Chainlink Confidential Compute Whitepaper

Lorenz Breidenbach Alex Coventry Siam Hussain
Yan Ji Dahlia Malkhi Christian Müller
Gregory Neven Chrysoula Stathakopoulou
Philipp Schindler Paweł Szałachowski*

Chainlink Labs

November 4, 2025

Abstract

Blockchains excel at integrity and availability but fall short on confidentiality, preventing many important applications handling sensitive data from running onchain. Existing approaches to decentralized confidential computation based on multi-party computation (MPC), fully homomorphic encryption (FHE), or zero-knowledge (ZK) proofs face performance or applicability limits, while trusted execution environments (TEEs) alone are too brittle to trust. This paper describes Chainlink Confidential Compute that combines TEEs with threshold cryptography to provide trust-minimized confidential computing as a decentralized service. The system enforces a need-to-know data minimization principle on TEEs, ensuring that a compromised TEE only learns the private inputs of requests assigned to it *after* it was compromised. It additionally features proactive key re-sharing of the master key and leverages cloud-hosted TEEs for performance and security. The system integrates seamlessly with the Chainlink Runtime Environment (CRE). We present its architecture, security model, and use cases—including confidential API access, private tokens, and privacy-preserving identity—–showing how Chainlink Confidential Compute makes practical decentralized confidential computing possible. Looking ahead, Chainlink Confidential Compute follows a progressive roadmap towards combining TEEs with zero-knowledge proofs, secure multi-party computation, and fully-homomorphic encryption, providing users with different choices depending on their preferred trust assumptions and performance needs.

*Authors in alphabetical order.

1 Introduction

Blockchains have proven remarkably effective at providing *integrity* and *availability* to a decentralized system. Once data is committed to a blockchain, it becomes practically immutable, and anyone can verify the correctness of the chain's history. Likewise, decentralized consensus and replication ensure that data remains , even if individual nodes fail or act maliciously.

In contrast, blockchains offer little in the way of *confidentiality*. By design, most blockchains embrace full transparency: every transaction and every contract state is public, and this openness is often celebrated as a feature rather than a flaw. Transparency enables verifiability, open participation, and broad trust without intermediaries. But it also means that sensitive business logic, financial data, or personal information cannot safely be processed onchain—ruling out a wide swath of applications that would otherwise be natural candidates for decentralized execution.

Existing approaches. Achieving decentralized confidentiality is hard. Several approaches have been explored, each with its own limitations:

- **Secure multi-party computation (MPC)** and **fully homomorphic encryption (FHE)** provide strong privacy guarantees in theory, but in practice they are complex to engineer and suffer from high performance overheads. This makes them challenging to deploy at scale for real-world blockchain applications.
- **Zero-knowledge (ZK) proofs** have become somewhat practical in recent years, but they are best suited for proving statements about a single party's data. They fall short when multiple independent data owners need to jointly compute over their private inputs while keeping them hidden.
- **Trusted Execution Environments (TEEs)**, also known as secure enclaves, offer an attractive hardware-based approach, but history has shown that they are not invulnerable. New attacks are discovered with troubling regularity, raising concerns about their long-term security.

Chainlink Confidential Compute. Our new technology called *Chainlink Confidential Compute* takes a different approach by combining the strengths of existing paradigms while mitigating their weaknesses. We deliberately rely on TEEs for their performance advantages, but design the system so that the damage from any TEE compromise is tightly contained.

The key idea is to integrate TEEs with *threshold encryption*, ensuring that no single machine, TEE or other, ever holds a master secret that, when compromised, would jeopardize confidentiality of the entire system. Access to private information is granted strictly on a *need-to-know basis*, following the principle of least privilege. That is, each component only receives the minimum permissions and data required to perform its role, and nothing more.

At the same time, the architecture leverages cloud-hosted TEEs provided by reputable cloud providers, benefiting from the providers' strong physical and organizational security measures. Moreover, their multi-billion cloud business itself serves as an external stake to the protocol: large providers have reputations to protect, and the risk of reputational damage creates an additional deterrent against mismanagement or abuse.

Decentralization is still preserved through cryptographic safeguards and distribution of trust. In particular, we do not rely on a single technology stack or cloud provider; instead, we envisage a heterogeneous pool of TEEs spanning different hardware vendors, cloud providers, and independent operators. This pool is orchestrated and assigned by a decentralized network of node operators, ensuring that no single organization or technology constitutes a single point of failure.

This design aims to make confidential computing practical, scalable, and compatible with the decentralized ethos that underpins blockchain systems.

2 System Architecture

2.1 System Overview

The architecture of Chainlink Confidential Compute combines decentralized orchestration with cloud-hosted trusted execution environments (TEEs), layered with cryptographic safeguards. The design avoids dependence on a single vendor or technology, relying instead on a heterogeneous pool of enclaves distributed across cloud providers and operators.

Entities. The main entities in the system are:

- **users**, who feed private inputs into the Chainlink Confidential Compute service by encrypting their inputs under the system's master public key and submitting the resulting ciphertexts to applications.
- **applications**, which may be centralized services or decentralized smart contracts, assemble encrypted user inputs together with a description of the code to be executed. This code specifies the algorithm that should be run on the private inputs contained in the ciphertexts, forming a computation request.
- **oracle nodes**, organized in a decentralized oracle network (DON), that jointly check the authenticity of requests, assigns requests to enclaves, checks compliance, and attests to responses.
- **decryption nodes**, also organized in a DON, that collectively hold the decryption master secret in threshold form, and that produce encrypted key shares for enclaves.
- **compute enclaves**, TEEs that receive encrypted inputs and key shares, execute computations, and return attested results.

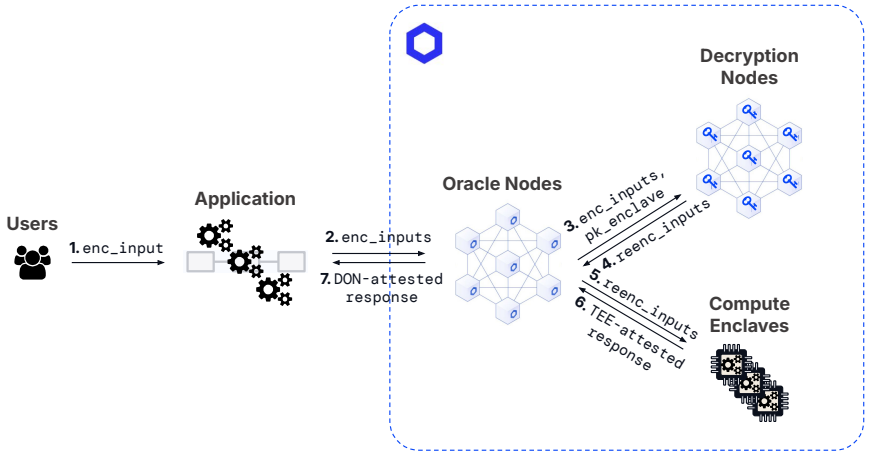

Figure 1: High-level architecture and workflow in Chainlink Confidential Compute. Steps 0–7 in the text illustrate how encrypted user inputs are processed by the oracle nodes, decryption nodes, and compute enclaves to yield a verified result.

Flow diagram. The overall workflow proceeds as depicted in Figure 1:

0. The decryption nodes jointly generates a public key and secret key shares for a threshold public-key encryption scheme. In such a scheme, the decryption key is split among many nodes so that no single node can decrypt alone; only a threshold subset of nodes acting together can recover plaintexts. In parallel, each compute enclave in the pool generates its own public-private key pair, used to receive re-encrypted inputs from the Confidential Data network.

1. Users encrypt their private inputs under the threshold public key and submit the resulting ciphertexts to an application.

2. Applications (either centralized services or decentralized smart contracts running on a blockchain) assemble encrypted inputs together with the code to be executed on these private input into computation requests and submit these to the Chainlink Confidential Compute service.

3. The oracle nodes verify the authenticity of each request (e.g., observing a smart contract on a blockchain), assigns it to one or more compute enclaves in the pool, and submits the quorum-signed encrypted inputs and the assigned enclave public key to the decryption nodes.

4. The decryption nodes apply their secret shares to the encrypted inputs to re-encrypt the encrypted inputs to the public key of the assigned enclave.

5. The oracle nodes forwards the quorum-signed re-encrypted inputs to the designated compute enclave(s).

6. The compute enclave(s) decrypt the private inputs, execute the specified computation, and produce an attestation over the result. This attestation proves not only that the computation was performed correctly, but also that it was executed inside genuine enclave hardware running in the designated cloud environment.

7. The oracle nodes verify the enclaves' attestations, quorum-sign the result, and return the signed result to the application.

2.2 Integration in the Chainlink Runtime Environment

The Chainlink Runtime Environment (CRE) provides a flexible framework for composing decentralized services. At its core, CRE supports *triggers* that initiate workflows, *workflows* that define sequences of actions and capabilities, *capabilities* that encapsulate services such as data feeds, randomness, or decryption, and *targets* that are the endpoints that consume workflow results. This abstraction allows developers to define modular and reusable decentralized processes.

Chainlink Confidential Compute in CRE. Within CRE, the main components of the Chainlink Confidential Compute architecture correspond to CRE roles as follows:

- The oracle nodes are instantiated as a *Workflow DON*, which coordinates requests, enforces compliance, and orchestrates capabilities.

- The decryption nodes will appear as a threshold decryption capability, also referred to as the *Vault DON* or *Confidential Data capability*.

- The pool of compute enclaves is exposed as a *Confidential Compute capability*.

A typical confidential workflow in CRE proceeds in the following steps:

1. A *trigger* activates the workflow, specifying encrypted inputs and a reference to the code that should be executed on them.

2. The Workflow DON is activated, verifying the request and assigning an enclave from the Confidential Compute capability.

3. The Workflow DON invokes the Confidential Data capability, which applies its threshold shares of the decryption key and produces encrypted key shares addressed to the assigned enclave.

4. The Workflow DON passes the computation request and encrypted key shares to the designated enclave in the Confidential Compute capability.

5. The enclave decrypts inputs, executes the code, and produces an attested response.

6. The Workflow DON verifies the enclave's attestation, signs the result, and forwards it to the specified *target*.

Through CRE integration, Chainlink Confidential Compute becomes a composable service inside the Chainlink ecosystem. The CRE workflow model provides the glue between triggers, confidential computation, threshold decryption, and result delivery, ensuring that the system's confidentiality and integrity guarantees carry over seamlessly into broader decentralized applications.

3 Configurable Security

The design of Chainlink Confidential Compute supports multiple security profiles. Depending on the requirements of a given application, users can select the configuration that offers the right balance of confidentiality, integrity, performance, and privacy.

3.1 Single Execution

At the most basic security level, each query is assigned to and executed by a single compute enclave in the pool.

Confidentiality. Confidentiality of the private inputs is ensured by multiple mechanisms:

- **Proactive threshold security.** The master decryption key is never held by any single node; instead, it is secret-shared among the decryption nodes. Security is guaranteed as long as fewer than a threshold of decryption nodes is corrupted *within the same epoch*. At every epoch, the master secret is proactively re-shared among the nodes, preventing adversaries from accumulating key shares over time.

- **Strict need-to-know principle and data minimization.** A compromised compute enclave only ever learns inputs for those requests assigned to it *after* it is compromised and *before* it is removed from the pool. This is achieved by using per-ciphertext threshold decryption shares and forward-secure encryption: enclaves generate a fresh public encryption key for each request, so that decryption shares can only be used in that request and do not expose past or future computations.

- **Binding ciphertexts with associated data.** Encryption labels (also known as associated data) can be used to bind ciphertexts to specific applications or decryption policies. This ensures that even if ciphertexts are replayed or redirected, they cannot be decrypted outside their intended context.

Integrity. Integrity of the result is achieved by the TEE's hardware attestation, verified by the oracle nodes, which then re-sign the response with a quorum signature. Execution integrity is therefore achieved as long as the assigned TEE is uncompromised and fewer than a threshold of oracle nodes are corrupt.

Availability. The system is designed to avoid single points of failure. A decentralized threshold network of oracle nodes and decryption nodes ensures that the service can continue despite individual failures or corruptions. A diverse pool of compute enclaves, sourced from multiple hardware vendors, cloud providers, and operators, provides redundancy. If a computation fails on one enclave, the oracle nodes can reassign it to another enclave in the pool.

Compliance. The strong privacy offered by Chainlink Confidential Compute must be balanced with safeguards against abuse. Compliance measures can be embedded directly into the application code, allowing developers to enforce relevant checks (e.g., KYC or data-use policies), including on private data.

3.2 Replicated Execution

The next security level improves execution integrity, at the cost of increasing privacy risk. It assigns each query to multiple enclaves, who execute it in parallel; oracle nodes compare the results before reporting back on chain.

This approach requires computations to be deterministic, which can be a problem for inherently randomized computations, e.g., encryption of outputs. Chainlink Confidential Compute therefore employs threshold-generated deterministic randomness provided by the key machines, to ensure that results are comparable.

Replicated execution provides stronger integrity guarantees, since inconsistencies across enclaves can be detected, but weaker privacy, since compromise of a single assigned enclave suffices to leak inputs.

3.3 Zero-knowledge Attestation

At this security level, each query is assigned to a single enclave, which executes it and additionally produces a zero-knowledge (ZK) proof of correct execution. Integrity of the result is now guaranteed mathematically, rather than based on trust in the secure hardware, while confidentiality is based on the hardware security of a *single* enclave. It comes at the cost of a considerable loss in performance compared to native execution, however, since generating zero-knowledge proofs is highly resource-intensive.

3.4 MPC/FHE Execution

For maximal decentralization of trust, each query is assigned to multiple enclaves, which perform the computation jointly using secure multiparty computation (MPC) or fully-homomorphic encryption (FHE).

In the case of MPC, each enclave only learns a secret share of every input, which requires an adapted threshold encryption scheme that distributes private inputs into shares across enclaves. In the case of FHE, all enclaves perform the operations on encrypted inputs using the homomorphic properties, and then jointly decrypt the public outputs.

Privacy is preserved as long as less than a threshold of the participating enclaves are compromised; integrity of the result is achieved under the same assumption. The performance penalty is typically even higher than for zero-knowledge attestation, though.

4 Use Cases

Chainlink Confidential Compute provides generic confidential computation as a decentralized service. Its flexibility allows developers to incorporate confidential logic into a wide range of applications without having to develop dedicated cryptographic primitives. We highlight several illustrative use cases below.

4.1 Confidential Connectivity

A common requirement in decentralized workflows is to call external APIs over HTTPS. In conventional designs, developers must embed their API credentials directly into the workflow, which exposes those secrets to all participating nodes in the Workflow DON.

With Chainlink Confidential Compute, this problem can be solved elegantly by following the Town Crier[1] design, enabling secure and verifiable HTTPS queries. API credentials are encrypted under the system's master public key and stored safely within the workflow. At execution time, the oracle nodes retrieve the encrypted credentials and forward them to a compute enclave. Inside the enclave, the credentials are decrypted, used to call out and authenticate to the external API, and immediately discarded. The API call therefore executes with both confidentiality and integrity guarantees, while preserving verifiability through attestation.

This mechanism naturally generalizes to more complex interactions, where additional encrypted secrets can be provided as private inputs, or where only selected parts of the API response are revealed in the output. For example, a user of the smart contract could supply an encrypted credit card number to enable a payment provider to process transactions securely, without exposing the card details to the blockchain, to the developer, or to any node outside the enclave.

[1] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. *ACM CCS 2016*.

4.2 Proprietary Data Feeds

Some data providers, particularly in the financial sector, sell timely and accurate data to customers for a fee, e.g., proprietary market indices or portfolio weights. They are obviously unwilling to make such data openly available on a public blockchain, where it could be freely copied and redistributed. Onchain protocols and DeFi applications may want to purchase and consume this data directly within blockchain workflows, enabling the next wave of decentralized finance built on private, high-quality information.

To satisfy contractual and regulatory requirements, the data providers would like to have an irrefutable record of when data was made available, and which customer accessed it at which time.

With blockchains and Chainlink Confidential Compute, these goals become compatible. Data providers can post the encrypted data feeds directly to an onchain smart contract, which serves as a privacy-preserving distribution hub. The smart contract maintains a list of approved subscribers, and when a subscriber requests access, it invokes the Chainlink Confidential Compute service to re-encrypt the data to that subscriber's public key, or to post it securely to an API endpoint of the subscriber's choice. Every interaction is logged immutably onchain, providing transparency and auditability without disclosing the data itself. The Chainlink Confidential Compute service could even trigger automated onchain actions such as executing token trades or adjusting collateral positions based on the decrypted data, enabling fully autonomous, privacy-preserving financial workflows.

This approach allows data providers to monetize confidential information safely, while giving decentralized applications verifiable access to trusted offchain data streams. In doing so, Chainlink Confidential Compute bridges the gap between traditional data markets and the emerging ecosystem of onchain finance, unlocking a new class of privacy-preserving data-driven applications.

4.3 Private Tokens

Another compelling use case is the creation of privacy-preserving digital assets. Chainlink Confidential Compute simplifies the design of private tokens in both balance confidentiality and transaction anonymity.

Hiding balances and transaction amounts. In the simplest model, an application maintains a balance table, but each balance entry is encrypted under the master public key. A transaction contains an encrypted amount. The smart contract verifies the sender's (plaintext) signature, looks up the encrypted sender and receiver balances, and submits a request to Chainlink Confidential Compute. Inside the enclave, the balances and transaction amount are decrypted, updated, and returned in encrypted form.

Compared to UTXO-based privacy designs, this account-based model is more efficient and also provides a simpler mental model for users, since they only need to keep track of balances rather than managing sets of individual coins or notes.

Hiding sender and receiver addresses. Privacy can be further strengthened by encrypting the sender and receiver addresses themselves. In this variant, the smart contract maintains an encrypted balance table, while transactions contain encrypted sender and receiver addresses along with an encrypted signature. A batch of such transactions is submitted to Chainlink Confidential Compute, where the enclave decrypts the balance table and the transactions, validates the transactions, applies the updates to the balance table, and re-encrypts the updated balance table. The enclave may report only a hash of the updated balance table onchain, while storing the full encrypted state on a separate data availability layer.

This design achieves stronger privacy than existing approaches based on zero-knowledge proofs or fully homomorphic encryption, while outperforming them in efficiency. Prototype benchmarks demonstrate significantly lower overhead in both computation and storage.

4.4 Identity

Identity verification is a critical building block for moving traditional finance onchain, where regulatory compliance and user privacy must coexist.

Simple credential checks. At its simplest, Chainlink Confidential Compute can verify digital credentials inside an enclave, releasing only a coarse-grained attribute onchain (e.g., whether a user is resident of a non-embargoed country). This allows smart contracts to enforce compliance without exposing sensitive personal data.

Chainlink Confidential Compute identity system. A more advanced pattern is to use Chainlink Confidential Compute as the basis for a full-fledged identity system by letting the enclave act as a "credential re-certifier". Here, a long-term re-certification key is generated inside the enclave and kept in its encrypted state. The enclave verifies any Web2 credentials as described above, checks that it satisfies the requested claim, and issues a one-time certificate for the user's blockchain address attesting to the claim.

The certificate can be used and verified directly onchain, without having to invoke the Chainlink Confidential Compute service. This approach enables efficient verification while maintaining user privacy and unlinkability. Optionally, encrypted credentials or detailed personally identifiable information (PII) can be stored encrypted onchain, on a Chainlink Confidential Compute data availability layer, or at a trusted escrow agent, so that regulators could de-anonymize suspicious transactions if needed.

Additional features such as claim revocation and Sybil-resistant pseudonyms can be added to the identity system in a fairly straightforward fashion.

Acknowledgements

We would like to thank Dan Boneh, Christian Cachin, and Ari Juels, and Michael Reiter for their review of earlier versions of this work, fruitful discussions about the architecture, and for reviewing security proofs of its underlying cryptographic protocols.