THIS DOCUMENT IS AN UNFINISHED DRAFT

Contract Level Compliance

Whitepaper V1

contractlevel.com | github.com/contractlevel/compliance

Abstract

Contract Level Compliance (CLC) is a framework for implementing regulatory compliance in smart contracts. Built to address the inevitable demand for identification of onchain actors in accordance with global regulations, CLC enables protocols to securely, privately, and efficiently verify a user's compliant status before executing ***restricted*** logic on their behalf.

By integrating the *Everest identity oracle* [1], *Chainlink Automation* [2], and *Concero V2* [3], Contract Level Compliance ensures that protocols across chains can confidently execute regulatory-compliant smart contracts, whilst maintaining the highest degree of decentralisation possible.

This new generation of regulatory-compliant smart contracts will enable the creation of public chain applications that any onchain actor *and* traditional institution can use, marking an industry first in the combination of both markets.

1. Introduction

The era of blockchain and DeFi as entirely unregulated global markets is gradually, but firmly coming to an end. Regulators across the world have already begun introducing guidelines and enforcement measures for protocols and service providers in the industry [4]. The EU's MiCA regulation is law [5] and the US president signed an executive order to provide regulatory clarity for blockchain and digital assets [6].

In the real world, people are required to disclose certain information about themselves to be able to use traditional financial services [7] and engage in other regulated activities, such as buying and selling property. Centralised exchanges are already required to collect KYC data on their customers [8]. Onchain protocols will be no different—onchain actors will require identification [9].

Regulators will not accept KYC data from anyone. They require the entity facilitating the KYC to enact certain measures including, but not limited to, ongoing monitoring and risk rating [10] — tasks beyond the scope of most smart contract projects.

As smart contracts and blockchains are built upon the foundational idea of decentralisation [11], this poses a unique challenge. Regulatory compliance relies on centralised oversight and trusted intermediaries, whereas onchain protocols function autonomously, without a governing authority to enforce identity verification or risk assessments. This structural

incompatibility makes it difficult to implement compliance without introducing centralisation risks or compromising the permissionless nature of blockchain networks.

Contract Level Compliance addresses this challenge by enabling protocols to enforce regulatory compliance directly within smart contracts, using Chainlink, the industry standard for offchain computation [12], and Everest, a licensed data provider to the Chainlink network [13].

2. Architectural Overview

The core functional objective of Contract Level Compliance is to make executing compliant smart contracts as simple as possible. To achieve this, CLC is built upon four core principles that guide its design and operation:

- **Security** – The system must function securely to ensure smart contracts behave as intended (see section on *Security* for more details).
- **Decentralisation** - The Chainlink network must be used for offchain communication (requests for identity data) and computation (automated response handling) because it provides the securest way of performing these activities without compromising decentralisation [12].
- **Privacy Protection** - No personal user information must be revealed onchain; only a "yes" or "no" response to the question "Has this address completed KYC with a regulated entity?" is provided.
- **Legal Compliance** - Identity data must be sybil-resistant to ensure it cannot be transferred in blockchain environments, preventing money laundering and identity fraud. Additionally, the data must be issued by a licensed and regulated entity to maintain compliance [10].

To achieve these design principles, the system relies on three key integrations that form the foundation upon which the *Router* and *Logic* contracts operate.

2.1 Integrations

Everest, Chainlink, and Concero each play an important role in ensuring that Contract Level Compliance functions efficiently across chains. Everest provides sybil-resistant identity verification [13], Chainlink facilitates secure offchain computation [12], and Concero enables fast, cost-effective crosschain messaging [3].

2.1.1 Everest

Everest is a regulated provider of sybil-resistant identity data to smart contracts. It solves the deduplicated identity problem by using biometric data as a private key, locking each identity to a verifiably unique human, preventing transferability and therefore sybil attacks. As a licensed entity, Everest satisfies regulatory demands for facilitating KYC [14], which is crucial for legal compliance.

Users voluntarily submit KYC information through Everest's platform and retain ownership of their data.

2.1.2 Chainlink

Chainlink is the industry standard for offchain communication and computation for smart contracts. Chainlink has proven itself as a secure standard for offchain computation, securing most of the DeFi industry since its inception and servicing traditional institutions [12].

2.1.3 Concero

Concero improves upon Chainlink CCIP [29] by reducing fees and crosschain transaction times, whilst maintaining the security of CCIP as a settlement layer. Concero achieves this with crosschain pools and its own crosschain messaging layer (built with Chainlink Functions [16] and Eigenlayer [17]) for transaction execution.

2.2 Router

The CLC Router contract will be deployed on every Concero Compatible Chain (CCC) by the contractlevel.com team. This contract will be used to make requests for the compliance status of an onchain actor, and then, if the actor is compliant, route the response to the appropriate implementation of the *Logic* contract.

2.2.1 Requests

Requests are made to the Everest Chainlink node [1] for a boolean indicating whether an onchain address has been linked to the deduplicated identity of a person who has completed KYC.

Requests must contain the address of the onchain actor whose KYC status is being queried and the address of the custom *Logic* implementation contract for callback.

Requests can be made by EOAs or smart contracts.

A fee is taken for each request (see section on *Fees* for more info).

2.2.2 Automation

The Everest Chainlink Consumer contract emits a Fulfilled event when requests are fulfilled. Chainlink's offchain Automation nodes continuously monitor for this event [18]. If an emitted event matches a pending request, the Chainlink Automation Forwarder calls the Router contract with the fulfilled response, which is then routed to the appropriate Logic implementation if the requested address is compliant.

2.2.3 IERC677Receiver

The Router implements the ERC677 Receiver interface to enable requests in a single transaction using the LINK token's `transferAndCall()` functionality [15].

2.2.4 Upgradeability

The Router implements the TransparentUpgradeableProxy [19] to store Chainlink Automation variables such as the Forwarder address and Upkeep ID as immutable. This will save on gas from SLOADs [20].

2.3 Logic

The CLC Logic contract is abstract and must be inherited to implement Contract Level Compliance. Custom implementations of the CLC Logic contract can be made by anyone.

Logic contracts include an internal function that must be overridden, where the code for a regulated activity or otherwise handling of a compliant-verified address may be defined. This internal function is called by an external function that can only be called by the Router. When a request is fulfilled and the onchain actor is compliant, this function is passed the requested address.

2.3.1 IERC165

Logic contracts implement the ERC165 standard interface [21], allowing the Router to check that the Logic address to call does indeed implement the expected Logic interface.

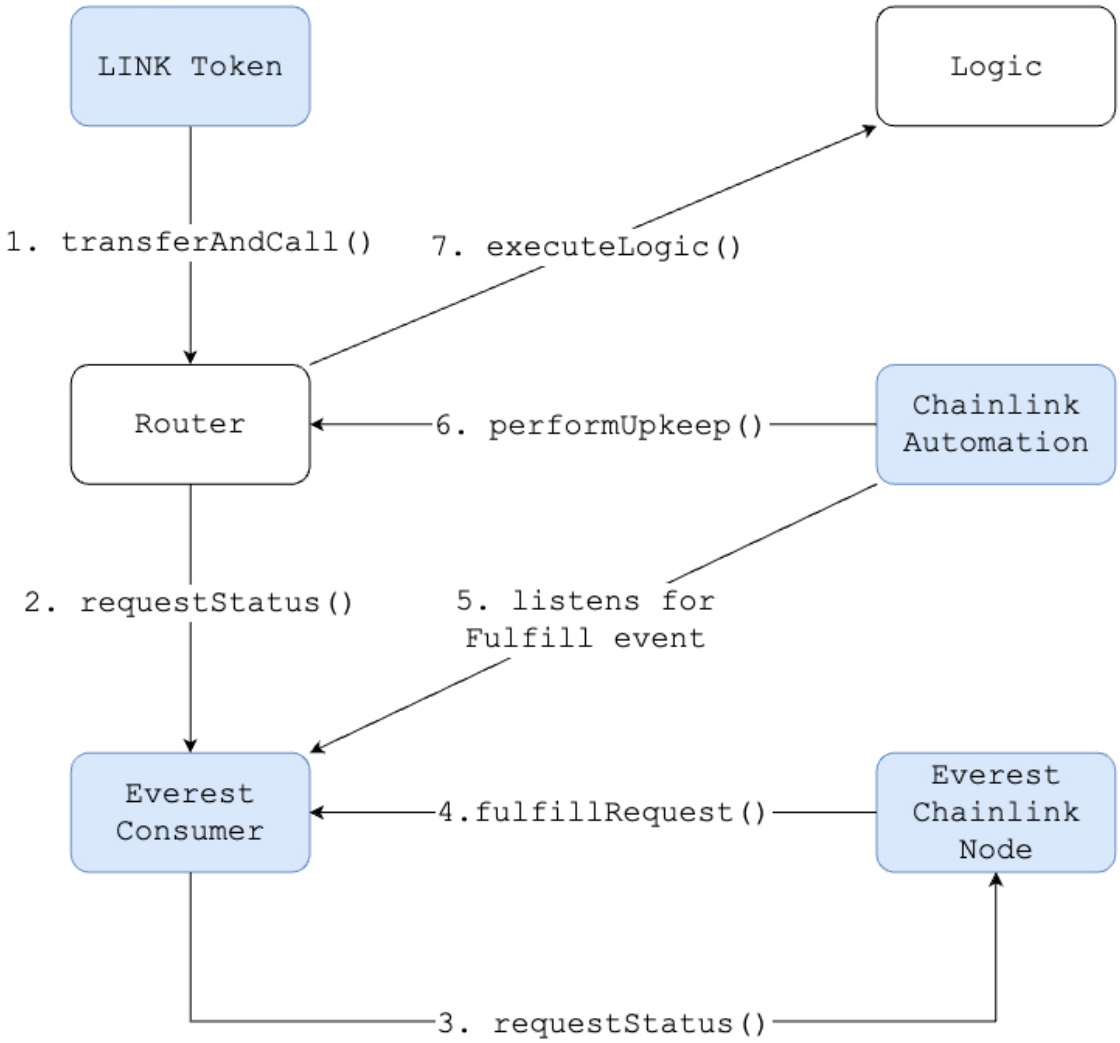
2.4 Calltrace

Requests for the compliant status of onchain actors and an automated callback passing the fulfilled response to a custom logic contract is the intended functionality for external use, and can be executed in a single transaction.

The CLC Router interacts with a few external services from the *Internet of Contracts* [22] (as well as the CLC Logic contract).

1. **LINK Token** - LINK is used to request the compliant status of an onchain actor. `transferAndCall()` can be used for single-transaction requests, or requests can be made directly from the Router after approving the token.
2. **Router** - The Router updates its state and forwards the request to the Everest Consumer.
3. **Everest Consumer** - The Everest Chainlink Consumer makes a request to the Everest Chainlink node.
4. **Everest Chainlink Node** - The Everest Chainlink node fulfills the request and sends the response back to the Everest Consumer.
5. **Chainlink Automation** - Chainlink offchain Automation nodes monitor the Everest Consumer for fulfilled request events. If a pending request exists, the Chainlink Automation Forwarder calls `performUpkeep()` on the Router.
6. **Router** - The Router updates its state and forwards the fulfilled response to the appropriate Logic contract.

7. **Logic** - If the onchain actor has completed KYC, the restricted logic executes on their behalf.



2.5 Crosschain Interoperability

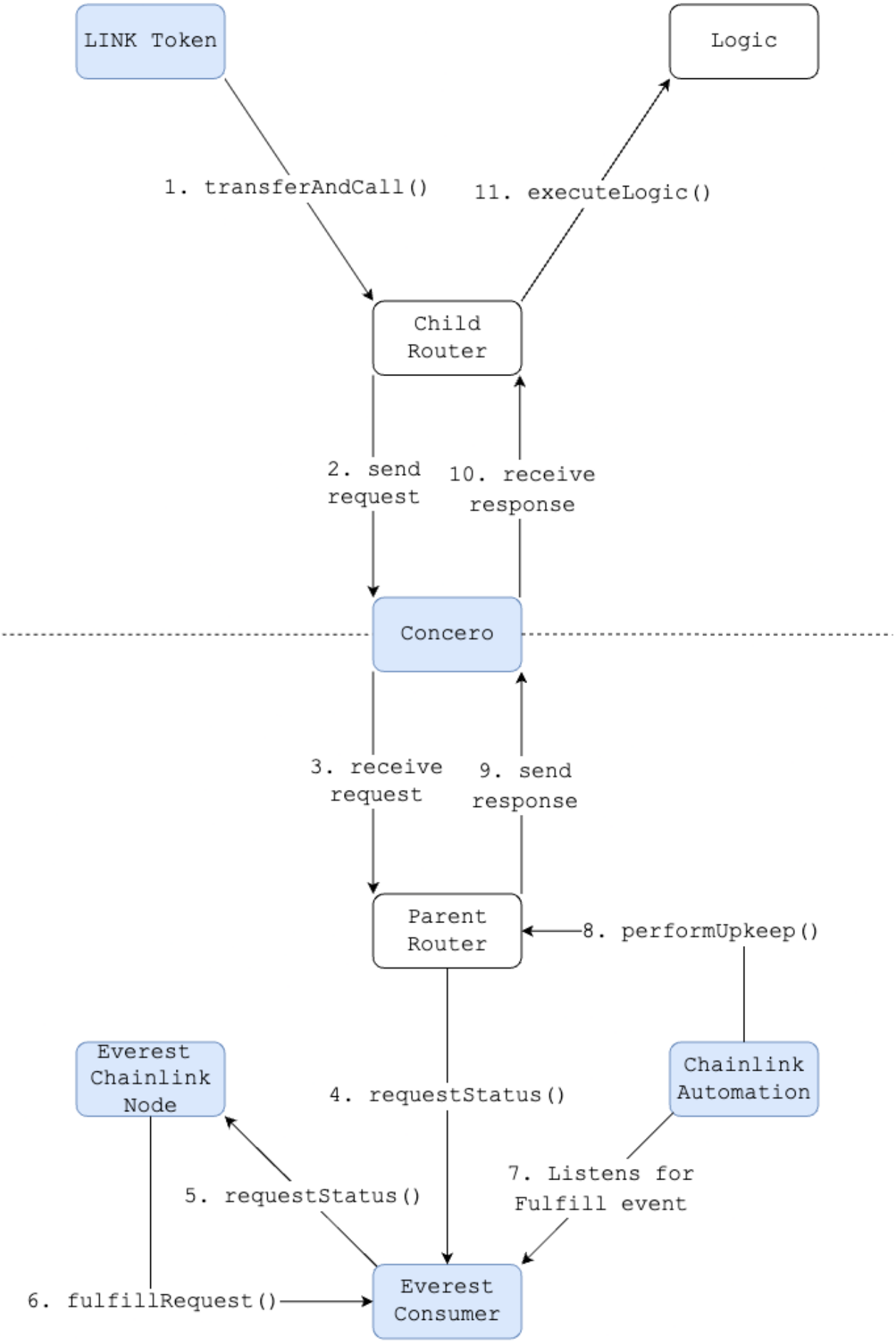
The Everest Chainlink node only needs to provide identity data to a single Concero Compatible Chain for Contract Level Compliance to be available on every CCC.

A “Parent” Router will be deployed on the chain where the Everest identity data is available and any requests made on other chains will be made to a “Child” Router on that chain. The Child Router will interact with the Concero infrastructure to route the request (and fee in LINK) to the Parent Router, which will route the fulfilled response back to the appropriate Child and Logic implementation.

2.5.1 Crosschain Transaction Flow/Calltrace

The transaction flow for a crosschain request and logic callback introduces additional steps of sending and receiving messages between “Child” and “Parent” Routers, through the Concero infrastructure.

LINK will need to be sent from the Child chain to the Parent chain in order to pay for the Automation and Node request services. Alternatively, a payment system allowing users to


LINK Token
Logic
1. transferAndCall()
11. executeLogic()
Child Router
2. send request
10. receive response
Concero
3. receive request
9. send response
Parent Router
8. performUpkeep()
Everest Chainlink Node
Chainlink Automation
4. requestStatus()
5. requestStatus()
7. Listens for Fulfill event
6. fulfillRequest()
Everest Consumer

initiate requests in other tokens that get swapped to LINK on the parent chain could be used.

3. Security

The highest priority for Contract Level Compliance is, and always will be, security. Smart contracts must behave as intended, and never behave in any way that is not intended. This is critical because CLC manages sensitive compliance and identity data onchain, where a vulnerability could falsify a user's compliance status or disrupt the system. The following measures uphold this priority, detailing the rigorous processes that secure CLC's operation and enable its use in regulated environments.

3.1 Testing

All Contract Level smart contracts undergo a comprehensive testing process to ensure their security and reliability. This includes:

- **Unit Tests:** Ensure 100% code coverage, using realistic scenarios like forked mainnets and randomized inputs.
- **Integration Tests:** Verify interactions between contracts, external dependencies, and deployment processes.
- **Static Analysis:** Detect vulnerabilities and logical errors without executing the code.
- **Invariant Tests:** Confirm that key properties—such as restricted logic only executing for KYC-verified actors—always hold true.
- **Formal Verification:** Mathematically prove contract correctness under all conditions using advanced tools.
- **Mutation Tests:** Introduce artificial bugs to strengthen test suites and specifications.
- **Test Networks:** Deploy and evaluate behaviour in real-world conditions.

Invariants are particularly important [26], as they define the system's unbreakable rules. For CLC, the foundational invariant is that restricted logic must only execute for actors who have completed KYC with Everest. Formal verification provides mathematical proof that these invariants are always maintained, offering the highest level of assurance.

3.2 Audits and Bug Bounty

Before deploying to production, CLC must undergo multiple independent security reviews from reputable firms, followed by competitive audits that engage the wider security community. These processes ensure that the contracts are thoroughly vetted for vulnerabilities and correctness. It is unacceptable to deploy unaudited smart contracts, as vulnerabilities are far more likely without these checks [23].

Post-deployment, a bug bounty program must be established to incentivise ethical hackers to continuously test the system's security [25]. This crowdsourced approach helps identify and

fix any overlooked vulnerabilities before they can be exploited, providing an ongoing layer of defense.

3.3 Gas Limit

Gas limits [27] are enforced on callbacks to prevent gasbombing or denial-of-service attacks, with configurable limits within defined bounds.

3.4 Multisig

A Safe [24] multisig wallet manages administrative functions like upgrades and fee withdrawals, preventing a single point of failure.

4. Fees

Each request takes a fee, covering the costs of external services provided by Everest, Chainlink and Concero, as well as the Contract Level protocol itself.

Fees are paid in LINK. Payment abstraction allowing requests to be paid in native or alternative tokens could be introduced in a later iteration.

Fee Type	Purpose	Applies To
Everest	Identity data request	All requests
Chainlink Automation	Upkeep execution costs	All requests
Concero	Crosschain transactions	Crosschain requests
Contract Level	Development and maintenance	All requests

4.1 Everest Fee

Covers the cost of requesting KYC status from the Everest Chainlink node.

4.2 Chainlink Automation Fee

Covers the minimum balance required for the associated Chainlink Automation upkeep to perform.

4.3 Concero Fee

Covers the cost of crosschain interactions facilitated by the Concero infrastructure. This fee is only taken for crosschain requests.

4.4 Contract Level Fee

Covers the costs of ongoing development and maintenance of the Contract Level ecosystem.

5. Future Developments

Once everything that has been laid out so far in this paper has been achieved, there will be ongoing developments to the Contract Level ecosystem.

5.1 Compliant Use Cases

The Contract Level *Compliance* infrastructure has been built from the beginning with modularity and usecase in mind. The Contract Level team will build new smart contracts on top of the CLC system. The exact use cases, and in which order they are built, are yet to be determined, but could include compliant insurance contracts, compliant derivatives contracts or compliant "wrappers" for, or forks of, popular smart contract protocols.

5.2 Native Token

There are currently no plans to deploy a native Contract Level token, and its specifics are yet to be determined. However, such a token would follow the ERC677 [15] and CCT [28] standards, with a fixed max supply across chains. The purpose of the token *could* be Contract Level revenue distribution.

6. Conclusion

We have proposed Contract Level Compliance (CLC), a framework for implementing regulatory compliance directly within smart contracts. CLC enables protocols to enforce compliance while preserving the core principles of decentralisation and reliability inherent to blockchain. As regulatory scrutiny of blockchain and DeFi grows, this framework addresses the challenge of aligning legal requirements with blockchain's autonomous nature by embedding compliance at the contract level.

Its modular design supports a broad spectrum of compliant use cases across chains, underpinned by rigorous measures that ensure dependable execution in regulated environments. Guided by security, decentralisation, privacy protection, and legal compliance, CLC offers a viable path for protocols to meet global legal standards without sacrificing blockchain's foundational strengths.

References

[1] Everest Identity Oracle - https://developer.everest.org/#everest-identity-oracle

[2] Chainlink Automation - https://docs.chain.link/chainlink-automation

[3] Concero V2 - https://blog.okron.cc/posts/concero-v2-technical-overview

[4] FSB Global Regulatory Framework for Crypto-asset Activities - https://www.fsb.org/2023/07/fsb-global-regulatory-framework-for-crypto-asset-activities

[5] MiCA - https://www.esma.europa.eu/esmas-activities/digital-finance-and-innovation/markets-crypto-assets-regulation-mica

[6] US Presidential Executive Order - https://www.whitehouse.gov/presidential-actions/2025/01/strengthening-american-leadership-in-digital-financial-technology

[7] Identification requirement in traditional activity - https://www.trulioo.com/industries/crypto-identity-verification/kyc

- https://www.fca.org.uk/firms/financial-crime/money-laundering-regulations

[8] Centralised exchanges required to adhere to KYC regulations - https://notabene.id/crypto-travel-rule-101/kyc-crypto

[9] Onchain identification requirements - https://public-inspection.federalregister.gov/2024-30496.pdf

- https://www.forbes.com/sites/shehanchandrasekera/2024/12/27/understanding-the-new-irs-defi-broker-tax-regulations
- https://blockworks.co/news/bipartisan-bill-defi-project-controllers

[10] Requirements for regulated entities facilitating KYC - https://www.fca.org.uk/firms/financial-crime/money-laundering-regulations

[11] Decentralisation and self-custody in smart contracts and blockchain - https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf

- https://bitcoin.org/bitcoin.pdf

[12] Chainlink as an industry standard - https://www.swift.com/news-events/press-releases/swift-ubs-asset-management-and-chainlink-successfully-complete-innovative-pilot-bridge-tokenized-assets-existing-payment-systems

[13] Everest as only provider of sybil resistant identity data - https://everestdotorg.medium.com/everest-is-now-a-data-provider-on-the-chainlink-network-bringing-novel-identity-data-to-377039492189

[14] Everest as licensed entity - https://www.businesswire.com/news/home/20210603005776/en/Everest-Secures-VFA-License-to-Provide-Regulated-DeFi-Globally

[15] ERC677 standard with transferAndCall() - https://github.com/ethereum/EIPs/issues/677

[16] Chainlink Functions - https://docs.chain.link/chainlink-functions

[17] Eigenlayer - https://docs.eigenlayer.xyz/

[18] Chainlink Log Trigger Automation - https://docs.chain.link/chainlink-automation/guides/log-trigger

[19] Transparent Upgradeable Proxy - https://docs.openzeppelin.com/contracts/4.x/api/proxy#TransparentUpgradeableProxy

[20] SLOAD - https://www.evm.codes/?fork=cancun#54

[21] ERC165 Standard Interface detection - https://eips.ethereum.org/EIPS/eip-165

[22] Internet of Contracts - https://blog.chain.link/from-tcpip-to-ccip/

[23] Unaudited vs audited smart contract hacks - https://rekt.news/leaderboard/

[24] Safe - https://docs.safe.global/home/what-is-safe

[25] Bug bounty efficacy - https://www.researchgate.net/publication/370621884_Predicting_the_Effectiveness_of_Blockchain_Bug_Bounty_Programs

[26] Importance of invariants - https://dacian.me/find-highs-before-external-auditors-using-invariant-fuzz-testing#heading-thinking-in-invariants

[27] Gas limits - https://ethereum.org/en/developers/docs/gas/#what-is-gas-limit

[28] Cross-Chain Token - https://docs.chain.link/ccip/concepts/cross-chain-tokens

[29] CCIP - https://docs.chain.link/ccip/