SolyzerX
BLOCKCHAIN SECURITY

CryptoGPT

Security Assesment

MARCH 2023

SolyzerX

SolyzerX.com

Prepared for:

Dejan Erja
CryptoGPT CTO

Prepared by:

Justin Aaron
SolyzerX Engineer

Introduction

Auditing Firm	SolyzerX
Client Firm	CryptoGPT
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0xD04E772BC0d591fBD288f2E2a86aFA3D3CB647F8
Blockchain	Ethereum
Centralization	Active Ownership
Website	https://www.cryptogpt.org/
Discord	https://discord.com/invite/cryptogpt
Telegram	http://t.me/CryptoGPTorg
Twitter	https://twitter.com/CryptoGPT_org
Report Date	March 11, 2023

Verify the authenticity of this report on our website:
https://solyzerx.com/projects/cryptogpt

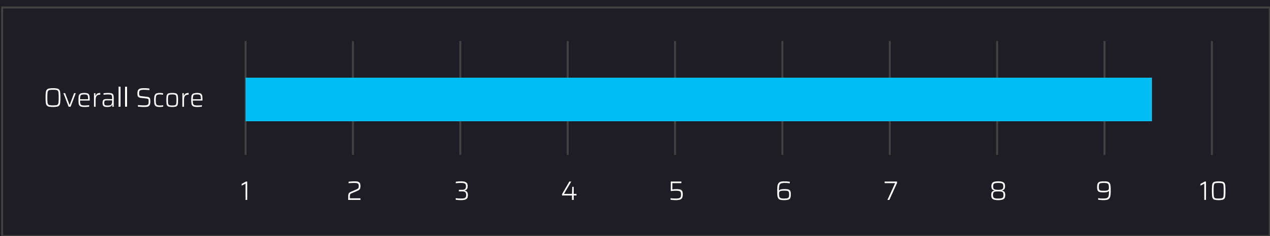
Executive Summary

SolyzerX has performed the automated and manual analysis of solidity codes. Solidity codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Severity	High	Medium	Low	Informational	Optimization
Count	0	0	4	10	0

Category	Denial of service	Data Validation	Arithmetic	Auditing and Logging	Undefined Behavior
Count	0	0	0	2	2

CryptoGPT smart contract source codes have achieved the following score: 9.4



Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

Please note that centralization priviledges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

Table of Contents

Scope of Work

SolyzerX volunteered to conduct a CryptoGPT (GPT) smart contract audit of their solidity source codes.

The audit scope of work is strictly limited to mentioned solidity file(s) only:

- CryptoGPT.sol

If source codes are not deployed on the main net, they can be modified or altered before main-net deployment. Verify the contract's deployment status below:

Public Contract Link	
https://etherscan.io/address/0xd04e772bc0d591fbd288f2e2a86afa3d3cb647f8#code	
Contract Name	CryptoGPT
Compiler Version	v0.8.9+commit.e5eed63a
License	MIT license

Audit Methodology

Smart contract audits are conducted using a set standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of SolyzerX's auditing process and methodology:

Connect

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

Audit

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
    - Remix IDE Developer Tool
    - Open Zeppelin Code Analyzer
    - Slither-SolyzerX
    - SWC Vulnerabilities Registry

- Simulations are performed to identify centralized exploits causing contract and/or trade locks.

- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	• Token Supply Manipulation • Access Control and Authorization • Assets Manipulation • Ownership Control • Liquidity Access • Stop and Pause Trading • Ownable Library Verification

Common Contract Vulnerabilities	• Integer Overflow • Lack of Arbitrary limits • Incorrect Inheritance Order • Typographical Errors • Requirement Violation • Gas Optimization • Coding Style Violations • Re-entrancy • Third-Party Dependencies • Potential Sandwich Attacks • Irrelevant Codes • Divide before multiply • Conformance to Solidity Naming Guides • Compiler Specific Warnings • Language Specific Warnings

Report

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to solidity codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

Publish

- The client may use the audit report internally or disclose it publicly.

It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

SolyzerX

Risk Categories

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to view:

Risk Type	Definition
High	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Medium	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Low	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Informational	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Undetermined	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed to mitigate the risk uncertainty.

All category breakdown which are identified in the audit report are categorized here for the reader to review:

Category Breakdown				
Denial of service	Data Validation	Arithmetic	Auditing and Logging	Undefined Behavior

Centralized Privileges

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- Privileged roles can be granted the power to `pause()` the contract in case of an external attack.
- Privileged roles can use functions like, `include()` , and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- The client can lower centralization-related risks by implementing below mentioned practices:
- Privileged role's private key must be carefully secured to avoid any potential hack.
- Privileged role should be shared by multi-signature (multi-sig) wallets.
- Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.
- Renouncing the contract ownership, and privileged roles.
- Remove functions with elevated centralization risk.

Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

Automated Analysis

Contract	Function	Visibility	Modifiers
ERC20	name	External	
	symbol	External	
	decimals	External	
	totalSupply	External	
	balanceOf	External	
	transfer	External	
	allowance	External	
	approve	External	
	transferFrom	External	
	_msgSender	Internal	
	_msgData	Internal	
	constructor	Public	
	name	Public	
	symbol	Public	
	decimals	Public	
	totalSupply	Public	
	balanceOf	Public	

	transfer	Public	
	allowance	Public	
	approve	Public	
	transferFrom	Public	
	increaseAllowance	Public	
	decreaseAllowance	Public	
	_transfer	Internal	
	_mint	Internal	
	_burn	Internal	
	_approve	Internal	
	_spendAllowance	Internal	
	_beforeTokenTransfer	Internal	
	_afterTokenTransfer	Internal	
IERC20	totalSupply	External	
	balanceOf	External	
	transfer	External	
	allowance	External	
	approve	External	

	transferFrom	External	
ERC20Burnable	constructor	Public	
	name	Public	
	symbol	Public	
	decimals	Public	
	totalSupply	Public	
	balanceOf	Public	
	transfer	Public	
	allowance	Public	
	approve	Public	
	transferFrom	Public	
	increaseAllowance	Public	
	decreaseAllowance	Public	
	_transfer	Internal	
	_mint	Internal	
	_burn	Internal	
	_approve	Internal	
	_spendAllowance	Internal	

	_beforeTokenTransfer	Internal	
	_afterTokenTransfer	Internal	
	name	External	
	symbol	External	
	decimals	External	
	totalSupply	External	
	balanceOf	External	
	transfer	External	
	allowance	External	
	approve	External	
	transferFrom	External	
	_msgSender	Internal	
	_msgData	Internal	
	burn	Public	
	burnFrom	Public	
IERC20Metadata	totalSupply	External	
	balanceOf	External	

	transfer	External	
	allowance	External	
	approve	External	
	transferFrom	External	
	name	External	
	symbol	External	
	decimals	External	
Context	_msgSender	Internal	
	_msgData	Internal	
BaseERC20	burn	Public	
	burnFrom	Public	
	constructor	Public	
	name	Public	
	symbol	Public	
	decimals	Public	
	totalSupply	Public	
	balanceOf	Public	

SolyzerX

	transfer	Public	
	allowance	Public	
	approve	Public	
	transferFrom	Public	
	increaseAllowance	Public	
	decreaseAllowance	Public	
	_transfer	Internal	
	_mint	Internal	
	_burn	Internal	
	_approve	Internal	
	_spendAllowance	Internal	
	_beforeTokenTransfer	Internal	
	_afterTokenTransfer	Internal	
	name	External	
	symbol	External	
	decimals	External	
	totalSupply	External	
	balanceOf	External	

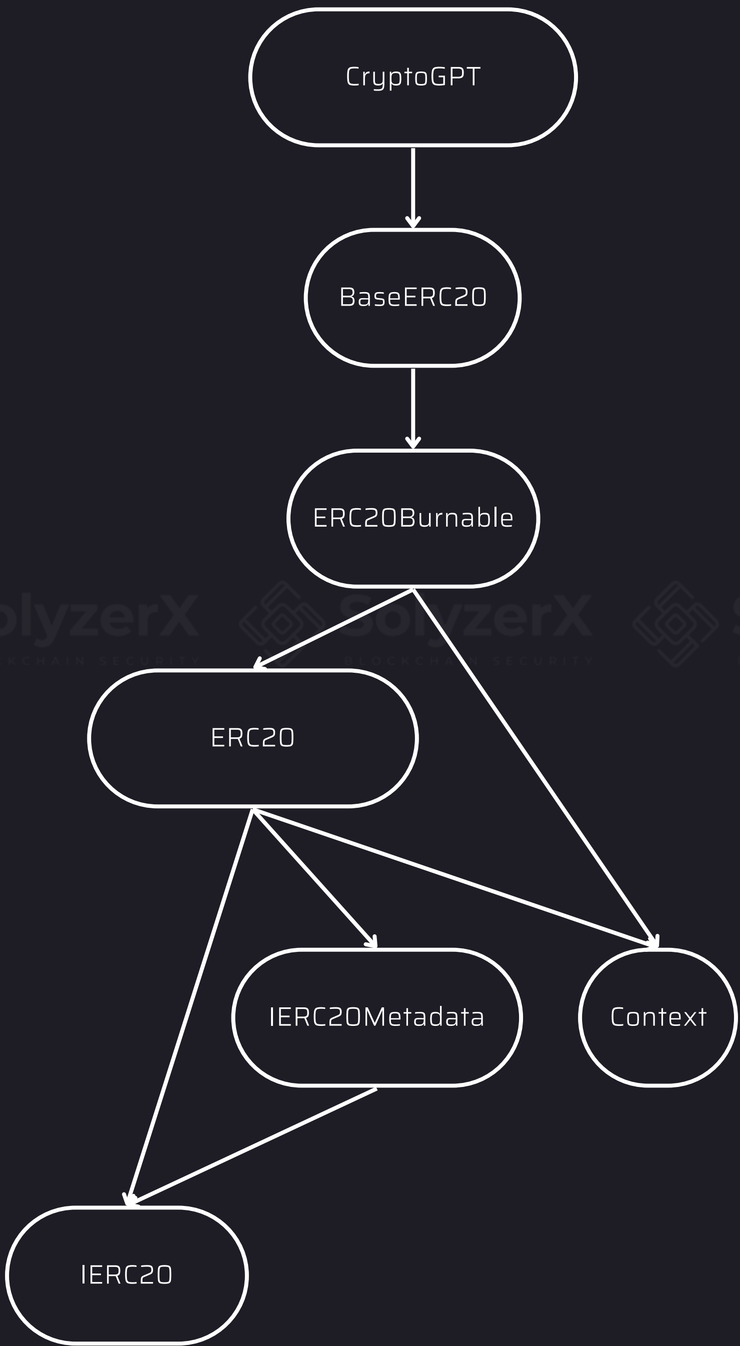
	transfer	External	
	allowance	External	
	approve	External	
	transferFrom	External	
	_msgSender	Internal	
	_msgData	Internal	
	constructor	Public	
CryptoGPT	constructor	Public	
	burn	Public	
	burnFrom	Public	
	constructor	Public	
	name	Public	
	symbol	Public	
	decimals	Public	
	totalSupply	Public	
	balanceOf	Public	
	transfer	Public	
	allowance	Public	

	approve	Public	
	transferFrom	Public	
	increaseAllowance	Public	
	decreaseAllowance	Public	
	_transfer	Internal	
	_mint	Internal	
	_burn	Internal	
	_approve	Internal	
	_spendAllowance	Internal	
	_beforeTokenTransfer	Internal	
	_afterTokenTransfer	Internal	
	name	External	
	symbol	External	
	decimals	External	
	totalSupply	External	
	balanceOf	External	
	transfer	External	
	allowance	External	
	approve	External	

	transferFrom	External	
	_msgSender	Internal	
	_msgData	Internal	
	constructor	Public	

SolyzerX

Inheritance Graph



Findings Summary

#	Title	Type	Severity
1	Local variable shadowing	Undefined Behavior	Low
2	Different pragma directives are used	Auditing and Logging	Informational
3	Dead-code	Undefined Behavior	Informational
4	Incorrect versions of Solidity	Auditing and Logging	Informational

Detailed Findings

1. Local variable shadowing	
Severity: Low	Difficulty: High
Type: Undefined Behavior	Finding ID: BaseERC20.sol#15&#16, CryptoGPT.sol#9&#10
Target: BaseERC20.sol & CryptoGPT.sol	

Description

BaseERC20.constructor(address,string,string,uint256).name (BaseERC20.sol#15) shadows:

- ERC20.name() (ERC20.sol#62-64) (function)
- IERC20Metadata.name() (IERC20Metadata.sol#17) (function)

BaseERC20.constructor(address,string,string,uint256).symbol (BaseERC20.sol#16) shadows:

- ERC20.symbol() (ERC20.sol#70-72) (function)
- IERC20Metadata.symbol() (IERC20Metadata.sol#22) (function)

CryptoGPT.constructor(address,string,string,uint256).name (CryptoGPT.sol#9) shadows:

- ERC20.name() (ERC20.sol#62-64) (function)
- IERC20Metadata.name() (IERC20Metadata.sol#17) (function)

CryptoGPT.constructor(address,string,string,uint256).symbol (CryptoGPT.sol#10) shadows:

- ERC20.symbol() (ERC20.sol#70-72) (function)
- IERC20Metadata.symbol() (IERC20Metadata.sol#22) (function)

Exploit Scenario:

```
pragma solidity ^0.4.24;

contract Bug {
    uint owner;

    function sensitive_function(address owner) public {
        // ...
        require(owner == msg.sender);
    }

    function alternate_sensitive_function() public {
        address owner = msg.sender;
        // ...
        require(owner == msg.sender);
    }
}
```

sensitive_function.owner shadows Bug.owner. As a result, the use of owner in sensitive_function might be incorrect.

Recommendation

Rename the local variables that shadow another component.

2. Different pragma directives are used	
Severity: Informational	Difficulty: High
Type: Auditing and Logging	Finding ID: ERC20.sol#4, ERC20Burnable.sol#4, IERC20Metadata.sol#4, Context.sol#4, BaseERC20.sol#2, CryptoGPT.sol#2
Target: ERC20.sol, ERC20Burnable.sol, IERC20Metadata.sol, Context.sol, BaseERC20.sol, CryptoGPT.sol	

Description

Different versions of Solidity are used:

- Version used: ['0.8.9', '^0.8.0']
- ^0.8.0 (ERC20.sol#4)
- ^0.8.0 (ERC20Burnable.sol#4)
- ^0.8.0 (IERC20Metadata.sol#4)
- ^0.8.0 (Context.sol#4)
- 0.8.9 (BaseERC20.sol#2)
- 0.8.9 (CryptoGPT.sol#2)

Recommendation

Use one Solidity version.

3. Dead-code	
Severity: **Informational**	Difficulty: **Medium**
Type: Undefined Behavior	Finding ID: Context.sol# 21-23
Target: Context.sol	

Description

Functions that are not sued.

Exploit Scenario:

```
contract Contract{
    function dead_code() internal() {}
}
```

dead_code is not used in the contract, and make the code's review more difficult.

Recommendation

Context._msgData() (Context.sol# 21-23) is never used and should be removed

SolyzerX

4. Incorrect versions of Solidity	
Severity: Informational	Difficulty: High
Type: Auditing and Logging	Finding ID: ERC20.sol# 4, IERC20.sol# 4, ERC20Burnable.sol# 4, IERC20Metadata.sol# 4, Context.sol# 4, BaseERC20.sol# 2, CryptoGPT.sol# 2
Target: ERC20.sol, IERC20.sol, ERC20Burnable.sol, IERC20Metadata.sol, Context.sol, BaseERC20.sol, CryptoGPT.sol	

Description

Pragma version^0.8.0 (ERC20.sol# 4) allows old versions
Pragma version^0.8.0 (IERC20.sol# 4) allows old versions
Pragma version^0.8.0 (ERC20Burnable.sol# 4) allows old versions
Pragma version^0.8.0 (IERC20Metadata.sol# 4) allows old versions
Pragma version^0.8.0 (Context.sol# 4) allows old versions
Pragma version0.8.9 (BaseERC20.sol# 2) allows old versions
Pragma version0.8.9 (CryptoGPT.sol# 2) allows old versions

solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statement.

Recommendation

solc-0.8.9 is not recommended for deployment.

Deploy with any of the following Solidity versions:

- 0.5.16 - 0.5.17
- 0.6.11 - 0.6.12
- 0.7.5 - 0.7.6
- 0.8.16

The recommendations take into account:

- Risks related to recent releases
- Risks of complex code generation changes
- Risks of new language features
- Risks of known bugs

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Disclaimers

SolyzerX provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high level of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

Confidentiality

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without SolyzerX's prior written consent.

No Financial Advice

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

SolyzerX

Technical Disclaimer

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULT OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, SOLYZERX HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, SOLYZERX SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, SOLYZERX MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

Timeliness Of Content

The content contained in this audit report is subject to change without any prior notice. SolyzerX does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

Links To Other Websites

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than SolyzerX. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such website's and social account's owners. You agree that SolyzerX is not responsible for the content or operation of such websites and social accounts and that SolyzerX shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

About SolyzerX

Founded in 2022 and headquartered in Malaysia, SolyzerX provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code.

We provide solidity development, testing, and auditing services. We work on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Velas, Oasis, etc.

SolyzerX is built by engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 4 core members, and 5+ casual contributors.

Website: https://solyzerx.com

Email: support@solyzerx.com

Github: https://github.com/SolyzerX

Telegram (Channel): https://t.me/SolyzerX

Telegram (Foundation): https://t.me/SolyzerXFoundation

solyzerx

solyzerx.com

t.me/solyzerx

t.me/solyzerxfoundation

SMART CONTRACT AUDITS | SOLIDITY DEVELOPMENT AND TESTING

RELENTLESSLY SECURING PUBLIC AND PRIVATE BLOCKCHAINS