BLOCKSTAND

Deliverable D2 – A reference architecture demonstrating integration of computable smart contracts with existing blockchain and distributed ledger technologies

Soumya Kanti Datta, Digiotouch

Funded by
the European Union

Introduction

A reference architecture for computable smart contracts is a cornerstone for software developers and aims to serve as a blueprint, providing a standardised approach to integrating complex computations with blockchain and distributed ledger technology (DLT). Such an architecture guides developers, product owners, and all relevant stakeholders, ensuring that smart contract solutions are robust, reliable, and aligned with industry best practices and regulatory requirements. The potential benefits are significant, as it overcomes several technical, economic, legal, compliance, and adoption challenges. It is essential for several reasons:

- **Accelerated development and adoption**: It paves the way for (a) standardisation of a common framework for developer, reducing time-to-develop, (b) incorporating well-established design patterns, security measures, and (c) compatibility between different blockchain platforms and off-chain components. As a result, it encourages a modular development approach for developer, test, maintain, and CI/CD of smart contracts. Such smart contract development architectural framework promotes the use of reusable components and services, speeding up the development process and reducing costs.
- **Mitigating security risks**: It considers well-known vulnerabilities providing resilient guidelines for security risk mitigation. It leads to providing an assurance that the architecture and system components are built on a secure foundation and demonstrates a commitment to open, secure standards. Trust and transparency establishment becomes easier.
- **Interoperability**: It will facilitate the development of smart contracts that can operate across different blockchain and DLT platforms, reducing redundancy and increasing efficiency while integrating them with existing enterprise systems, promoting smoother and more effective interoperability.
- **Compliance and governance**: It is easier to ensure that smart contracts comply with relevant European (and/or international) regulatory requirements that are crucial for their deployment in finance, healthcare, and other market sectors.
- It allows focusing on **core business logic development** by providing a standardised foundation where business and software developers can collaborate to create their own unique value propositions. Therefore, a reference architecture paves the way for testing out new ideas and services which further encourages the development of novel tools, products, and services that exploit smart contracts.

In addition to the technical benefits, smart contract reference architecture will lead to many business benefits, especially on the front of improved decision making. Having a reference architecture will create a shared and common language and understanding of how smart contracts interact with blockchain and DLTs at the decision making level of the companies. In terms of cost-benefit analysis when launching a new smart contract product or service, it will enable evaluation of different deployment options based on specific criteria.

Proposed reference architecture

The proposed reference architecture requires the following components from an end user as well as software developer perspectives.

- End users are going to interact (initiate and monitor) with the computable smart contract(s) through an easy to use user interface (UI).

- A well-known entry point for all client requests is needed which authenticates and routes the incoming requests to the protected smart contract services. An API gateway is an ideal choice here.
- A software layer where smart contracts are developed and deployed using existing software frameworks. They should support version control and management.
- Interfaces to blockchain platforms are needed where the computable smart contracts are integrated and executed.
- Oracles should be present to provide smart contracts with external data and providing secure interfaces to obtain real-world events and information.
- A service or a middleware layer is required to handle communication between blockchain and off-chain components. For the smart contracts, event listeners are needed to monitor blockchain events.
- Authentication, encryption, auditing, access logging, confidentiality mechanisms are paramount to the architecture to ensure secure and private interactions from the end users with the computable smart contracts.
- In addition to that, a compliance and governance module is required that oversees that all smart contracts comply with relevant regulatory requirements and provides means for resolving disputes.

Based on such common requirements, the reference architecture proposed by this project include four primary components. They are summarised below.

- **Blockchain network**: This is the underlying distributed ledger technology platform with which the computable smart contract must be integrated and deployed to. The network (e.g., Ethereum, Hyperledger Fabric) thus hosts the smart contract and manages its execution environment.
- **Off-chain computation environment:** This component handles several aspects including any (heavy) computational workload, UI for end users, API gateway, middleware, security, and confidentiality aspects. This environment can be analogous to a cloud based platform, a distributed computing network, or specialised hardware accelerators.
- **Oracle network**: This acts as a bridge between the on-chain and off-chain worlds. It feeds data into the smart contract and relays computation results back to the chain.
- **Data Storage:** This component stores the data required for computations. It can be a decentralized storage solution or a centralized database.

Key considerations behind developing the reference architecture are summarised below.

- Ensuring the security of data, computations, and smart contract execution is paramount. This includes implementing data encryption, access control, and secure communication channels to protect sensitive information and maintain system integrity.
- The off-chain computation environment should be capable of handling increasing computational loads as the system grows. Utilizing load balancing and distributed computing techniques can help achieve this, ensuring that the system remains responsive and efficient under higher demands.
- For the oracle network to be effective, it must be trusted to provide accurate and reliable data. It's essential to have mechanisms in place for verifying the data and computation results, ensuring that the information used in smart contracts is dependable.

- Balancing the cost of off-chain computations with their benefits is crucial. The architecture should be designed to optimize costs without compromising performance, ensuring that the system remains economically viable.
- To maximize flexibility and future-proofing, the architecture should be compatible with various blockchain platforms and off-chain computation environments. This ensures that the system can adapt to different technologies and integrate seamlessly with other systems.

Another set of key considerations are necessary to create a generic wrapper for smart contract integration and deployment across different blockchains, due to its inherent complex nature. The key challenges here are to overcome differences in:

- **Programming languages:** Solidity for Ethereum, Rust for Solana, etc.
- **Deployment processes:** Different tools and commands.
- **Transaction structures:** Variation in data formats and fees.
- **Network interactions:** Diverse RPC endpoints and communication protocols.

Key considerations for such a generic wrapper are the following.

- **Abstraction Layer:**
    - Define a common interface or abstract class representing a smart contract.
    - Implement specific implementations for different blockchains.
    - Provide methods for deployment, interaction, and query.
- **Configuration Management:**
    - Use configuration files to store blockchain-specific parameters (network URLs, contract addresses, etc.).
    - Allow for easy switching between different blockchain environments.
- **Dependency Management:**
    - Utilize dependency management tools to handle blockchain SDKs and other required libraries.
- **Error Handling:**
    - Implement robust error handling mechanisms to gracefully handle exceptions and provide informative error messages.
- **Testing:**
    - Write comprehensive unit and integration tests to ensure wrapper correctness.
    - Consider using test networks for cost-effective testing.

Proposed reference architecture

The proposed reference architecture is depicted below. It comprises of several key components, which are detailed below.

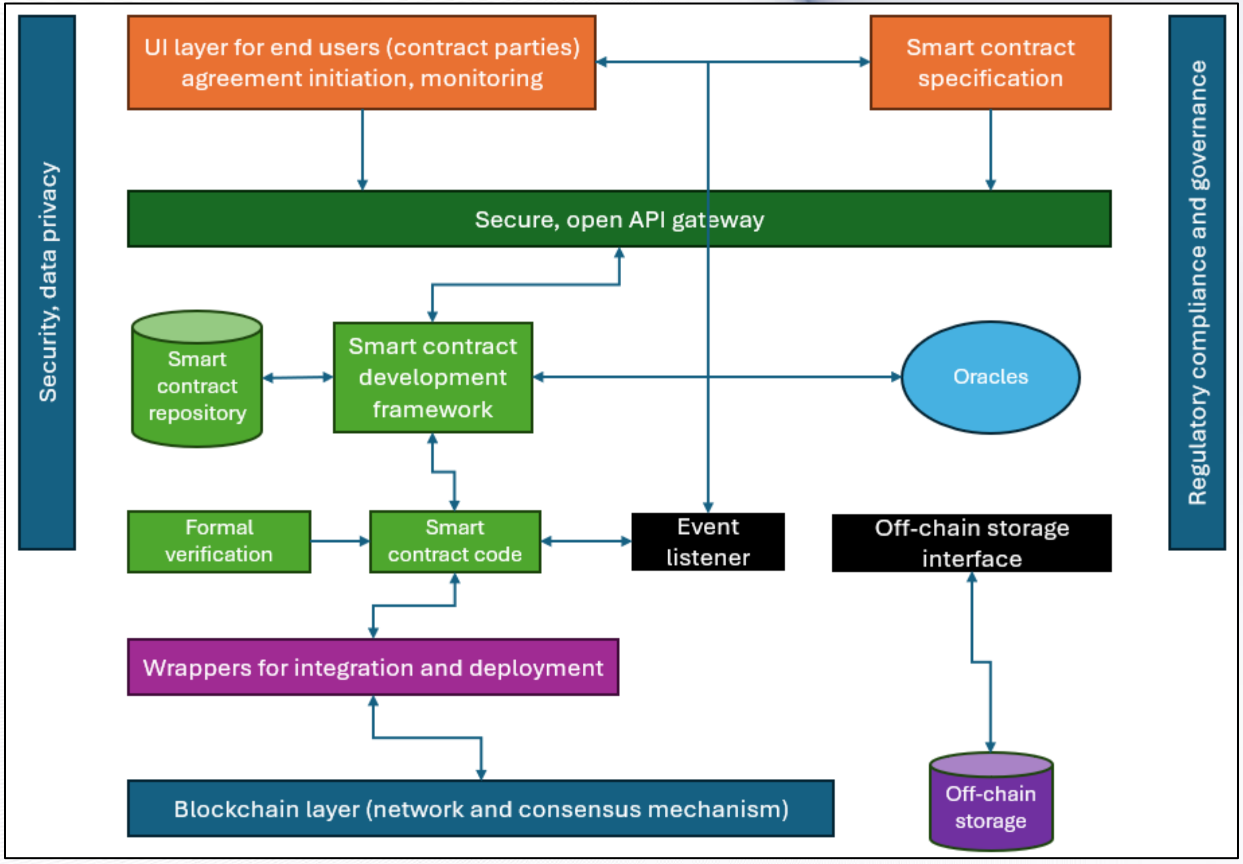
Security, data privacy
UI layer for end users (contract parties) agreement initiation, monitoring
Smart contract specification
Secure, open API gateway
Smart contract repository
Smart contract development framework
Oracles
Formal verification
Smart contract code
Event listener
Off-chain storage interface
Wrappers for integration and deployment
Blockchain layer (network and consensus mechanism)
Off-chain storage
Regulatory compliance and governance

Figure 1: Proposed reference architecture.

User Interface (UI) layer: The UI layer serves as the human-computer interface, enabling users to interact with the smart contract system. It encompasses a web or a mobile app that facilitates end user actions such as initiating computable smart contract agreement, secure transactions, monitoring contract states, and engaging with the underlying blockchain network.

API gateway: It acts as a reverse proxy and a control point. The API Gateway manages inbound and outbound API traffic from a software deployment perspective. Key functions include handling authentication, authorisation, rate limiting, and routing requests to appropriate backend services. This layer enhances system performance, security, and maintainability.

Smart contract layer: It houses the core logic of the system, comprising self-executing contracts defined in code. This layer is composed of following components -

- **Smart contract development framework**: It is essentially the software framework where the computation aspects of the contract are handled. It provides tools and libraries for creating, testing, and deploying smart contracts (e.g., Truffle, Hardhat).
- **Repository**: It is an administrative component which manages versioning and states of smart contracts deployed in the underlying blockchain network(s).
- **Smart contract code**: This module houses the self-executing pieces of code that automate and enforce the terms of contracts (agreed in the UI layer between the contracting parties) without the need for intermediaries.

- **Formal verification**: This module is in charge of evaluating the correctness of the smart contract and its code with respect to a formal specification created smart contract specification module.

Blockchain Layer: the foundation of the overall system, the blockchain layer underpins the execution of smart contracts and recording of transactions. Three essential components of this layer include:

- **Blockchain network**: The distributed ledger (e.g., Ethereum, Hyperledger Fabric) responsible for maintaining a shared, immutable record of transactions.
- **Consensus mechanism**: The algorithm ensuring agreement among network participants on the ledger's state (e.g., Proof of Work, Proof of Stake).
- **Oracles**: External data providers that bridge the gap between the blockchain and the real world by feeding off-chain information into smart contracts.

Middleware layer: This intermediary layer facilitates communication and integration between the smart contract layer and the blockchain layer. It consists of:

- **Event listeners**: Monitors blockchain events and triggers corresponding actions within the system. Event listeners facilitate automated workflows across blockchain platforms and off-chain systems by monitoring events and initiating corresponding actions, such as triggering a smart contract when specific conditions are met (e.g., initiating payment when goods are delivered).
- **Off-chain storage interface**: Open interface for web services like storing data external to the blockchain (e.g., in a cloud storage) for optimizing performance and cost.

Security services: It is a set of services covering technical components through authentication, authorization, encryption, and audit/logging mechanisms and is applicable to the entire reference architecture components.

Compliance and governance: Ensuring adherence to legal and regulatory frameworks, this layer includes:

- **Compliance verification**: Automated checks to validate smart contract alignment with regulations.
- **Dispute resolution**: Mechanisms for managing conflicts arising from smart contract execution.

Wrappers for integration and deployment: while this is the most complex component to develop, from an architectural approach, it consists of three main sub-components. They are elaborated below.

- **Wrapper class:**
  - It encapsulates common smart contract functionalities (e.g., deployment, calling functions, querying state).
  - It also provides abstract methods for blockchain-specific implementations.
- **Blockchain specific implementation:**

  - In the second step, it will inherit from the wrapper class and provide concrete implementations for integration and deployment within each blockchain.
  - In addition to that, it handle blockchain-specific micro technical details, such as, transaction building, signing, and sending.
- **Software configuration:**
  - It is recommended to use a configuration file to store blockchain-specific parameters and enable dynamic configuration updates.

The proposed reference architecture in the report is designed with versatility and interoperability in mind. It incorporates several key features that ensure seamless interaction, communication, and data exchange across different platforms, protocols, and systems which are summarised below.

- **Modular design**
  - Modular components (e.g., the user interface, smart contract layer, middleware, and blockchain network) allow different software modules of the architecture can be developed, deployed, and updated independently. This ensures that specific modules, such as the off-chain computation environment or the oracle network, can be replaced or updated to support new technologies, new blockchain platforms, or external systems without disrupting the rest of the architecture. This modularity also enables easy integration of new blockchain platforms (e.g., Ethereum, Hyperledger, or Solana) or external data sources as the ecosystem evolves.
- **Oracle network**
  - It serves as a crucial intermediary for enabling smart contracts to access off-chain data. This allows the architecture to integrate real-world data from various external sources ensuring that the smart contracts can operate on different blockchain platforms and receive data from trusted external systems while maintaining the accuracy and security of this data. Oracles ensure that blockchain networks, which are generally isolated from external data, can interact and communicate with off-chain components like web services or cloud-based platforms.
- **Blockchain agnostic smart contract layer**
  - The architecture is designed to be blockchain agnostic, meaning it can support multiple blockchain platforms like Ethereum, Hyperledger, or Solana. The generic wrapper for smart contract integration and deployment across different blockchains abstracts the heterogeneity in:
    - Programming languages (e.g., Solidity, Rust)
    - Transaction structures (e.g., data formats, fees)
    - Network interactions (e.g., RPC endpoints, communication protocols)
  - This abstraction ensures that smart contracts can be easily ported and run on different existing blockchain networks, making the architecture versatile and interoperable across various DLTs.
- **Interoperability via the generic wrapper**

  - A generic wrapper for integration and deployment across different blockchains allows for easy connection and interaction with multiple blockchain systems. This component simplifies the development and deployment of smart contracts across different platforms by providing a unified interface for managing blockchain-specific details. The wrapper allow for the abstraction of low-level blockchain-specific functions, making it possible for multiple blockchain platforms to interact seamlessly and exchange data.

- **Event listener for blockchain event monitoring**
  - The architecture dedicates a software building block on event listeners that monitor blockchain events and trigger appropriate responses. This functionality ensures that real-time blockchain data (e.g., transactions, state changes) can be synchronised with off-chain systems.

Alignment of the individual components with current landscape of standards

A study has been undertaken determining the alignment of the relevant architecture building blocks with current landscape of standards. It shows which individual components are built on which standards and where more work from standard development organisations are required. This section thus helps in avoiding fragmentation of efforts as well as ensure that blockchain ecosystem stakeholders actually adopt the proposed reference architecture.

- **Blockchain network**
  - The architecture supports different existing blockchain platforms like Ethereum and Hyperledger Fabric, which are widely used and based on well-established protocols and standards. For example, Ethereum follows the ERC-20[1] standard for tokens, ERC-721[2] for NFTs, and Hyperledger Fabric follows the ISO/TC 307[3] standard.
- **Smart contract layer**
  - The architecture supports frameworks like Truffle or Hardhat for developing smart contracts as well as programming languages like Solidity (for Ethereum) and Rust (for Solana) which adhere to widely accepted development practices within the blockchain community. However, more formalised standards are required for cross-chain smart contract deployment and interoperability across different blockchain environments which is a broader goal being proposed by this project.
- **API Gateway**
  - This component exploits widely adopted industry standards like OAuth 2.0, OpenID Connect, and RESTful APIs ensuring that the rest of the components can interact among them and external systems in a standardised and secure manner.
- **Oracle network**
  - Standardisation of oracle services is still a work in progress. Greater collaboration between projects like Chainlink[4] and international standard development

[1] https://ethereum.org/en/developers/docs/standards/tokens/erc-20/
[2] https://ethereum.org/en/developers/docs/standards/tokens/erc-721/
[3] https://www.iso.org/committee/6266604.html
[4] https://chain.link/

organisations is needed to formalise how oracles should function across different blockchain platforms.

Regulatory and legislative context for the reference architecture

The reference architecture must conform to the European and national regulatory frameworks, satisfying compliance and governance aspects as well. This will further ensure that the smart contracts developed and deployed through the generic wrapper will also conform to the European legal frameworks. Here are the key regulations and legislations that are directly relevant to the reference architecture.

- **EU GDPR**
  - The architecture must ensure compliance with GDPR[5], especially in lawfully handling of personal data processed within smart contracts or stored off-chain.
- **Markets in Crypto-Assets (MiCA) Regulation**[6]
  - It is essential for ensuring that smart contract development and deployment through the architecture, especially in the financial sector, meet necessary regulatory compliance. It focuses on governance and operational resilience of distributed ledger technologies. MiCA ensures that the computable smart contracts must adhere to transparency, auditability, and dispute resolution procedures.
- **Anti-Money Laundering Directive**
  - The fifth and sixth Anti-Money Laundering Directives[7] (AMLD5 and AMLD6) aim to bring more transparency to cryptocurrency and digital transactions to prevent money laundering and any unlawful financing. The reference architecture takes it into account under the regulatory framework building block which will be in charge of reporting any suspicious activities to relevant financial authorities.
- **Digital Operational Resilience Act (DORA)**
  - It addresses the digital resilience of financial systems using blockchain or distributed ledger technologies[8]. It mandates that financial institutions and crypto-asset service providers must have robust IT systems to withstand operational risks. In response to this regulation, the software building blocks of the architecture must be installed on such an infrastructure that is operationally resilient in blockchain deployments.
- **Payment Services Directive 2 (PSD2)**
  - Given that smart contracts are often related to financial transaction related use cases, this[9] is a highly relevant legislation.
- **Data Governance Act (DGA)**
  - For smart contracts involving large amount of data processing and sharing (e.g., between For organizations or across borders), the architecture must comply with the DGA[10].

Potential future activities

European Blockchain Services Infrastructure[11] (EBSI), established by the European Commission (EC), sets the groundwork for cross-border blockchain services, emphasising standardisation

[5] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679
[6] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020PC0593
[7] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32018L0843
[8] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52020PC0595
[9] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32015L2366
[10] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2020:767:FIN
[11] https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/What+is+ebsi

across blockchain platforms and interoperability. While the reference architecture aligns with EBSI's goal of enabling smart contracts to operate across different blockchain environments (such as verifiable credentials[12], track and trace), an in-depth alignment of the proposed architecture with should be conducted, especially for powering European cross-border use cases involving such computable smart contracts.

Another potential future activities involve the eIDAS Regulation (Regulation 910/2014)[13]. This regulation supports cross-border recognition of electronic identification (eID) and trust services. The main concerned component of the proposed architecture is the API gateway which must comply with this regulation to ensure that electronic transactions are legally binding across borders.

The above two aspects are beyond the scope of the current project and hence has been presented here as potential future activities.

Conclusion

In a nutshell, a well-defined reference architecture provides SMEs and other ecosystem players with a solid foundation for building innovative and scalable applications based on computable smart contracts, enabling them to compete effectively in the digital economy. The proposed reference architecture demonstrating integration of computable smart contracts with blockchain and DLTs brings several benefits for the SMEs and the European blockchain ecosystem.

Benefits for SMEs

- **Reduced development costs:** SMEs can leverage the reference architecture to expedite development and reduce costs associated with smart contract integration and deployment on existing blockchain and DLT infrastructure.
- **Offering advanced capabilities:** By integrating computable smart contracts, SMEs can offer powerful computational tools to their customers which were previously not possible.
- **Enhanced trust and transparency:** Blockchain technology, combined with computable smart contracts, aids in building and maintaining trust with customers and partners.
- **Competitive advantage:** By adopting such reference architecture early, SMEs can gain a competitive edge in their industry.

Benefits for the European blockchain ecosystem

- **Interoperability:** A common architecture fosters compatibility between different blockchain platforms and off-chain computation environments, promoting a more interconnected ecosystem.
- **Accelerated development:** By providing a pre-defined structure, developers can focus on application-specific logic rather than reinventing the wheel for each smart contract project.
- **Risk mitigation:** A well-tested reference architecture can help identify potential vulnerabilities and security risks, reducing the likelihood of costly errors.
- **Standardization:** A shared architecture promotes consistency in development practices and facilitates the creation of industry-wide standards.

[12] https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/EBSI+Verifiable+Credentials
[13] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014R0910

In addition to the mentioned advantages, this project also recommends software developer upskilling, further enlargement of the European blockchain ecosystem, and rapidly building trust and acceptance on smart contract for it to become mainstream application.

To ensure that the **reference architecture for computable smart contracts remains relevant, secure, and trustworthy as the European blockchain ecosystem rapidly evolves**, it must be proactively adapted to regulations & compliance, future-proofing strategies, and blockchain advances. While the architecture has been proposed with modular and flexible software components, continuous monitoring of regulatory landscape changes/updates and governance & self-adaption are necessary.

DISCLAIMER