data generated by connected devices from surveillance and unauthorized access. Our platform provides a secure environment for IoT applications, ensuring data integrity and privacy, in the future with goals such as:

Smart Home Security: Protecting user data from unauthorized access and monitoring.
Healthcare Devices: Ensuring patient data privacy and security in medical IoT applications.
Supply Chain Management: Enhancing transparency and security in IoT-enabled logistics.

9.1.5. Identity Verification and Data Protection

SECAI offers a robust identity verification solution, ensuring that users' identities are protected and verified without compromising privacy. This is particularly useful for businesses that require secure and private identity management solutions.

Secure Identity Management: Protecting user identities through blockchain-based verification.
Data Encryption: Ensuring data privacy and security in all transactions and interactions.
User-Controlled Access: Allowing users to control who can access their data and for what purpose.

9.2 Case Studies

- Pilot Project: Collaboration with a decentralized exchange (DEX) to implement AI-driven market predictions, achieving improved liquidity and reduced transaction costs.

10. Legal Analysis and Compliance Strategy

10.1 Legal Framework

SECAI is committed to delete searches from users, to ensure that no search data is stored by Secure AI AS.

10.2 Compliance Monitoring

Automated smart contracts monitor for regulatory changes and adjust protocols accordingly by way of manually implementing alternative smart contracts if needed, ensuring continuous alignment with global standards.

11. Financing and Fundraising

11.1 Funding Requirements

Initial funding requirements cover platform development, legal compliance, marketing, and operational expenses.

11.2 Fundraising Strategy

- Private Sale: Early-stage funding to secure initial capital has been conducted.
- Public Offering: Token offering to the public to build a user base and secure further infrastructure costs is expected to be conducted in 2024 or 2025.

12. Team

Secure AI is a limited company registered in the high tech hub of Oslo, the main capital of Scandiavian IT jewel Norway. Secure AI has Norwegian registration number 833574922. Norway is considered an excellent location for AI security for several reasons:

1. Strong Regulatory Framework and Privacy Laws
   - Norway has robust data protection regulations, aligned with the EU's General Data Protection Regulation (GDPR). This ensures that AI systems are developed and used in ways that respect individual privacy and data security.
   - The country also has strong cybersecurity laws and policies, providing a solid legal foundation for AI security measures.

2. High Digital Maturity and Infrastructure
   - Norway ranks high in global digital readiness and maturity indices. The country's advanced digital infrastructure, including high-speed internet and widespread technology adoption, supports the development and deployment of AI securely.
   - Norway's government actively invests in digital transformation and AI research, fostering a conducive environment for AI security advancements.

3. Stable Political Environment
   - Norway is known for its stable political environment, which is crucial for the development and implementation of long-term AI security strategies.
   - The country has a transparent and corruption-free government, promoting trust in public institutions and regulatory bodies involved in AI governance.

4. Strong Research and Development Ecosystem
   - Norway has a well-developed research ecosystem with several institutions and universities

focusing on AI, cybersecurity, and technology development.
- Collaborative partnerships between academia, government, and private sectors enhance AI innovation while ensuring security is prioritized.

5. Ethical AI Focus
- Norway has a strong commitment to ethical AI development. The Norwegian government and organizations advocate for ethical AI practices that prioritize transparency, accountability, and fairness, contributing to the overall security of AI systems.
- The country's approach to AI emphasizes responsible use and safeguards against potential misuse or malicious activities.

6. Highly Skilled Workforce
- Norway boasts a highly educated workforce with expertise in AI, cybersecurity, data science, and software development, essential for building and maintaining secure AI systems.
- Norwegian universities and technical institutions offer specialized programs and research opportunities in AI and cybersecurity, cultivating talent that understands the complexities of AI security.

7. Focus on Cybersecurity
- Norway invests heavily in cybersecurity, a critical component of AI security. The country has developed a comprehensive national cybersecurity strategy to protect critical infrastructure and digital assets.
- Norwegian companies and government agencies are actively involved in global cybersecurity initiatives and collaborations, ensuring they stay ahead of emerging threats.

8. Strong Collaboration and International Partnerships
- Norway participates actively in international AI and cybersecurity collaborations, including those with the EU and other Nordic countries. This involvement fosters the sharing of best practices, research, and innovations in AI security.
- The country's collaboration with other technologically advanced nations ensures access to the latest developments in AI security technologies and methodologies.

9. Focus on Human Rights and Democratic Values
- Norway's commitment to human rights and democratic values aligns with global principles for trustworthy AI. This focus ensures that AI systems are developed and deployed in ways that prioritize human safety, security, and rights.
- The country is an advocate for transparency and accountability in AI, essential for maintaining secure and ethical AI practices.

10. Low Levels of Corruption
- With one of the lowest levels of corruption globally, Norway offers a trustworthy environment for AI development and deployment. This reduces the risk of insider threats or malicious activities that could compromise AI security.

These factors make Norway an attractive and secure location for the development, deployment, and regulation of AI technologies, particularly concerning security.

Core Team:

The Core Team consists of the following:

Board of Directors Chairman Mr. Peter O. Carlsson is a programming specialist with extensive experience from blockchain, AI and cryptography, and additional experience from the Norwegian Financial Industry.

Board of Directors Member Mr. Mathias Haugland is an autodidact expert in blockchain, AI and cryptography, that envisioned a world where AI could be truly decentralized, merging the vision of crypto with the privacy concerns most seasoned Investors have against both hacking, phishing and unautorized surveillance.

They coordinate the team of contributors that want to contribute to a more secure AI community.

13. Long-Term Vision and Environmental Impact

13.1 Vision

SECAI aims to become the leading platform for decentralized AI and privacy-focused financial transactions, setting new standards in data security, user privacy, and regulatory compliance.

13.2 Environmental Impact

By leveraging the Ethereum network's PoS consensus, SECAI minimizes energy consumption, contributing to a sustainable blockchain ecosystem.

14. Conclusion

SECAI represents a revolutionary step towards integrating decentralized AI and financial transactions, offering unique features in privacy protection, regulatory compliance, and sustainable economic growth. The decentralised AI platform is poised to address the growing demand for secure, efficient, and private digital transactions in an evolving global landscape.

We invite you to join us on this journey towards a secure, intelligent, and private AI and financial ecosystem. Together, we can redefine the way AI searches and financial transactions are conducted, empowering individuals and businesses to thrive in a digital age free from surveillance and data exploitation.

Contact Information

For more information, please visit our website: www.SecureAI.me
Email: post@secureai.me
Twitter: https://x.com/SecureAI_Ofc
Telegram: https://t.me/SecureAiPort

For more information, please visit our website: www.SecureAI.me
Appendices

A: Technical Information: Source Code, token distribution models.
B: Legal Disclaimers: Legal notices and compliance statements.

Appendix A:

Technical Information: Source Code including main contract and wallet adresses

Ethereum Contract address:
0xcc41767Ad0007CE1EF3c296Eb5f72E086F1bDfdC

Wallet address for Secure AI AS holdings:
0x049Da70EC0805f62fcc3BdC873E09ABDab7755A8

Wallet address for Secure AI AS stake reward holdings:
0x1FDc3b91e3cA2EaFbAb581A31a955a870a7b38cE

Tax for sustainable payment of decentralized AI network:
1 %

Source Code:
Source Code:


```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "./core/ERC20Taxable.sol";
import "./core/utils/BlackList.sol";
import "@openzeppelin/contracts/security/Pausable.sol";
import "@openzeppelin/contracts/proxy/utils/Initializable.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract TaxableToken is Initializable, ERC20Taxable, Pausable, Ownable, BlackList {
    constructor() {
        _disableInitializers();
    }

    function initialize(
        address _owner,
        string memory _name,
        string memory _symbol,
        uint8 _decimals,
        uint256 _initialSupply,
        uint256 _maxSupply,
        uint256 _taxFeePerMille,
        address _taxAddress
    ) external initializer {
        _transferOwnership(_owner);
        ERC20.init(
            _name,
            _symbol,
            _decimals,
            _maxSupply == type(uint256).max ? type(uint256).max : _maxSupply * 10 ** _decimals
        );
        ERC20Taxable.init(_taxFeePerMille, _taxAddress);
        _mint(_owner, _initialSupply * 10 ** _decimals);
    }

    function pause() public onlyOwner {
        _pause();
    }

    function unpause() public onlyOwner {
        _unpause();
    }
```

```
    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }

    function blockAccount(address _account) public onlyOwner {
        _blockAccount(_account);
    }

    function unblockAccount(address _account) public onlyOwner {
        _unblockAccount(_account);
    }

    function setTaxRate(uint256 _newTaxFee) public onlyOwner {
        _setTaxRate(_newTaxFee);
    }

    function setTaxAddress(address _newTaxAddress) public onlyOwner {
        _setTaxAddress(_newTaxAddress);
    }

    function setExclusionFromTaxFee(address _account, bool _status) public onlyOwner {
        _setExclusionFromTaxFee(_account, _status);
    }

    function _beforeTokenTransfer(address from, address to, uint256 amount) internal override whenNotPaused {
        require(!isAccountBlocked(to), "BlackList: Recipient account is blocked");
        require(!isAccountBlocked(from), "BlackList: Sender account is blocked");

        super._beforeTokenTransfer(from, to, amount);
    }
}
```

2 of 11

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.9.0) (access/Ownable.sol)

pragma solidity ^0.8.0;

import "../utils/Context.sol";

/**
 * @dev Contract module which provides a basic access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * By default, the owner account will be the one that deploys the contract. This
 * can later be changed with {transferOwnership}.
 *
 * This module is used through inheritance. It will make available the modifier
 * `onlyOwner`, which can be applied to your functions to restrict their use to
 * the owner.
 */
abstract contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    /**
     * @dev Initializes the contract setting the deployer as the initial owner.
     */
    constructor() {
        _transferOwnership(_msgSender());
```

```
    }

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        _checkOwner();
        _;
    }

    /**
     * @dev Returns the address of the current owner.
     */
    function owner() public view virtual returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if the sender is not the owner.
     */
    function _checkOwner() internal view virtual {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
    }

    /**
     * @dev Leaves the contract without owner. It will not be possible to call
     * `onlyOwner` functions. Can only be called by the current owner.
     *
     * NOTE: Renouncing ownership will leave the contract without an owner,
     * thereby disabling any functionality that is only available to the owner.
     */
    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));
    }

    /**
     * @dev Transfers ownership of the contract to a new account (`newOwner`).
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);
    }

    /**
     * @dev Transfers ownership of the contract to a new account (`newOwner`).
     * Internal function without access restriction.
     */
    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```



```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.9.0) (proxy/utils/Initializable.sol)

pragma solidity ^0.8.2;

import "../../utils/Address.sol";
```

```
/**
 * @dev This is a base contract to aid in writing upgradeable contracts, or any kind of contract that will be deployed
 * behind a proxy. Since proxied contracts do not make use of a constructor, it's common to move constructor logic to an
 * external initializer function, usually called `initialize`. It then becomes necessary to protect this initializer
 * function so it can only be called once. The {initializer} modifier provided by this contract will have this effect.
 *
 * The initialization functions use a version number. Once a version number is used, it is consumed and cannot be
 * reused. This mechanism prevents re-execution of each "step" but allows the creation of new initialization steps in
 * case an upgrade adds a module that needs to be initialized.
 *
 * For example:
 *
 * [.hljs-theme-light.nopadding]
 * ```solidity
 * contract MyToken is ERC20Upgradeable {
 *     function initialize() initializer public {
 *         __ERC20_init("MyToken", "MTK");
 *     }
 * }
 *
 * contract MyTokenV2 is MyToken, ERC20PermitUpgradeable {
 *     function initializeV2() reinitializer(2) public {
 *         __ERC20Permit_init("MyToken");
 *     }
 * }
 * ```
 *
 * TIP: To avoid leaving the proxy in an uninitialized state, the initializer function should be called as early as
 * possible by providing the encoded function call as the `_data` argument to {ERC1967Proxy-constructor}.
 *
 * CAUTION: When used with inheritance, manual care must be taken to not invoke a parent initializer twice, or to
ensure
 * that all initializers are idempotent. This is not verified automatically as constructors are by Solidity.
 *
 * [CAUTION]
 * ====
 * Avoid leaving a contract uninitialized.
 *
 * An uninitialized contract can be taken over by an attacker. This applies to both a proxy and its implementation
 * contract, which may impact the proxy. To prevent the implementation contract from being used, you should invoke
 * the {_disableInitializers} function in the constructor to automatically lock it when it is deployed:
 *
 * [.hljs-theme-light.nopadding]
 * ```
 * /// @custom:oz-upgrades-unsafe-allow constructor
 * constructor() {
 *     _disableInitializers();
 * }
 * ```
 * ====
 */
abstract contract Initializable {
    /**
     * @dev Indicates that the contract has been initialized.
     * @custom:oz-retyped-from bool
     */
    uint8 private _initialized;

    /**
     * @dev Indicates that the contract is in the process of being initialized.
     */
    bool private _initializing;
```

```
/**
 * @dev Triggered when the contract has been initialized or reinitialized.
 */
event Initialized(uint8 version);

/**
 * @dev A modifier that defines a protected initializer function that can be invoked at most once. In its scope,
 * `onlyInitializing` functions can be used to initialize parent contracts.
 *
 * Similar to `reinitializer(1)`, except that functions marked with `initializer` can be nested in the context of a
 * constructor.
 *
 * Emits an {Initialized} event.
 */
modifier initializer() {
    bool isTopLevelCall = !_initializing;
    require(
        (isTopLevelCall && _initialized < 1) || (!Address.isContract(address(this)) && _initialized == 1),
        "Initializable: contract is already initialized"
    );
    _initialized = 1;
    if (isTopLevelCall) {
        _initializing = true;
    }
    _;
    if (isTopLevelCall) {
        _initializing = false;
        emit Initialized(1);
    }
}

/**
 * @dev A modifier that defines a protected reinitializer function that can be invoked at most once, and only if the
 * contract hasn't been initialized to a greater version before. In its scope, `onlyInitializing` functions can be
 * used to initialize parent contracts.
 *
 * A reinitializer may be used after the original initialization step. This is essential to configure modules that
 * are added through upgrades and that require initialization.
 *
 * When `version` is 1, this modifier is similar to `initializer`, except that functions marked with `reinitializer`
 * cannot be nested. If one is invoked in the context of another, execution will revert.
 *
 * Note that versions can jump in increments greater than 1; this implies that if multiple reinitializers coexist in
 * a contract, executing them in the right order is up to the developer or operator.
 *
 * WARNING: setting the version to 255 will prevent any future reinitialization.
 *
 * Emits an {Initialized} event.
 */
modifier reinitializer(uint8 version) {
    require(!_initializing && _initialized < version, "Initializable: contract is already initialized");
    _initialized = version;
    _initializing = true;
    _;
    _initializing = false;
    emit Initialized(version);
}

/**
 * @dev Modifier to protect an initialization function so that it can only be invoked by functions with the
 * {initializer} and {reinitializer} modifiers, directly or indirectly.
 */
```

```
    modifier onlyInitializing() {
        require(_initializing, "Initializable: contract is not initializing");
        _;
    }

    /**
     * @dev Locks the contract, preventing any future reinitialization. This cannot be part of an initializer call.
     * Calling this in the constructor of a contract will prevent that contract from being initialized or reinitialized
     * to any version. It is recommended to use this to lock implementation contracts that are designed to be called
     * through proxies.
     *
     * Emits an {Initialized} event the first time it is successfully executed.
     */
    function _disableInitializers() internal virtual {
        require(!_initializing, "Initializable: contract is initializing");
        if (_initialized != type(uint8).max) {
            _initialized = type(uint8).max;
            emit Initialized(type(uint8).max);
        }
    }

    /**
     * @dev Returns the highest version that has been initialized. See {reinitializer}.
     */
    function _getInitializedVersion() internal view returns (uint8) {
        return _initialized;
    }

    /**
     * @dev Returns `true` if the contract is currently initializing. See {onlyInitializing}.
     */
    function _isInitializing() internal view returns (bool) {
        return _initializing;
    }
}
```



```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.7.0) (security/Pausable.sol)

pragma solidity ^0.8.0;

import "../utils/Context.sol";

/**
 * @dev Contract module which allows children to implement an emergency stop
 * mechanism that can be triggered by an authorized account.
 *
 * This module is used through inheritance. It will make available the
 * modifiers `whenNotPaused` and `whenPaused`, which can be applied to
 * the functions of your contract. Note that they will not be pausable by
 * simply including this module, only once the modifiers are put in place.
 */
abstract contract Pausable is Context {
    /**
     * @dev Emitted when the pause is triggered by `account`.
     */
    event Paused(address account);

    /**
     * @dev Emitted when the pause is lifted by `account`.
     */
```

```
event Unpaused(address account);

bool private _paused;

/**
 * @dev Initializes the contract in unpaused state.
 */
constructor() {
    _paused = false;
}

/**
 * @dev Modifier to make a function callable only when the contract is not paused.
 *
 * Requirements:
 *
 * - The contract must not be paused.
 */
modifier whenNotPaused() {
    _requireNotPaused();
    _;
}

/**
 * @dev Modifier to make a function callable only when the contract is paused.
 *
 * Requirements:
 *
 * - The contract must be paused.
 */
modifier whenPaused() {
    _requirePaused();
    _;
}

/**
 * @dev Returns true if the contract is paused, and false otherwise.
 */
function paused() public view virtual returns (bool) {
    return _paused;
}

/**
 * @dev Throws if the contract is paused.
 */
function _requireNotPaused() internal view virtual {
    require(!paused(), "Pausable: paused");
}

/**
 * @dev Throws if the contract is not paused.
 */
function _requirePaused() internal view virtual {
    require(paused(), "Pausable: not paused");
}

/**
 * @dev Triggers stopped state.
 *
 * Requirements:
 *
 * - The contract must not be paused.
 */
```

```
    function _pause() internal virtual whenNotPaused {
        _paused = true;
        emit Paused(_msgSender());
    }

    /**
     * @dev Returns to normal state.
     *
     * Requirements:
     *
     * - The contract must be paused.
     */
    function _unpause() internal virtual whenPaused {
        _paused = false;
        emit Unpaused(_msgSender());
    }
}
```

5 of 11

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (token/ERC20/extensions/IERC20Metadata.sol)

pragma solidity ^0.8.0;

import "../IERC20.sol";

/**
 * @dev Interface for the optional metadata functions from the ERC20 standard.
 *
 * _Available since v4.1._
 */
interface IERC20Metadata is IERC20 {
    /**
     * @dev Returns the name of the token.
     */
    function name() external view returns (string memory);

    /**
     * @dev Returns the symbol of the token.
     */
    function symbol() external view returns (string memory);

    /**
     * @dev Returns the decimals places of the token.
     */
    function decimals() external view returns (uint8);
}
```

6 of 11

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.9.0) (token/ERC20/IERC20.sol)

pragma solidity ^0.8.0;

/**
 * @dev Interface of the ERC20 standard as defined in the EIP.
 */
interface IERC20 {
    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
```

```
 * another (`to`).
 *
 * Note that `value` may be zero.
 */
event Transfer(address indexed from, address indexed to, uint256 value);

/**
 * @dev Emitted when the allowance of a `spender` for an `owner` is set by
 * a call to {approve}. `value` is the new allowance.
 */
event Approval(address indexed owner, address indexed spender, uint256 value);

/**
 * @dev Returns the amount of tokens in existence.
 */
function totalSupply() external view returns (uint256);

/**
 * @dev Returns the amount of tokens owned by `account`.
 */
function balanceOf(address account) external view returns (uint256);

/**
 * @dev Moves `amount` tokens from the caller's account to `to`.
 *
 * Returns a boolean value indicating whether the operation succeeded.
 *
 * Emits a {Transfer} event.
 */
function transfer(address to, uint256 amount) external returns (bool);

/**
 * @dev Returns the remaining number of tokens that `spender` will be
 * allowed to spend on behalf of `owner` through {transferFrom}. This is
 * zero by default.
 *
 * This value changes when {approve} or {transferFrom} are called.
 */
function allowance(address owner, address spender) external view returns (uint256);

/**
 * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
 *
 * Returns a boolean value indicating whether the operation succeeded.
 *
 * IMPORTANT: Beware that changing an allowance with this method brings the risk
 * that someone may use both the old and the new allowance by unfortunate
 * transaction ordering. One possible solution to mitigate this race
 * condition is to first reduce the spender's allowance to 0 and set the
 * desired value afterwards:
 * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
 *
 * Emits an {Approval} event.
 */
function approve(address spender, uint256 amount) external returns (bool);

/**
 * @dev Moves `amount` tokens from `from` to `to` using the
 * allowance mechanism. `amount` is then deducted from the caller's
 * allowance.
 *
 * Returns a boolean value indicating whether the operation succeeded.
 *
```

```
     * Emits a {Transfer} event.
     */
    function transferFrom(address from, address to, uint256 amount) external returns (bool);
}
```



```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.9.0) (utils/Address.sol)

pragma solidity ^0.8.1;

/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     *
     * Furthermore, `isContract` will also return true if the target contract within
     * the same transaction is already scheduled for destruction by `SELFDESTRUCT`,
     * which only has an effect at the end of a transaction.
     * ====
     *
     * [IMPORTANT]
     * ====
     * You shouldn't rely on `isContract` to protect against flash loan attacks!
     *
     * Preventing calls from contracts is highly discouraged. It breaks composability, breaks support for smart wallets
     * like Gnosis Safe, and does not provide security since it can be circumvented by calling from a contract
     * constructor.
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // This method relies on extcodesize/address.code.length, which returns 0
        // for contracts in construction, since the code is only stored at the end
        // of the constructor execution.

        return account.code.length > 0;
    }

    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
```

```
     *
     * https://consensys.net/diligence/blog/2019/09/stop-using-soliditys-transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.8.0/security-considerations.html#use-the-checks-effects-interactions-pattern[checks-effects-interactions pattern].
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        (bool success, ) = recipient.call{value: amount}("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }

    /**
     * @dev Performs a Solidity function call using a low level `call`. A
     * plain `call` is an unsafe replacement for a function call: use this
     * function instead.
     *
     * If `target` reverts with a revert reason, it is bubbled up by this
     * function (like regular Solidity function calls).
     *
     * Returns the raw returned data. To convert to the expected return value,
     * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
     *
     * Requirements:
     *
     * - `target` must be a contract.
     * - calling `target` with `data` must not revert.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data) internal returns (bytes memory) {
        return functionCallWithValue(target, data, 0, "Address: low-level call failed");
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
     * `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCall(
        address target,
        bytes memory data,
        string memory errorMessage
    ) internal returns (bytes memory) {
        return functionCallWithValue(target, data, 0, errorMessage);
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but also transferring `value` wei to `target`.
     *
     * Requirements:
     *
     * - the calling contract must have an ETH balance of at least `value`.
     * - the called Solidity function must be `payable`.
     *
     * _Available since v3.1._
```

```
 */
function functionCallWithValue(address target, bytes memory data, uint256 value) internal returns (bytes memory) {
    return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
}

/**
 * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but
 * with `errorMessage` as a fallback revert reason when `target` reverts.
 *
 * _Available since v3.1._
 */
function functionCallWithValue(
    address target,
    bytes memory data,
    uint256 value,
    string memory errorMessage
) internal returns (bytes memory) {
    require(address(this).balance >= value, "Address: insufficient balance for call");
    (bool success, bytes memory returndata) = target.call{value: value}(data);
    return verifyCallResultFromTarget(target, success, returndata, errorMessage);
}

/**
 * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
 * but performing a static call.
 *
 * _Available since v3.3._
 */
function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {
    return functionStaticCall(target, data, "Address: low-level static call failed");
}

/**
 * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
 * but performing a static call.
 *
 * _Available since v3.3._
 */
function functionStaticCall(
    address target,
    bytes memory data,
    string memory errorMessage
) internal view returns (bytes memory) {
    (bool success, bytes memory returndata) = target.staticcall(data);
    return verifyCallResultFromTarget(target, success, returndata, errorMessage);
}

/**
 * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
 * but performing a delegate call.
 *
 * _Available since v3.4._
 */
function functionDelegateCall(address target, bytes memory data) internal returns (bytes memory) {
    return functionDelegateCall(target, data, "Address: low-level delegate call failed");
}

/**
 * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
 * but performing a delegate call.
 *
 * _Available since v3.4._
 */
```

```
function functionDelegateCall(
    address target,
    bytes memory data,
    string memory errorMessage
) internal returns (bytes memory) {
    (bool success, bytes memory returndata) = target.delegatecall(data);
    return verifyCallResultFromTarget(target, success, returndata, errorMessage);
}

/**
 * @dev Tool to verify that a low level call to smart-contract was successful, and revert (either by bubbling
 * the revert reason or using the provided one) in case of unsuccessful call or if target was not a contract.
 *
 * _Available since v4.8._
 */
function verifyCallResultFromTarget(
    address target,
    bool success,
    bytes memory returndata,
    string memory errorMessage
) internal view returns (bytes memory) {
    if (success) {
        if (returndata.length == 0) {
            // only check isContract if the call was successful and the return data is empty
            // otherwise we already know that it was a contract
            require(isContract(target), "Address: call to non-contract");
        }
        return returndata;
    } else {
        _revert(returndata, errorMessage);
    }
}

/**
 * @dev Tool to verify that a low level call was successful, and revert if it wasn't, either by bubbling the
 * revert reason or using the provided one.
 *
 * _Available since v4.3._
 */
function verifyCallResult(
    bool success,
    bytes memory returndata,
    string memory errorMessage
) internal pure returns (bytes memory) {
    if (success) {
        return returndata;
    } else {
        _revert(returndata, errorMessage);
    }
}

function _revert(bytes memory returndata, string memory errorMessage) private pure {
    // Look for revert reason and bubble it up if present
    if (returndata.length > 0) {
        // The easiest way to bubble the revert reason is using memory via assembly
        /// @solidity memory-safe-assembly
        assembly {
            let returndata_size := mload(returndata)
            revert(add(32, returndata), returndata_size)
        }
    } else {
        revert(errorMessage);
    }
```

```
    }
}
```

8 of 11

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (utils/Context.sol)

pragma solidity ^0.8.0;

/**
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
}
```

9 of 11

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol";
import "@openzeppelin/contracts/proxy/utils/Initializable.sol";

contract ERC20 is Initializable, IERC20Metadata {
    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowances;

    uint256 private _totalSupply;
    uint256 private _maxSupply;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    function init(
        string memory name_,
        string memory symbol_,
        uint8 decimals_,
        uint256 maxSupply_
    ) internal onlyInitializing {
        _name = name_;
        _symbol = symbol_;
        _decimals = decimals_;

        _maxSupply = maxSupply_;
    }

    function name() public view virtual override returns (string memory) {
```

```
    return _name;
}

function symbol() public view virtual override returns (string memory) {
    return _symbol;
}

function decimals() public view virtual override returns (uint8) {
    return _decimals;
}

function totalSupply() public view virtual override returns (uint256) {
    return _totalSupply;
}

function maxSupply() public view virtual returns (uint256) {
    return _maxSupply;
}

function balanceOf(address account) public view virtual override returns (uint256) {
    return _balances[account];
}

function transfer(address to, uint256 amount) public virtual override returns (bool) {
    _transfer(msg.sender, to, amount);
    return true;
}

function allowance(address owner, address spender) public view virtual override returns (uint256) {
    return _allowances[owner][spender];
}

function approve(address spender, uint256 amount) public virtual override returns (bool) {
    _approve(msg.sender, spender, amount);
    return true;
}

function transferFrom(address from, address to, uint256 amount) public virtual override returns (bool) {
    _spendAllowance(from, msg.sender, amount);
    _transfer(from, to, amount);
    return true;
}

function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
    address owner = msg.sender;
    _approve(owner, spender, allowance(owner, spender) + addedValue);
    return true;
}

function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
    address owner = msg.sender;
    uint256 currentAllowance = allowance(owner, spender);
    require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
    unchecked {
        _approve(owner, spender, currentAllowance - subtractedValue);
    }

    return true;
}

function burn(uint256 amount) public virtual {
    _burn(msg.sender, amount);
}
```

```
function burnFrom(address account, uint256 amount) public virtual {
    _spendAllowance(account, msg.sender, amount);
    _burn(account, amount);
}

function _transfer(address from, address to, uint256 amount) internal virtual {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");

    _beforeTokenTransfer(from, to, amount);

    uint256 fromBalance = _balances[from];
    require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");
    unchecked {
        _balances[from] = fromBalance - amount;
    // Overflow not possible: the sum of all balances is capped by totalSupply, and the sum is preserved by
    // decrementing then incrementing.
        _balances[to] += amount;
    }

    emit Transfer(from, to, amount);

    _afterTokenTransfer(from, to, amount);
}

function _mint(address account, uint256 amount) internal virtual {
    require(_totalSupply + amount <= _maxSupply, "ERC20: cap exceeded");
    require(account != address(0), "ERC20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);

    _totalSupply += amount;
    unchecked {
    // Overflow not possible: balance + amount is at most totalSupply + amount, which is checked above.
        _balances[account] += amount;
    }
    emit Transfer(address(0), account, amount);

    _afterTokenTransfer(address(0), account, amount);
}

function _burn(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: burn from the zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
    unchecked {
        _balances[account] = accountBalance - amount;
    // Overflow not possible: amount <= accountBalance <= totalSupply.
        _totalSupply -= amount;
    }

    emit Transfer(account, address(0), amount);

    _afterTokenTransfer(account, address(0), amount);
}

function _approve(address owner, address spender, uint256 amount) internal virtual {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");
```

```
        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    function _spendAllowance(address owner, address spender, uint256 amount) internal virtual {
        uint256 currentAllowance = allowance(owner, spender);
        if (currentAllowance != type(uint256).max) {
            require(currentAllowance >= amount, "ERC20: insufficient allowance");
            unchecked {
                _approve(owner, spender, currentAllowance - amount);
            }
        }
    }

    function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual {}

    function _afterTokenTransfer(address from, address to, uint256 amount) internal virtual {}
}
```

10 of 11

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "./ERC20.sol";
import "@openzeppelin/contracts/proxy/utils/Initializable.sol";

/**
 * @title ERC20Taxable
 * @dev Extension of {ERC20} that adds a tax rate permille.
 */
abstract contract ERC20Taxable is Initializable, ERC20 {
    // the permille rate for taxable mechanism
    uint256 private _taxRate;

    // the deposit address for tax
    address private _taxAddress;

    mapping(address => bool) private _isExcludedFromTaxFee;

    /**
     * @dev Sets the value of the `_taxRate` and the `_taxAddress`.
     */
    function init(
        uint256 taxFeePerMille_,
        address taxAddress_
    ) internal onlyInitializing {
        _setTaxRate(taxFeePerMille_);
        _setTaxAddress(taxAddress_);
        _setExclusionFromTaxFee(msg.sender, true);
        _setExclusionFromTaxFee(taxAddress_, true);
    }

    /**
     * @dev Moves `amount` of tokens from `sender` to `recipient` minus the tax fee.
     * Moves the tax fee to a deposit address.
     *
     * Requirements:
     *
     * - `to` cannot be the zero address.
     * - the caller must have a balance of at least `amount`.
     */
```

```
function transfer(address to, uint256 amount) public virtual override returns (bool) {
    address owner = msg.sender;

    if (_taxRate > 0 && !(_isExcludedFromTaxFee[owner] || _isExcludedFromTaxFee[to])) {
        uint256 taxAmount = (amount * _taxRate) / 1000;

        if (taxAmount > 0) {
            _transfer(owner, _taxAddress, taxAmount);
            unchecked {
                amount -= taxAmount;
            }
        }
    }

    _transfer(owner, to, amount);

    return true;
}

/**
 * @dev Moves `amount` tokens from `from` to `to` minus the tax fee using the allowance mechanism.
 * `amount` is then deducted from the caller's allowance.
 * Moves the tax fee to a deposit address.
 *
 * Requirements:
 *
 * - `from` and `to` cannot be the zero address.
 * - `from` must have a balance of at least `amount`.
 * - the caller must have allowance for ``from``'s tokens of at least `amount`.
 */
function transferFrom(address from, address to, uint256 amount) public virtual override returns (bool) {
    address spender = msg.sender;
    _spendAllowance(from, spender, amount);

    if (_taxRate > 0 && !(_isExcludedFromTaxFee[from] || _isExcludedFromTaxFee[to])) {
        uint256 taxAmount = (amount * _taxRate) / 1000;

        if (taxAmount > 0) {
            _transfer(from, _taxAddress, taxAmount);
            unchecked {
                amount -= taxAmount;
            }
        }
    }

    _transfer(from, to, amount);

    return true;
}

/**
 * @dev Returns the permille rate for taxable mechanism.
 *
 * For each transfer the permille amount will be calculated and moved to deposit address.
 */
function taxFeePerMille() external view returns (uint256) {
    return _taxRate;
}

/**
 * @dev Returns the deposit address for tax.
 */
function taxAddress() external view returns (address) {
```

```
        return _taxAddress;
    }

    /**
     * @dev Returns the status of exclusion from tax fee mechanism for a given account.
     */
    function isExcludedFromTaxFee(address account) external view returns (bool) {
        return _isExcludedFromTaxFee[account];
    }

    /**
     * @dev Sets the amount of tax fee permille.
     *
     * WARNING: it allows everyone to set the fee. Access controls MUST be defined in derived contracts.
     *
     * @param taxFeePerMille_ The amount of tax fee permille
     */
    function _setTaxRate(uint256 taxFeePerMille_) internal virtual {
        require(taxFeePerMille_ < 1000, "ERC20Taxable: taxFeePerMille_ must be less than 1000");

        _taxRate = taxFeePerMille_;
    }

    /**
     * @dev Sets the deposit address for tax.
     *
     * WARNING: it allows everyone to set the address. Access controls MUST be defined in derived contracts.
     *
     * @param taxAddress_ The deposit address for tax
     */
    function _setTaxAddress(address taxAddress_) internal virtual {
        require(taxAddress_ != address(0), "ERC20Taxable: taxAddress_ cannot be the zero address");

        _taxAddress = taxAddress_;
    }

    /**
     * @dev Sets the exclusion status from tax fee mechanism (both sending and receiving).
     *
     * WARNING: it allows everyone to set the status. Access controls MUST be defined in derived contracts.
     *
     * @param account_ The address that will be excluded or not
     * @param status_ The status of exclusion
     */
    function _setExclusionFromTaxFee(address account_, bool status_) internal virtual {
        _isExcludedFromTaxFee[account_] = status_;
    }
}
```

11 of 11

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

abstract contract BlackList {
    mapping (address => bool) private _isBlackListed;

    /**
     * @dev Emitted when the `_account` blocked.
     */
    event BlockedAccount(address indexed _account);
```

```
    /**
     * @dev Emitted when the `_account` unblocked.
     */
    event UnblockedAccount(address indexed _account);

    function isAccountBlocked(address _account) public view returns (bool) {
        return _isBlackListed[_account];
    }

    /**
     * @dev Add account to black list.
     *
     * WARNING: it allows everyone to set the address. Access controls MUST be defined in derived contracts.
     *
     * @param _account The address to be blocked
     */
    function _blockAccount (address _account) internal virtual {
        require(!_isBlackListed[_account], "Blacklist: Account is already blocked");
        _isBlackListed[_account] = true;

        emit BlockedAccount(_account);
    }

    function _unblockAccount (address _account) internal virtual {
        require(_isBlackListed[_account], "Blacklist: Account is already unblocked");
        _isBlackListed[_account] = false;

        emit UnblockedAccount(_account);
    }
}
```

Settings

```
{
  "optimizer": {
    "enabled": true,
    "runs": 200
  },
  "evmVersion": "paris",
  "outputSelection": {
    "*": {
      "*": [
        "evm.bytecode",
        "evm.deployedBytecode",
        "devdoc",
        "userdoc",
        "metadata",
        "abi"
      ]
    }
  },
  "libraries": {}
}
```

Contract ABI


```
[{"inputs":[],"stateMutability":"nonpayable","type":"constructor"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"owner","type":"address"},{"indexed":true,"internalType":"address","name":"spender","type":"address"},{"indexed":false,"internalType":"uint256","name":"value","type":"uint256"}],"name":"Approval","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"_account","type":"address"}],"name":"BlockedAccount","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"internalType":"uint8","na
```

me":"version","type":"uint8"}],"name":"Initialized","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"previousOwner","type":"address"},{"indexed":true,"internalType":"address","name":"newOwner","type":"address"}],"name":"OwnershipTransferred","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"internalType":"address","name":"account","type":"address"}],"name":"Paused","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"from","type":"address"},{"indexed":true,"internalType":"address","name":"to","type":"address"},{"indexed":false,"internalType":"uint256","name":"value","type":"uint256"}],"name":"Transfer","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"_account","type":"address"}],"name":"UnblockedAccount","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"internalType":"address","name":"account","type":"address"}],"name":"Unpaused","type":"event"},{"inputs":[{"internalType":"address","name":"owner","type":"address"},{"internalType":"address","name":"spender","type":"address"}],"name":"allowance","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"spender","type":"address"},{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"approve","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"account","type":"address"}],"name":"balanceOf","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"_account","type":"address"}],"name":"blockAccount","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"burn","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"account","type":"address"},{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"burnFrom","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"decimals","outputs":[{"internalType":"uint8","name":"","type":"uint8"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"spender","type":"address"},{"internalType":"uint256","name":"subtractedValue","type":"uint256"}],"name":"decreaseAllowance","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"spender","type":"address"},{"internalType":"uint256","name":"addedValue","type":"uint256"}],"name":"increaseAllowance","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"_owner","type":"address"},{"internalType":"string","name":"_name","type":"string"},{"internalType":"string","name":"_symbol","type":"string"},{"internalType":"uint8","name":"_decimals","type":"uint8"},{"internalType":"uint256","name":"_initialSupply","type":"uint256"},{"internalType":"uint256","name":"_maxSupply","type":"uint256"},{"internalType":"uint256","name":"_taxFeePerMille","type":"uint256"},{"internalType":"address","name":"_taxAddress","type":"address"}],"name":"initialize","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"_account","type":"address"}],"name":"isAccountBlocked","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"account","type":"address"}],"name":"isExcludedFromTaxFee","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"maxSupply","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"to","type":"address"},{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"mint","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"name","outputs":[{"internalType":"string","name":"","type":"string"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"owner","outputs":[{"internalType":"address","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"pause","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"paused","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"renounceOwnership","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"_account","type":"address"},{"internalType":"bool","name":"_status","type":"bool"}],"name":"setExclusionFromTaxFee","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"_newTaxAddress","type":"address"}],"name":"setTaxAddress","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"uint256","name":"_newTaxFee","type":"uint256"}],"name":"setTaxRate","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"symbol","outputs":[{"internalType":"string","name":"","type":"string"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"taxAddress","outputs":[{"internalType":"address","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"taxFeePerMille","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"totalSupply","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"to","type":"address"},{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"transfer","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"from","type":"address"},{"internalType":"address","name":"to","type":"address"},{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"transferFrom","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"newOwner","type":"address"}],"name":"transferOwnership","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"_account","type":"address"}],"name":"unblockAccount","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"unpause","outputs":[],"stateMutability":"nonpayable","type":"function"}]

Appendix B:

Legal Disclaimers:

This white paper and all attached documents are being furnished to you solely for your own personal use and on a confidential basis. It may not be reproduced, redistributed or passed on, in whole or in part, to any other person. You should also be aware that the distribution of this white paper and the attached documents in certain jurisdictions may be restricted by law. Any persons receiving this white paper and the attached documents should inform themselves of and observe any such restrictions.

Important information and disclaimer
This white paper (the "white paper") has been produced by Secure AI AS (the "Company") exclusively for potential decetralised AI users in connection with the offering of SECAI Tokens (the "Tokens") by the Company (the "Offering"). By reading this white paper, you will be deemed to have agreed to the restrictions and terms included herein and acknowledged that you understand the legal and regulatory sanctions attached to the misuse, disclosure or improper circulation of the white paper. This white paper is strictly confidential and may not be redistributed, in whole or in part, to any other person. This white paper has not been reviewed by or registered with any public authority or token exchange and does not constitute a prospectus. The Managers of the Company have not independently verified any of the information contained herein through due diligence procedures or other investigations and no formal due diligence investigations have been carried out. By reading this white paper, you acknowledge that you will be solely responsible for your own assessment of the market and the market position of the Company and that you will conduct your own analysis and be solely responsible for forming your own view of the potential future performance of the businesses of the Company. This document contains certain forward-looking statements relating to the business, financial performance and results of the Company and/or the industry in which it operates. Forward-looking statements concern future circumstances and results and other statements that are not historical facts, sometimes identified by the words "believes", "expects", "predicts", "intends", “goals”, "projects", "plans", "estimates", "aims", "foresees", "anticipates", "targets", and similar expressions. The forward-looking statements contained in this white paper, including assumptions, opinions and views of the Company or cited from third party sources are solely opinions and forecasts which are subject to risks, uncertainties and other factors that may cause actual events to differ materially from any anticipated development. Neither the Company or any of its respective parent or subsidiary undertakings or any such person's officers or employees ("Representatives") provides any assurance that the assumptions underlying such forwardlooking statements are free from errors nor does any of them accept any responsibility for the future accuracy of the opinions expressed in this white paper or the actual occurrence of the forecasted developments. Neither the Company or the Managers assume any obligation, except as required by law, to update this white paper, including any forward-looking statements or to conform these forwardlooking statements to our actual results. This white paper does not constitute an offer to sell or a solicitation of an offer to buy any securities or utility tokens in any jurisdiction to any person to whom it is unlawful to make such an offer or solicitation in such jurisdiction. The contents of this white paper are not to be construed as financial, legal, business, PURCHASE, tax or other professional advice. You should consult with your own professional advisers for any such matter and advice. No white paper or warranty (express or implied) is made as to, and no reliance should be placed on, any information, including projections, estimates, targets and opinions, contained herein, and no liability whatsoever is accepted as to any errors, omissions or misstatements contained herein, and, accordingly, neither the Company, the Managers or any of its Representatives accepts any liability whatsoever arising directly or indirectly from the use of this document. Actual experience may differ, and those differences may be material. A PURCHASE IN THE TOKENS IS ONLY SUITABLE IF YOU HAVE SUFFICIENT KNOWLEDGE,

SOPHISTICATION AND EXPERIENCE IN AI, BLOCKCHAIN, FINANCIAL AND BUSINESS MATTERS TO BE CAPABLE OF EVALUATING THE MERITS AND RISKS OF AN PURCHASE DECISION RELATING TO THE ISSUER'S TOKENS, AND IF YOU ARE ABLE TO BEAR THE ECONOMIC RISK, AND TO WITHSTAND A COMPLETE LOSS, OF YOUR PURCHASE. A PURCHASE IN THE TOKENS INVOLVES RISK, AND SEVERAL FACTORS COULD CAUSE THE ACTUAL RESULTS, PERFORMANCE OR ACHIEVEMENTS OF THE COMPANY TO BE MATERIALLY DIFFERENT FROM ANY FUTURE RESULTS, PERFORMANCE OR ACHIEVEMENTS THAT MAY BE EXPRESSED OR IMPLIED BY STATEMENTS AND INFORMATION IN THIS WHITE PAPER, INCLUDING, AMONG OTHERS, RISKS OR UNCERTAINTIES ASSOCIATED WITH THE COMPANY'S BUSINESS, SEGMENTS, DEVELOPMENT, GROWTH MANAGEMENT, FINANCING, MARKET ACCEPTANCE AND RELATIONS WITH CUSTOMERS, AND, MORE GENERALLY, GENERAL ECONOMIC AND BUSINESS CONDITIONS, CHANGES IN DOMESTIC AND FOREIGN LAWS AND REGULATIONS, TAXES, CHANGES IN COMPETITION AND PRICING ENVIRONMENTS, FLUCTUATIONS IN CURRENCY EXCHANGE RATES AND INTEREST RATES AND OTHER FACTORS. SHOULD ONE OR MORE OF THESE RISKS OR UNCERTAINTIES MATERIALISE, OR SHOULD UNDERLYING ASSUMPTIONS PROVE INCORRECT, ACTUAL RESULTS MAY VARY MATERIALLY FROM THOSE DESCRIBED IN THIS WHITE PAPER. Neither the Company, the Managers, or any of their respective Representatives, has taken any actions to allow the distribution of this white paper in any jurisdiction where action would be required for such purposes. The white paper has not been registered with, or approved by, any public authority, stock exchange or regulated market. The distribution of this white paper, as well as any subscription, purchase, sale or transfer of securities of the Issuer may be restricted by law in certain jurisdictions, and the recipient of this white paper should inform itself about, and observe, any such restriction. Any failure to comply with such restrictions may constitute a violation of the laws of any such jurisdiction. None of the Company or the Managers, or any of their respective Representatives, shall have any responsibility or liability whatsoever (in negligence or otherwise) arising directly or in|directly from any violations of such restrictions. This announcement is not an offer for sale or purchase of securities in the United States or any other country. The securities referred to herein have not been registered under the U.S. Securities Act of 1933, as amended (the "U.S. Securities Act"), and may not be sold in the United States absent registration or pursuant to an exemption from registration under the U.S. Securities Act. The Company has not registered and does not intend to register its securities in the United States or to conduct a public offering of its securities in the United States. Any offer for sale or purchase of securities will be made by means of an offer document that may be obtained by certain qualified investors from the Company. Copies of this white paper are not being made and may not be distributed or sent into the United States, Canada, Australia, Japan or any other jurisdiction in which such distribution would be unlawful or would require registration or other measures. In any EEA Member State that has implemented Regulation (EU) 2017/1129 of the European Parliament and of the Council of 14 June 2017 on the prospectus to be published when tokens are offered to the public or admitted to trading on a regulated market (together with any applicable implementing measures in any member state, ("the Prospectus Regulation"), this communication is only addressed to and is only directed at qualified investors in that Member State within the meaning of the Prospectus Regulation. This white paper is only directed at (a) persons who are outside the United Kingdom, (b) PURCHASE professionals within the meaning of Article 19 of the Financial Services and Markets Act 2000 (Financial Promotion) Order 2005 (the "Order"), (c) persons falling within Article 49 (2) (a) to (d) of the Order, or (d) persons to whom any invitation or inducement to engage in PURCHASE activity can be communicated in circumstances where Section 21(1) of the Financial Services and Markets Act 2000 does not apply. The Company and/or their employees may hold shares or tokens or interests in the Company and may, as principal or agent, buy or sell such tokens. The Managers may have other financial interests in transactions involving these tokens. This white paper speaks as of 20. September 2024. There may have been changes in matters which affect

the Company subsequent to the date of this white paper. Neither the delivery of this white paper nor any further discussions of the Company with any of the recipients shall, under any circumstances, create any implication that there has been no change in the affairs of the Company since such date. Neither the Company nor the Managers assume any obligation to update or revise the white paper. This white paper is subject to Norwegian law, and any dispute arising in respect of this white paper is subject to the exclusive jurisdiction of Norwegian courts with Oslo city court (Nw: Oslo tingrett) as exclusive venue.

Risks in relation to Secure AI AS and the industry in which Secure AI AS operates:
• The Issuer's ability to raise new equity capital is limited.
• The Issuer is dependent on the continued desire of its employees to remain employees.
• The Issuer's business is inherently tied to the business of its employees.
• Changes in laws and regulation may have an adverse effect on Secure AI AS's profitability.
• Secure AI AS's international activities increase the compliance risks associated with economic and trade, sanctions imposed by the United States, the European Union and other jurisdictions.
• Changes in the domestic and international political environment may impact Secure AI AS's profitability.
• There is risk associated with the compliance with the terms of the Issuer's financing arrangements .
• Secure AI AS is exposed to risk due to increased competition.
• Loss of reputation may have a material adverse effect on the financial condition of Secure AI AS.
• Secure AI AS is dependent upon a successfully demand for its products.
• Secure AI AS's profitability is dependent upon the price of its products and market balance.
• Secure AI AS is dependent upon its suppliers.
• Increased costs may have a material adverse effect on the financial condition of Secure AI AS.
• Secure AI AS's is regulated by Norwegian laws and international laws, and if Secure AI AS is held liable for breaches of such law, it may have a material adverse effect on Secure AI AS's financial condition.
• There is operating risks related to the way Secure AI AS operates.
• Secure AI AS may not be able to maintain sufficient insurance coverage.
• Secure AI AS may not be able to adapt to technological change quickly enough to keep up with its competitors.
• Risks relating to litigation, disputes and claims.
• Secure AI AS is exposed to risks relating to cyber-attacks.
Risk related to financing and market risks:
• Secure AI AS may require additional financing in the future.
• Secure AI AS is exposed to liquidity risk.
• Secure AI AS generates income from other jurisdictions than Norway, and is therefore subject to foreign currency risk.
• Secure AI AS is exposed to credit risks.
Risks related to the Tokens:
• A trading market for the Tokens may not develop, and market prices may be volatile.
• Secure AI AS's ability to meet its payment obligations is dependent on its financial performance, and in the event Secure AI AS's future debts becomes due, its assets would be available to satisfy obligations under such debts before the Tokens.
• There are risks related to amendments to the terms and conditions of the Tokens.
• Individual Tokenholders do not have the right to take legal actions against the Issuer.
• Purchase of Tokens inherently involves risks related to the value of the Tokens.
• The transferability of the Tokens may be limited in certain jurisdictions.

Subscription Restrictions:

The SECAI utility tokens shall only be offered to (i) non-“U.S. persons” in “offshore transactions” (each as defined in Rule 902 of Regulation S under the U.S. Securities Act of 1933, as amended (the “Securities Act”)), and (ii) to a limited number of persons located in the United States, its territories and possessions that are reasonably believed to be “qualified institutional buyers” (“QIBs”) (as defined in Rule 144A under the Securities Act (“Rule 144A”)) in transactions meeting the requirements of Rule 144A or another exemption from the registration requirements of the Securities Act. In addition to the Application Agreement that each investor will be required to execute, each U.S. investor that wishes to purchase SECAI utility tokens will be required to execute and deliver to the Issuer a certification in a form to be provided by the Issuer stating, among other things, that the investor is a QIB. The SECAI utility tokens have not and will not be registered under the U.S. Securities Act, or under the laws of any other jurisdiction. The SECAI utility tokens may not be offered or sold within the United States to, or for the account or benefit of, any U.S. Person (as such terms are defined in regulations), except pursuant to an exemption from the registration requirements of the U.S. Securities Act. Failure to comply with these restrictions may constitute a violation of applicable securities legislation.

Transfer Restrictions:
Tokenholders located in the United States will not be permitted to transfer the SECAI utility tokens except (a) subject to an effective registration statement under the Securities Act, (b) to a person that the Tokenholder reasonably believes is a QIB within the meaning of Rule 144A that is purchasing for its own account, or the account of another QIB, to whom notice is given that the resale, pledge or other transfer may be made in reliance on Rule 144A, (c) outside the United States in accordance with Regulation S under the Securities Act in a transaction on a regulated Token exchange, and (d) pursuant to an exemption from registration under the Securities Act provided by Rule 144 there under (if available). The SECAI utility tokens may not, subject to applicable Canadian laws, be traded in Canada for a period of four months and a day from the date the SECAI utility tokens were originally issued.

Governing Law:
Norwegian law.