United States Patent Application Publication	20250258956
Kind Code	A1
Publication Date	August 14, 2025
Inventor(s)	Crabtree; Jason et al.

FEDERATED DISTRIBUTED COMPUTATIONAL GRAPH ARCHITECTURE FOR BIOLOGICAL SYSTEM ENGINEERING AND ANALYSIS

Abstract

A federated distributed computational system enables secure collaboration across institutions for unified biological and multiomics data analysis. It comprises interconnected computational nodes managed by a central federation manager. Each node includes specialized components: a local computational engine for biological data processing, a privacy-preservation system, a knowledge integration component leveraging dynamic knowledge graphs, and a secure communication interface. The federation manager coordinates computational activities while ensuring security, privacy, legality, and contractual adherence. This architecture allows institutions, citizen scientists, and patients to collaborate on complex biological analyses without compromising sensitive data. By enabling shared computational resources and expertise, the system facilitates breakthrough discoveries while maintaining confidentiality. Additionally, it supports pro-rata or contractually defined participation in resultant benefits or knowledge, ensuring equitable collaboration.

Inventors: **Crabtree; Jason (Vienna, VA), Kelley; Richard (Woodbridge, VA), Hopper; Jason (Halifax, CA), Park; David (Fairfax, VA)**

Applicant: **QOMPLX LLC** (Reston, VA)

Family ID: **1000008492111**

Appl. No.: **19/060600**

Filed: **February 21, 2025**

Related U.S. Application Data

parent US continuation-in-part 19009889 20250103 PENDING child US 19060600
parent US continuation-in-part 19008636 20250103 PENDING child US 19009889
parent US continuation-in-part 18656612 20240507 PENDING child US 19008636
parent US continuation-in-part 18952932 20241119 PENDING child US 19060600

parent US continuation-in-part 18900608 20240927 PENDING child US 18952932
parent US continuation-in-part 18801361 20240812 PENDING child US 18900608
parent US continuation-in-part 18662988 20240513 PENDING child US 18801361
parent US continuation-in-part 18656612 20240507 PENDING child US 18952932
parent US continuation-in-part 18662988 20240513 PENDING child US 18656612
us-provisional-application US 63551328 20240208

Publication Classification

Int. Cl.: **G06F21/62** (20130101); **G06F21/53** (20130101)

U.S. Cl.:

CPC **G06F21/6245** (20130101); **G06F21/53** (20130101);

Background/Summary

CROSS-REFERENCE TO RELATED APPLICATIONS [0001] Priority is claimed in the application data sheet to the following patents or patent applications, each of which is expressly incorporated herein by reference in its entirety: [0002] Ser. No. 19/009,889 [0003] Ser. No. 19/008,636 [0004] Ser. No. 18/656,612 [0005] 63/551,328 [0006] Ser. No. 18/529,932 [0007] Ser. No. 18/900,608 [0008] Ser. No. 18/801,361 [0009] Ser. No. 18/662,988

BACKGROUND OF THE INVENTION
Field of the Art
[0010] The present invention relates to the field of distributed computational systems, and more specifically to federated architectures that enable secure cross-institutional collaboration while maintaining data privacy.
Discussion of the State of the Art
[0011] Recent advances in AI-driven gene editing tools, including CRISPR-GPT and OpenCRISPR-1, have demonstrated the potential of artificial intelligence in designing novel CRISPR editors. However, these systems typically operate in isolation, limited by centralized architectures and predetermined operational parameters. Current solutions lack the ability to effectively coordinate large-scale genomic interventions across multiple institutions while maintaining data privacy and enabling real-time optimization of computational workflows.
[0012] The limitations of current approaches extend beyond just architectural constraints. Traditional distributed computing solutions have struggled to handle the unique challenges posed by handling the unique complexities of biological data analysis, particularly when dealing with sensitive genomic information that must be kept private while still enabling meaningful cross-institutional collaboration. Existing systems often require centralizing data in ways that create security vulnerabilities or impose rigid operational frameworks that limit the flexibility and scope of analyses that can be performed.
[0013] Furthermore, current solutions lack the ability to dynamically adapt to changing computational demands and varying privacy requirements across different institutions. While some systems attempt to address privacy through encryption or data anonymization, these approaches often compromise the ability to perform complex, real-time analyses across distributed datasets. This is particularly problematic in biological research, where insights often emerge from examining patterns across diverse, multi-scale, and heterogeneous data sources.
[0014] Additionally, existing platforms struggle to effectively coordinate large-scale computational

tasks across institutional boundaries while maintaining local autonomy and security protocols. The challenge of balancing institutional independence with collaborative capability has led to fragmented and siloed solutions that fail to realize the full potential of distributed biological research.

[0015] What is needed is a flexible, federated architecture that can maintain data privacy while enabling secure cross-institutional collaboration, dynamically adapt to varying computational demands and privacy requirements, and support real-time optimization of distributed biological analyses across multiple scales and timeframes.

SUMMARY OF THE INVENTION

[0016] Accordingly, the inventor has conceived and reduced to practice a system and method for secure cross-institutional collaboration in distributed computational environments. The core system comprises a plurality of computational nodes coordinated by a federation manager, where each node contains specialized components for processing biological data while maintaining privacy controls. The federation manager coordinates distributed computation across the plurality of nodes, maintains a dynamic resource inventory, implements secure information exchange protocols, and facilitates cross-institutional collaboration while preserving data privacy. Through this comprehensive coordination approach, the system enables secure, scalable, and efficient collaboration across institutional boundaries while maintaining the confidentiality and sovereignty of sensitive data.

[0017] According to a preferred embodiment, each computational node incorporates a local computational engine that processes biological data, a privacy preservation subsystem that protects sensitive information, a knowledge integration component that manages biological data relationships and knowledge graph database on epidemiology, biology, and chemistry, and a communication interface that enables secure information exchange between nodes. The federation manager dynamically coordinates all computational activities across this network while ensuring data privacy is maintained throughout all processes.

[0018] According to another embodiment, the federated system enables flexible distribution of fundamental domain knowledge (such as epidemiology and biology) and specialized models across the network, without requiring a centralized repository. These knowledge graphs and domain-specific models can be dynamically shared across subgraphs of the federated graph, either by moving the models to execute in proximity to local datasets, or by transmitting data to the models for processing with results returned across the graph. This approach supports “knowledge in flight,” where domain expertise can be dynamically and flexibly relocated based on computational and data locality requirements.

[0019] According to another preferred embodiment, the system implements a multi-temporal modeling framework that analyzes biological data across multiple time scales, from rapid molecular interactions to long-term organismal changes, while enabling dynamic feedback integration. This framework allows for real-time adaptation of computational strategies and resource allocation based on ongoing analysis results, while maintaining security protocols across institutional boundaries.

[0020] According to an aspect of an embodiment, the system incorporates genome-scale editing capabilities through a specialized subsystem that coordinates multi-locus editing operations with real-time validation. This subsystem enables complex genomic modifications while maintaining the security and privacy requirements essential for sensitive biological data.

[0021] According to another aspect of an embodiment, the system utilizes synthetic data generation to facilitate cross-domain knowledge transfer through adaptive optimization. This capability enables institutions to collaborate effectively while protecting proprietary information and maintaining compliance with data privacy specifications and regulations.

[0022] According to yet another aspect of an embodiment, the system implements blind execution protocols that enable collaborative computation while maintaining strict data privacy between

nodes. These protocols ensure that institutions can participate in joint research efforts without compromising sensitive information or violating security policies. This includes the ability to execute code, algorithms in full or part, machine learning models, or other software code on any computational node within the federated graph where resources are available. This execution acts as a serverless code execution feature within the federated graph.

[0023] According to methodological aspects of the invention, the system implements methods for establishing and operating the federated distributed computational system that mirror the above-described system capabilities. These methods encompass all operational aspects including automated node configuration, multi-layer privacy preservation, dynamic knowledge integration, AI-driven synthetic data generation, multi-temporal modeling, and adaptive genome-scale editing, all while maintaining secure cross-institutional collaboration.

Description

BRIEF DESCRIPTION OF THE DRAWING FIGURES

[0024] FIG. **1** is a block diagram illustrating an exemplary architecture of federated distributed computational graph (FDCG) for biological system engineering and analysis.

[0025] FIG. **2** is a block diagram illustrating an exemplary architecture of multi-scale integration framework.

[0026] FIG. **3** is a block diagram illustrating an exemplary architecture of federation manager subsystem.

[0027] FIG. **4** is a block diagram illustrating an exemplary architecture of knowledge integration subsystem.

[0028] FIG. **5** is a block diagram illustrating an exemplary architecture of genome-scale editing protocol subsystem.

[0029] FIG. **6** is a block diagram illustrating an exemplary architecture of multi-temporal analysis framework subsystem.

[0030] FIG. **7** is a method diagram illustrating the initial node federation process of which an embodiment described herein may be implemented.

[0031] FIG. **8** is a method diagram illustrating distributed computation workflow of which an embodiment described herein may be implemented.

[0032] FIG. **9** is a method diagram illustrating the knowledge integration process of which an embodiment described herein may be implemented.

[0033] FIG. **10** is a method diagram illustrating multi-temporal analysis of which an embodiment described herein may be implemented.

[0034] FIG. **11** is a method diagram illustrating genome-scale editing process of which an embodiment described herein may be implemented.

[0035] FIG. **12** illustrates an exemplary computing environment on which an embodiment described herein may be implemented.

DETAILED DESCRIPTION OF THE INVENTION

[0036] The inventor has conceived and reduced to practice a federated distributed computational system that enables secure cross-institutional collaboration for biological data analysis and engineering. The system implements a novel architectural framework that transcends traditional centralized approaches through a distributed network of computational nodes coordinated by a federation manager. The core architecture comprises multiple interconnected computational nodes, each containing specialized components for processing biological data, such as local computational engine, a privacy-preserving subsystem employing differential privacy and homomorphic encryption, and a dynamic knowledge integration component for cross-domain data linking, all while maintaining strict privacy controls. These nodes operate within a federated distributed

computational graph architecture specifically designed for genome-scale operations and multi-temporal biological system modeling. The federation manager coordinates all distributed computation across the network, employing adaptive scheduling algorithms, real-time feedback loops, and secure multi-party computation while ensuring data privacy is maintained throughout all processes.

[0037] Each computational node incorporates a local computational engine for processing biological data, a privacy preservation system that protects sensitive information, a knowledge integration component that manages biological data relationships, and a secure communication interface. Through this comprehensive coordination approach, the system enables efficient collaboration across institutional boundaries while maintaining the confidentiality of sensitive data through advanced blind execution protocols.

[0038] The system implements both multi-scale integration capabilities for coordinating analysis across molecular, cellular, tissue, and organism levels, as well as multi-temporal modeling frameworks that enable simultaneous analysis across different time scales. These capabilities are enhanced through distributed machine learning components throughout the architecture, enabling sophisticated pattern recognition, simulation optimization, and predictive modeling while maintaining data privacy.

[0039] This architectural framework provides a flexible foundation that can be adapted for various epidemiological analysis, biological analysis and engineering applications while maintaining consistent security and privacy guarantees across all implementations. The system's modular design allows for the incorporation of additional specialized components as needed for specific use cases, while the core architecture ensures secure and efficient cross-institutional collaboration.

[0040] The invention implements a federated distributed computational graph architecture specifically designed for biological system analysis and engineering. This architectural approach enables secure collaborative computation across institutional boundaries while maintaining strict data privacy controls. The system's graph-based architecture allows complex biological computations to be distributed across multiple nodes while preserving security through selective information sharing and blind execution protocols, providing a secure and scalable framework for distributed biological and multiomics computations.

[0041] The federated distributed computational graph architecture represents biological computations as interconnected processing nodes within a dynamic graph structure. Each node in this graph represents a complete computational system capable of autonomous operation, while edges between nodes represent secure channels for data exchange and collaborative processing. Computational tasks are decomposed into discrete operations that can be distributed across multiple nodes, with the federation manager dynamically adjusting the graph topology and orchestrating task execution while preserving institutional boundaries. This federation enables institutions to maintain control over their sensitive biological data and proprietary methods while participating in collaborative research through secure graph edges managed by standardized protocols. The graph-based approach is particularly well-suited for biological system engineering and analysis due to the inherently interconnected nature of biological processes across multiple scales. Just as biological systems operate through complex networks of molecular interactions, cellular pathways, and tissue-level communications, the computational graph architecture enables parallel processing of these multi-scale relationships while maintaining the security requirements essential for sensitive genetic and molecular data. This architectural alignment between biological systems and computational representation enables sophisticated analysis of complex biological relationships while preserving the privacy controls necessary for cross-institutional collaboration in genomic and epidemiologic research and engineering.

[0042] In the context of biological system engineering, the federated distributed computational graph serves multiple critical functions. It enables partitioning of complex genomic analyses across participating nodes, coordinates multi-temporal modeling across different time scales, and

facilitates secure knowledge sharing between institutions. The architecture supports both centralized and decentralized implementation patterns, providing flexibility to adapt to different institutional requirements and security needs.

[0043] When implemented in a decentralized pattern, computational nodes handling biological data operate as peer entities, coordinating through secure gossip protocols that maintain data privacy while enabling resource discovery and workload distribution. Each node advertises only its computational capabilities and available resources, and public datasets that the node owner has explicitly designated for sharing, never exposing sensitive biological data or proprietary analytical methods. This pattern is particularly valuable for collaborative genome engineering projects where institutions need to maintain strict control over their genetic data and engineering protocols.

[0044] In centralized implementations, a primary coordination node maintains a high-level view of the federation's resources and processes while preserving the autonomy of individual nodes. This approach enables efficient distribution of large-scale genomic analyses and engineering tasks across the federation while ensuring that sensitive biological data remains protected within each participating institution's security boundary.

[0045] The federation manager component plays a crucial role in orchestrating biological computations across the distributed graph architecture. It maintains a dynamic inventory of computational resources, decomposes complex biological analyses into discrete tasks, and matches these tasks with appropriate nodes based on their capabilities and security requirements. The manager facilitates secure information exchange between components while enforcing strict data protection policies across the federation.

[0046] This architectural framework supports blind and partially blind execution patterns, where computational tasks involving sensitive biological data are encoded into graphs that can be partitioned and selectively obscured through multi-party computation protocols. This enables institutions to collaborate on complex biological analyses without exposing proprietary data or methods. The system implements dynamic task allocation based on real-time conditions, allowing for adaptive resource distribution as computational requirements evolve during complex biological analyses.

[0047] The architecture provides particular value for biological research and engineering scenarios that involve sensitive genetic data, proprietary engineering methods, or regulatory compliance requirements. It enables secure cross-institutional collaboration while maintaining the strict data privacy controls necessary for biological research and development.

[0048] In accordance with a preferred embodiment, the system implements a multi-scale integration framework that coordinates biological analysis across molecular, cellular, tissue, and organism levels. The molecular processing engine handles the integration of protein, RNA, and metabolite data, while the cellular system coordinator manages cell-level data and pathway analysis. These components work in concert with the tissue integration layer and organism scale manager to maintain consistency across biological scales through the cross-scale synchronization system.

[0049] The molecular processing engine employs machine learning models to identify patterns and predict interactions between different molecular components. These models are trained on standardized datasets using federated learning approaches while maintaining privacy through federated learning approaches. The cellular system coordinator implements graph-based algorithms to analyze pathway relationships and cellular networks, enabling complex multi-scale analyses while preserving data security.

[0050] The federation manager maintains system-wide coordination through several integrated components. The resource tracking system continuously monitors node availability, computational capacity, and capabilities, enabling efficient task distribution across the federation. The blind execution coordinator implements secure computation protocols that allow collaborative analysis while maintaining strict data privacy. This coordinator employs advanced cryptographic techniques

to enable computations on sensitive data without exposing the underlying information.
[0051] A key aspect of the federation manager is its distributed task scheduler, which manages cross-institutional workflows through sophisticated orchestration algorithms. The security protocol engine enforces privacy policies and access controls across all nodes, while the node communication system handles secure inter-node messaging and synchronization. These components work together to enable complex collaborative analyses while maintaining institutional data boundaries.
[0052] The knowledge integration system implements a comprehensive approach to biological data management. Its vector database provides efficient storage and retrieval of biological data, while the knowledge graph engine maintains complex relationship networks across multiple scales. The temporal versioning system tracks data history and changes, working in concert with the provenance tracking system to ensure complete data lineage. The ontology management system maintains standardized biological terminology and relationships, enabling consistent interpretation across institutions.
[0053] For genome-scale editing operations, the system may implement specialized components for coordinating complex genetic modifications. The CRISPR design coordinator manages edit design across multiple loci, ensuring precise targeting and alignment with experimental objectives, while the validation engine performs real-time verification of editing outcomes. The off-target analysis system employs machine learning models to predict and monitor unintended effects with high accuracy and minimize collateral damage, working alongside the repair pathway predictor to model DNA repair outcomes. These components are integrated through the edit orchestration system, which coordinates parallel editing operations while maintaining security protocols.
[0054] The multi-temporal analysis framework enables sophisticated temporal modeling through several integrated components. The temporal scale manager coordinates analysis across different time domains, while the feedback integration system enables dynamic model updating based on real-time results. The rhythm analysis component processes biological rhythms and cycles, working with the scale translation engine to convert between different temporal scales. These components are supported by the prediction system, which employs machine learning models to forecast system behavior across multiple time scales.
[0055] In accordance with various embodiments, the system implements specific protocols and mechanisms to enable secure distributed computation across biological scales. The communication interface at each node employs standardized APIs that abstract the underlying implementation details while maintaining consistent security protocols. These interfaces support both synchronous and asynchronous communication patterns, enabling flexible workflow coordination across the federation.
[0056] The blind execution protocols are implemented through a multi-layer encryption scheme that enables computational nodes to process sensitive biological data without accessing the underlying information. When a node initiates a computation request, the federation manager's security protocol engine generates encrypted computation graphs that partition the analysis into discrete steps. Each participating node receives only the information necessary to perform its assigned computations, with results aggregated through secure multi-party computation protocols to maintain data privacy and integrity.
[0057] The system's vector database implementation utilizes specialized indexing structures optimized for biological data types. These structures enable efficient querying of high-dimensional biological data while maintaining strict access controls. The database supports both exact and approximate nearest neighbor searches, enabling similarity-based queries across biological datasets while preserving data privacy through differential privacy mechanisms.
[0058] The knowledge graph engine implements a distributed graph database architecture that maintains consistency through a consensus protocol. Biological relationships are encoded using standardized ontologies, with the ontology management system maintaining mappings between

institutional terminology and standard references. The temporal versioning system implements a multi-version concurrency control mechanism that enables concurrent access while maintaining data consistency.

[0059] For genome-scale editing operations, the system implements a specialized pipeline architecture that coordinates edit design and validation across multiple nodes. The CRISPR design coordinator employs machine learning models to optimize edit strategies, identifying precise loci and minimizing off-target effects, while the validation engine implements real-time monitoring protocols that track editing progress and outcomes. These components interact through a message-passing interface that maintains security boundaries while enabling complex coordination patterns.

[0060] The multi-temporal analysis framework implements a hierarchical time management system that coordinates analyses across different temporal scales. Time series data is processed through specialized stream processing engines that maintain temporal consistency while enabling real-time analysis. The prediction system employs ensemble learning approaches that combine multiple machine learning models to generate robust forecasts while maintaining privacy through federated learning protocols.

[0061] Resource allocation across the federation may be managed through a distributed scheduling system that optimizes task distribution based on node capabilities and current workload. The scheduler implements a priority-based queuing mechanism that ensures critical tasks receive appropriate resources while maintaining overall system efficiency. This scheduling system works in concert with the resource tracking system to maintain optimal resource utilization across the federation.

[0062] In accordance with various embodiments, the system implements specific protocols and mechanisms to enable secure distributed computation across biological scales. The communication interface at each node employs standardized APIs that abstract the underlying implementation details while maintaining consistent security protocols. These interfaces support both synchronous and asynchronous communication patterns, enabling flexible workflow coordination across the federation.

[0063] The blind execution protocols are implemented through a multi-layer encryption scheme that enables computational nodes to process sensitive biological data without accessing the underlying information. When a node initiates a computation request, the federation manager's security protocol engine generates encrypted computation graphs that partition the analysis into discrete steps. Each participating node receives only the information necessary to perform its assigned computations, with results aggregated through secure multi-party computation protocols.

[0064] The system's vector database implementation utilizes specialized indexing structures optimized for biological data types. These structures enable efficient querying of high-dimensional biological data while maintaining strict access controls. The database supports both exact and approximate nearest neighbor searches, enabling similarity-based queries across biological datasets while preserving data privacy through differential privacy mechanisms.

[0065] The knowledge graph engine implements a distributed graph database architecture that maintains consistency through a consensus protocol. Biological relationships are encoded using standardized ontologies, with the ontology management system maintaining mappings between institutional terminology and standard references. The temporal versioning system implements a multi-version concurrency control mechanism that enables concurrent access while maintaining data consistency.

[0066] For genome-scale editing operations, the system implements a specialized pipeline architecture that coordinates edit design and validation across multiple nodes. The CRISPR design coordinator employs machine learning models to optimize edit strategies, while the validation engine implements real-time monitoring protocols that track editing progress and outcomes. These components interact through a message-passing interface that maintains security boundaries while enabling complex coordination patterns across the federation.

[0067] The multi-temporal analysis framework implements a hierarchical time management system that coordinates analyses across different temporal scales. Time series data is processed through specialized stream processing engines that maintain temporal consistency while enabling real-time analysis. The prediction system employs ensemble learning approaches that combine multiple machine learning models to generate robust forecasts while maintaining privacy through federated learning protocols.

[0068] Resource allocation across the federation is managed through a distributed scheduling system that optimizes task distribution based on node capabilities and current workload. The scheduler implements a priority-based queuing mechanism that ensures critical tasks receive appropriate resources while maintaining overall system efficiency. This scheduling system works in concert with the resource tracking system to maintain optimal resource utilization across the federation.

[0069] In accordance with various embodiments, the system may implement multiple layers of security and privacy protection mechanisms designed to safeguard sensitive biological data while enabling secure cross-institutional collaboration.

[0070] The privacy preservation system may incorporate advanced encryption protocols that can protect data both at rest and in transit. These protocols could include homomorphic encryption techniques that may enable computations on encrypted data without decryption, potentially allowing institutions to collaborate on sensitive analyses while maintaining data privacy. The system may also implement secure multi-party computation protocols that could enable multiple parties to jointly compute functions over their inputs while keeping those inputs private.

[0071] Access control mechanisms may be implemented through a flexible framework that could support various authentication and authorization schemes. The system may utilize role-based access control that could be enhanced with attribute-based policies, potentially enabling fine-grained control over data access and computational operations. These mechanisms may be augmented with context-aware security policies that could dynamically adapt to changing operational conditions.

[0072] The blind execution protocols may be implemented through multiple possible approaches. One potential implementation could involve secure enclaves that establish trusted execution environments for sensitive computations. Another approach might utilize zero-knowledge proofs that could enable nodes to verify computation results without accessing the underlying data. The system architecture may support integration of various privacy-preserving computation techniques as they emerge. In one aspect multi-party computation can be achieved through a combination of using Shamir's secret sharing algorithm to break the data into shares, using secure computation protocols such as garbled circuits or homomorphic encryption for computation. Privacy aware graph algorithms may be used when appropriate. For example, intermediate node visits in breath first search traversals may remain private.

[0073] Audit mechanisms may be implemented to maintain comprehensive trails of system operations while preserving privacy. These mechanisms could employ privacy-preserving logging techniques that may record essential operational data without exposing sensitive information. The system may support configurable audit policies that could be tailored to specific institutional requirements and regulatory frameworks.

[0074] The federation manager may implement security orchestration protocols that could coordinate privacy-preserving operations across the distributed system. These protocols might include secure key management systems that could enable dynamic key rotation and distribution while maintaining operational continuity and protecting sensitive operations. The system may also support integration with existing institutional security infrastructure through standardized interfaces.

[0075] In accordance with various embodiments, the system architecture may accommodate multiple implementation variations to support diverse institutional requirements and biological

research needs. The core architecture's flexibility enables adaptation across different operational contexts while maintaining fundamental security and collaboration capabilities.
[0076] The federation manager may be implemented through various architectural patterns that align with specific institutional requirements. In some embodiments, the manager might operate as a distributed service across multiple nodes, potentially enabling enhanced reliability and load distribution. Alternative implementations could utilize a hierarchical approach where multiple federation managers might coordinate across different organizational boundaries, potentially enabling scalable management of large research networks.
[0077] The computational nodes may implement varying internal architectures based on available resources and specific research requirements. Some nodes might utilize specialized hardware accelerators for specific biological computations, while others could operate on standard computing infrastructure. The system architecture may accommodate this heterogeneity through abstraction layers that could standardize node interactions regardless of underlying implementation details.
[0078] Knowledge integration components may be adapted to support different data storage and processing paradigms. Some implementations might utilize distributed database systems optimized for biological data types, while others could integrate with existing institutional data repositories. The architecture may support multiple approaches to data organization and retrieval while maintaining consistent security protocols across variations.
[0079] The privacy preservation system may incorporate different protection mechanisms based on specific security requirements and regulatory frameworks. Some implementations might emphasize homomorphic encryption for sensitive genomic data, while others could prioritize secure multi-party computation for collaborative analyses. The system architecture may support integration of various privacy-preserving technologies as they emerge and evolve.
[0080] Workflow orchestration may be implemented through different coordination patterns depending on specific research requirements. Some embodiments might employ event-driven architectures for real-time analysis, while others could utilize batch processing approaches for large-scale genomic studies. The system may support multiple execution patterns while maintaining consistent security and privacy guarantees across implementations.
[0081] These implementation variations demonstrate the architecture's adaptability while preserving its fundamental capabilities for secure cross-institutional collaboration in biological research and engineering.
[0082] In accordance with various embodiments, the system architecture may support integration with diverse existing biological research infrastructure and systems while maintaining security and privacy guarantees across integrated components.
[0083] The federated system may implement standardized integration interfaces that could enable secure communication with established research databases and analysis platforms. These interfaces might support multiple data exchange protocols and formats commonly used in biological research, potentially allowing institutions to leverage existing data resources while maintaining privacy controls. The architecture may accommodate both synchronous and asynchronous integration patterns based on specific operational requirements.
[0084] Integration with existing authentication and authorization systems may be achieved through flexible security frameworks that could support various identity management protocols. The system architecture may enable institutions to maintain their established security infrastructure while implementing additional privacy-preserving mechanisms for cross-institutional collaboration. This approach could potentially allow seamless integration with existing institutional security policies and compliance frameworks.
[0085] The knowledge integration components may support connectivity with various types of biological databases and analysis platforms. This could include integration with genomic databases, protein structure repositories, pathway databases, and other specialized biological data sources. The system architecture may enable secure access to these resources while maintaining privacy controls

over sensitive research data.
[0086] Computational workflows may be designed to integrate with existing analysis pipelines and tools commonly used in biological research. The system may support multiple approaches to workflow integration, potentially enabling institutions to maintain their established research methodologies while gaining the benefits of secure cross-institutional collaboration. This integration capability could extend to various types of analysis software, visualization tools, and computational platforms.
[0087] Data transformation and exchange mechanisms may be implemented to enable secure integration with legacy systems and databases. These mechanisms could support multiple data formats and exchange protocols while maintaining privacy controls over sensitive information. The system architecture may accommodate various approaches to data integration while ensuring consistent security guarantees across integrated components.
[0088] In accordance with various embodiments, the system architecture may incorporate various scaling capabilities to accommodate growth from small research collaborations to large multi-institutional deployments while maintaining security and performance characteristics.
[0089] The federation manager may implement adaptive scaling mechanisms that could enable dynamic adjustment of system resources based on operational requirements. These mechanisms might support both horizontal scaling through the addition of computational nodes and vertical scaling through enhancement of existing node capabilities. The system architecture may accommodate various approaches to resource scaling while maintaining consistent security protocols and privacy guarantees across the federation.
[0090] Computational workload distribution may be implemented through flexible scheduling frameworks that could optimize resource utilization across different scales of operation. The system may support multiple approaches to workload balancing, potentially enabling efficient operation across deployments ranging from small research groups to large institutional networks. These frameworks might adapt to changing computational requirements while maintaining privacy controls over sensitive research data. These workloads may also be both distributed across the computational graph as allowed by resource and data requirements, as well as individual workloads be dynamically moved and allocated to new resources as needed based on graph demand.
[0091] The knowledge integration components may incorporate scalable data management approaches that could efficiently handle growing volumes of biological data. These approaches might include various strategies for distributed data storage and retrieval, potentially enabling the system to scale with increasing data requirements while maintaining performance characteristics. The system architecture may support multiple approaches to data scaling while preserving security guarantees across different operational scales.
[0092] Network communication capabilities may be implemented through scalable protocols that could efficiently handle increasing numbers of participating nodes. These protocols might support various approaches to managing network traffic and maintaining communication efficiency across different scales of deployment. The system may accommodate multiple strategies for scaling network operations while maintaining secure communication channels between participating institutions.
[0093] Security and privacy mechanisms may be designed to scale efficiently with growing system deployment. These mechanisms might implement various approaches to managing security policies and privacy controls across expanding institutional networks. The system architecture may support multiple strategies for scaling security operations while maintaining consistent protection of sensitive research data across all operational scales.
[0094] In accordance with various embodiments, the system architecture may incorporate error handling and recovery mechanisms designed to maintain operational reliability while preserving security and privacy requirements across the federation.
[0095] The federation manager may implement fault detection protocols that could identify various

types of system failures or inconsistencies. These protocols might utilize different approaches to monitoring system health and detecting potential issues across the distributed architecture. The system may support multiple strategies for fault detection while maintaining privacy controls over sensitive operational data.

[0096] Recovery mechanisms may be implemented through flexible frameworks that could respond to different types of system failures. The system architecture might support various approaches to maintaining operational continuity during node failures, network interruptions, or other system disruptions. These mechanisms may include different strategies for maintaining data consistency and workflow progress while preserving security guarantees during recovery operations. Some specific examples of how to maintain data consistency and workflow progress while preserving security guarantees include the following approaches: For transactional systems, implementations should utilize atomic transactions across related operations, implement two-phase commit protocols for distributed systems, maintain transaction logs for rollback capabilities, version all data changes within transactions, and use optimistic or pessimistic locking as appropriate. State management requires storing workflow state in durable storage (particularly in systems like DynamoDB, RDS, or Postgres), using checkpointing to track progress reliably, implementing idempotency keys for operations, maintaining audit logs of state transitions, and employing state machines for complex workflows. Recovery patterns should incorporate retry mechanisms with exponential backoff, utilize Dead Letter Queues (DLQ) for failed operations, create compensating transactions for rollbacks, implement saga patterns for distributed workflows, and store recovery points in secure, encrypted storage. Security considerations must be maintained throughout, including encryption during recovery operations, secure token rotation during long-running processes, least-privilege access for recovery operations, comprehensive audit logging of recovery actions, and ensuring sensitive data remains encrypted both at rest and in transit. Workflow integrity is maintained through unique correlation IDs across distributed systems, event sourcing for reliable history, well-defined consistency boundaries in distributed systems, distributed locks for critical sections, and circuit breakers for failing components. Data consistency is achieved through strong consistency where required, implementation of ACID properties for critical operations, use of CRDTs for distributed data structures, maintenance of materialized views for complex queries, and implementation of version vectors for conflict resolution. Finally, monitoring and validation encompasses implementing health checks for system components, using data validation at each step, monitoring workflow progress and timing, tracking resource usage during recovery, and implementing automated testing of recovery procedures.

[0097] The Dynamically Partitioned Federated Enclave Framework represents an enhancement to the existing privacy preservation subsystem, introducing granular enclaving capabilities that can be established within or across computational nodes at runtime. This embodiment's core innovation centers on the seamless instantiation of secure enclaves that segregate data handling for specific workflows, responding to emergent sensitivity levels or policy-driven requirements.

[0098] These enclaves function as ephemeral, distinct logical spaces, existing only for the duration of specific computational tasks—such as large-scale protein folding, multi-omic analysis, or genome-wide association studies—and automatically dissolving upon validated task completion. The framework transcends traditional static node-level compartmentalization by implementing on-demand enclaves that can be subdivided within a single node or span multiple nodes under managed constraints, thereby minimizing sensitive data exposure to any individual enclave participant.

[0099] The technical implementation relies on secure enclaves formed through lightweight virtualization layers, microVM hypervisors, or trusted execution modules (including Intel SGX, AMD SEV, or ARM TrustZone). Within this framework, the federation manager subsystem **300** manages dedicated cryptographic key pairs for each enclave instantiation, facilitating initial key exchanges through a secure handshake process overseen by the security protocol engine subsystem

340. Following authorization, the blind execution coordinator **320** handles computational task partitioning according to user-defined enclaving policies, ensuring cryptographic isolation of data from different research groups or institutions. This enclaving methodology encompasses memory access, storage buffers, and inter-process communication, creating effective isolation between enclaves and preventing unauthorized data crossover. The resource tracking subsystem **310** maintains oversight of enclave-capable node availability, manages key distribution lifecycles (including rotation for extended or shortened enclaves), and coordinates system-wide workload scheduling to prevent ephemeral enclaves from overwhelming the federation's computational capacity.

[0100] The established enclaves operate beneath a restricted interface layer exposed to the knowledge integration subsystem **400**, which receives only obfuscated or tokenized references from the enclaved data, such as hashed or partial identifiers for genomic sequence subsets, rather than unencrypted information. Privacy-preserving transformations mediate all queries to the knowledge graph engine or vector database, minimizing extraneous data exposure. The federation manager initiates a secure teardown procedure upon task completion, wherein ephemeral enclaves undergo a zero-knowledge finalization step that purges in-enclave ephemeral keys and deallocates associated resources, ensuring no residual data remains accessible to subsequent jobs. This embodiment's implementation of runtime enclaving enables dynamic enforcement of privacy boundaries in real time, allows security levels to be tailored to specific task requirements, and enhances the system's capability to manage multi-institutional collaborations where certain projects may require heightened data segregation even within individual nodes.

[0101] The system may implement state management protocols that could track and restore computational progress across distributed operations. These protocols might support various approaches to maintaining workflow state information while preserving privacy requirements. The architecture may accommodate different strategies for managing operational state across participating nodes while maintaining security boundaries during system recovery.

[0102] Data consistency mechanisms may be implemented to handle various types of synchronization failures across the federation. The system might support multiple approaches to maintaining data consistency during system disruptions while preserving privacy controls over sensitive research data. These mechanisms may include different strategies for detecting and resolving data conflicts while maintaining security guarantees across participating institutions.

[0103] The system architecture may support an implementation of audit mechanisms that could track error conditions and recovery operations while maintaining privacy requirements. These mechanisms might employ various approaches to logging system events and recovery actions without exposing sensitive information. The system may accommodate different strategies for maintaining audit trails while preserving security and privacy guarantees during error handling operations.

[0104] Communication recovery protocols may be implemented to handle various types of network failures or interruptions. These protocols might support different approaches to maintaining secure communication channels during system disruptions. The architecture may accommodate multiple strategies for restoring communication while preserving security guarantees across the federation.

[0105] In accordance with various embodiments, the system architecture may incorporate design elements that could enable adaptation to emerging technologies and methodologies in biological research and distributed computing while maintaining core security and collaboration capabilities.

[0106] The federation manager may be designed to accommodate future advances in distributed computing architectures and protocols. This extensibility might support integration of emerging computational paradigms, potentially including but not limited to new approaches to distributed processing, advanced privacy-preserving computation techniques, or novel methods for secure collaboration. The system architecture may support various approaches to incorporating new technological capabilities while maintaining backward compatibility with existing

implementations.
[0107] Knowledge integration components may be implemented through extensible frameworks that could adapt to evolving biological data types and analysis methodologies. These frameworks might support various approaches to incorporating new data structures, analytical methods, and research tools as they emerge in the field of biological research. The system architecture may accommodate different strategies for extending knowledge integration capabilities while maintaining security guarantees across new implementations.
[0108] Spatio-Temporal Knowledge Graph Integration for federated CRISPR experimentation and multi-omics workflows. In this embodiment, we introduce additional mechanisms that: incorporate spatial (tissue or location-based) constraints into CRISPR design and delivery decisions, and track temporal data over multiple timepoints or experiment rounds (e.g., multi-week CRISPR screens), updating knowledge graph (KG) subgraphs in real time. By integrating location- and time-specific knowledge in a distributed knowledge graph, the system can refine CRISPR design recommendations or pipeline logic over the entire life cycle of an experiment. For spatial or tissue-specific CRISPR designs, the knowledge graph data model for spatial context encompasses several key components. The distributed KG includes hierarchical ontologies describing tissues, cell lines, organoids, or in vivo models. Each cell line or tissue node is connected to metadata edges capturing typical constraints (e.g., “HeLa cells are known to favor Lentivirus transduction,” “Primary neuronal culture has high sensitivity to transfection reagents,” or “Cardiac muscle tissue has a high incidence of immune response to certain Cas9 proteins”). For local microenvironment and HPC logs, each tissue or cell line node links to local HPC usage logs or microenvironment parameters (oxygen tension, pH, growth factors). This architecture enables the system to represent that “CellLineA in Lab5 at Node48 HPC cluster is running 10 CRISPR tasks,” or “this lab's HPC pipeline for analyzing off-target is currently at 80% load.” It may also be appreciated that HPC to microservices migration or hybrid architectures are actively being explored by the scientific community and may be further enhanced by the system in an optional embodiment ranging from pure HPC, pure microservices/cloud or hybrid. For example, Log and administrative and performance data in an HPC environment. The microenvironment data (like drug concentrations, co-culture conditions) is stored as properties or linked sub-entities in the KG, enabling more precise CRISPR design constraints. Vector delivery constraints are represented through another edge or subgraph that indicates vector feasibility: e.g., “AAV-based vectors have low efficiency in TissueX” or “Electroporation is poorly tolerated in these fragile iPSCs.” By modeling these relationships, the knowledge graph becomes a “domain hub” for which CRISPR system or vector is recommended under certain spatiotemporal-biological conditions.
[0109] In one embodiment, the system integrates a spatio-temporal knowledge graph to model both spatial (e.g., tissue-, organ-, or lab-specific) constraints and temporal evolution of biological data. Each node in the knowledge graph represents a biological entity (e.g., gene, protein, tissue, cell line), augmented with location metadata (such as tissue type or physical lab site) and timestamped edges that track interactions, events, and changes over time. The system builds or refines subgraphs dynamically at each timepoint (e.g., daily, weekly) as new experimental data emerges, and merges them into comprehensive multi-timepoint “longitudinal” subgraphs upon completion of an experimental phase.
[0110] To support spatio-temporal data integrity, the knowledge graph engine implements ephemeral subgraphs for each timepoint. The ephemeral subgraphs capture intermediate states of biological systems—for instance, CRISPR editing efficiencies at different days in a multi-week experiment—and record potential off-target effects discovered only at later timepoints. A versioning component ensures that subgraphs are archived and “rolled up” to form historical snapshots of the entire experiment. The knowledge graph engine, in concert with the federation manager, manages these ephemeral subgraphs to respect user-defined privacy policies. For instance, each ephemeral subgraph can be subject to ephemeral enclave constraints, meaning it is

instantiated in a trusted execution environment and destroyed upon successful data integration or upon the user's request.

[0111] During multi-round screening or iterative CRISPR (or other gene or multi-omics related) editing campaigns, feedback loops enable the system to adapt new guide RNA designs or prime editing constructs for the next round based on patterns observed in prior timepoints. The scale translation subsystem (in the multi-temporal analysis framework) adjusts how local HPC clusters or microservices process 3D tissue organoid data from each ephemeral subgraph. This ensures that location-specific constraints (e.g., organoids in specialized microfluidic devices) and time-dependent readouts (e.g., weekly changes in gene expression) are fully captured before scheduling subsequent editing steps or predictive analyses.

[0112] Such a spatio-temporal knowledge graph approach addresses complex, real-time collaborative studies across institutions. Multiple labs can “subscribe” to relevant ephemeral subgraphs, viewing only anonymized or synthetic-data summaries from other participants while collectively refining cross-institutional insights. This dynamic enclaving of ephemeral subgraphs and location-aware metadata significantly advances secure data handling over conventional static or purely temporal knowledge graphs, enabling flexible yet privacy-preserving cross-site research.

[0113] The workflow for location-specific CRISPR design begins with the user request and LLM planner input phase. When a user (or automated pipeline) initiates a request like “I want to knock out gene ABC in TissueX,” the system triggers location-specific queries to the KG. During this process, the system identifies relevant nodes or edges capturing TissueX constraints, possible vector options, and historical HPC usage or success rates. The query execution phase then commences, where the system issues a parametric SPARQL (or similar) query to the knowledge graph. This query structure follows the pattern: “SELECT DISTINCT ?deliveryMethod WHERE {?deliveryMethod:hasDeliveryEfficacyFor:TissueX. ?deliveryMethod:hasOffTargetProfile?profile . . . }.” Through this query, the system obtains a ranked list of feasible CRISPR systems (Cas12a, Cas9 variants, prime editors) and recommended vector approaches (lentivirus, plasmid transfection, etc.), factoring known constraints from the KG. In the final LLM-driven decision or suggestion phase, the Task Executor or LLM Agent merges this KG-based data with the user's experimental goals (e.g., “High editing efficiency,” “Minimize immunogenic risk”). This culminates in a final design suggestion that references the relevant graph nodes, providing specific recommendations such as: “For TissueX in your institution's HPC constraints, we recommend prime editing with dCas9-based approach and a specialized liposome-based delivery due to lower local immune response.”

[0114] Additional technical components include a spatial reasoning engine which can handle advanced constraints such as 3D tissue geometry or organ subregions to further refine the recommended approach. This enables sophisticated decision-making, such as recognizing when a tissue is a 3D hepatic organoid and determining that direct plasmid transfection would be suboptimal, leading to routing to a microfluidic-based approach instead. Additionally, HPC integration is achieved through HPC logs incorporated into the KG, enabling the system to check node availability and capabilities, such as determining when “Node48 can run the off-target pipeline quickly with GPU acceleration.” The temporal summaries and multi-timepoint pipeline encompasses several key components. For ephemeral subgraphs at each timepoint, we acknowledge that many CRISPR experiments proceed over multiple days/weeks, collecting data or re-transducing at set intervals. We propose ephemeral subgraphs that “snapshot” each timepoint. The ephemeral subgraph creation process involves the system automatically spawning “Timepoint Subgraph” nodes at T=0, T=1wk, T=2wk, T=3wk, T=4wk for a single 4-week CRISPR screen. Each subgraph references updated metrics, including off-target accumulations, cell viability, guide RNA dropout or enrichment, and morphological changes. Data linking ensures each ephemeral subgraph is connected to prior timepoints for continuity through relationships such as “(Timepoint T=2wk)-[childOf].fwdarw.(Timepoint T=1wk).” Off-target predictions or newly discovered side

effects are represented as edges between gRNA nodes and newly discovered cleavage sites. The lifecycle management of these subgraphs allows for their merger into a final “longitudinal subgraph” or archival once the screen completes. This ephemeral approach ensures the KG remains dynamic, reflecting real-time data from HPC analyses or lab observations.

[0115] For multi-round CRISPR screens, adaptive rounds play a key role. In multi-round screens (e.g., gene knockout in 2-3 stages, or iterative selection steps), the system updates each ephemeral subgraph with new HPC analysis. This enables dynamic adaptation-if a certain gRNA is failing at T=1wk, the system might propose a new design by T=2wk. Automated off-target recalculation is implemented through the pipeline setting up scheduled tasks (via the Federation Manager) at each timepoint to recalculate off-target accumulations or coverage. These updates are written back to the ephemeral subgraph for that timepoint. The LLM Agent guidance component enables the LLM to see the newly updated subgraphs and run queries such as “Which guides had a 30% or greater on-target editing by T=1wk?” Based on these analyses, the agent can re-plan the next iteration, noting for example “We see guide #2 is suboptimal; let's propose an alternative guide in the next library.” The technical flow for multi-timepoint summaries begins with scheduled data harvest. At each timepoint (weekly, daily, or a user-defined schedule), the HPC pipeline ingests new readouts (NGS or qPCR data). A specialized “Temporal Data Manager” writes these results into ephemeral subgraph nodes. For KG and Vector DB integration, off-target embeddings or “signature embeddings” for each condition are stored in a vector DB, with the ephemeral subgraph referencing these embeddings. This structure enables semantic or k-NN queries across timepoints, such as “Find any timepoint that has a similar off-target distribution to T=2wk in a previous experiment.” The downstream tools component allows the multi-timepoint subgraphs to feed into the “Multi-Temporal Analysis” subsystem described in the overall architecture, enabling the LLM to produce new experiment instructions or collate final results for the user. Implementation notes regarding data structures specify that graph storage utilizes a distributed or cloud-based triple store or property graph (e.g., Neptune, JanusGraph, Blazegraph, or Neo4j) for the spatio-temporal knowledge graph. Temporal edge tagging ensures each relationship (like “hasOffTargetRate= . . . ”) includes a valid-from, valid-to timestamp or an event-based approach. For APIs and protocols, the Federation Manager organizes “graph update” events after each HPC pipeline completes, while LLM Agents rely on a “Graph Query Microservice” that surfaces relevant subgraph slices for the current experiment's timepoint and tissue.

[0116] Privacy considerations dictate that tissue or cell line data might be partially synthetic if the real environment is IP-protected or sensitive. Additionally, the ephemeral subgraphs can be ephemeral enclaves if data is only needed for short intervals before being anonymized. The user workflow begins with the user (or an automated script) setting up a multi-round screen. At T=0, CRISPR design is chosen with Tissue constraints. As timepoint ephemeral subgraphs appear, HPC processes the data, writes new off-target logs, and changes the subgraph edges. The LLM then re-checks or re-plans for T=1wk and subsequent timepoints. An example scenario of a multi-week, multi-round CRISPR screen in hepatic organoids illustrates this process: On Day 0, when a user indicates they want to disrupt a set of metabolic genes in a 3D hepatic organoid model, the knowledge graph references that these organoids respond poorly to plasmid transfection, leading the system to recommend an AAV vector with a prime editor. By Day 7, HPC logs update the ephemeral subgraph with the measured success rate of editing, and off-target analysis from the HPC pipeline shows new hotspots. The LLM agent, seeing the ephemeral subgraph, flags 2 guides as suboptimal. At Day 14, when the user triggers a second round, the ephemeral subgraph for T=14 merges prior data and re-plans with newly recommended guides. Finally, the system merges ephemeral subgraphs into a final “longitudinal record” that the knowledge graph can reference for future designs in hepatic organoids.

[0117] By adding Spatio-Temporal Knowledge Graph Integration, the system achieves several key capabilities. It manages location-specific CRISPR design constraints, recommended vectors, and

HPC usage conditions, while dynamically creating ephemeral subgraphs for each timepoint or iteration in multi-week CRISPR screens to track off-target and viability over time. The system also enables adaptive or iterative re-planning across multiple rounds, with real-time HPC logs feeding back into the knowledge graph. This embodiment significantly exceeds the typical single-run approach (e.g., CRISPR-GPT's “one experiment setup”). It supports multi-lab synergy, improved privacy, real-time adaptiveness, and deeper domain knowledge expressed in a graph format—a clear differentiator from simpler LLM-based design agents.

[0118] The privacy preservation system may be designed to incorporate future advances in security technologies and protocols beyond current differential privacy, emerging homomorphic encryption and current best practices. This extensibility might also support integration of emerging in-rest or in-transit or in-computation encryption methods, new approaches to secure computation (e.g., formal methods), or other advanced privacy-preserving techniques. The system architecture may support various approaches to enhancing privacy protection while maintaining compatibility with existing security, compliance and auditability implementations.

[0119] Computational workflows may be implemented through flexible frameworks that could adapt to new biological research methodologies and analysis techniques. These frameworks might support various approaches to incorporating emerging research tools and analytical methods. The system architecture may accommodate different strategies for extending computational capabilities while maintaining security and privacy guarantees across new implementations.

[0120] Integration capabilities may be designed to support future biological research infrastructure and platforms. This extensibility might enable secure integration with emerging research tools, databases, and analysis platforms while maintaining privacy controls. The system architecture may support various approaches to expanding integration capabilities while preserving security guarantees across new connections.

[0121] The federated CRISPR-GPT-style system can integrate with laboratory automation (e.g., Hamilton robots, Opentrons) and perform closed-loop, adaptive re-planning of CRISPR experiments. We highlight relevant robotics frameworks (such as Robot Operating System 2 (ROS2), action notation modeling language (ANML)), exemplary planning/search mechanisms (Monte Carlo tree search reinforcement learning, upper confidence bound for trees with super-exponential regret), and how these tie into knowledge graph updates, HPC instrumentation logs, and iterative human-machine teaming. The embodiment focusing on synergy with automated laboratory robotics and closed-loop lab execution expands upon the original CRISPR-GPT approach (which focuses heavily on planning and protocol design) to physically enact those protocols through lab automation hardware in a closed-loop manner. The system not only generates the experiment design but also issues instructions to laboratory robots and manages real-time data feedback. The high-level workflow begins with experiment plan generation, where the system (like CRISPR-GPT) determines a CRISPR editing protocol, specifying reagents, volumes, timings, and so on. The LLM Agent or orchestrator then translates these tasks into actionable scripts for robotics platforms. For action execution on lab robots, we have connected laboratory automation hardware —e.g., Hamilton pipetting robots, Opentrons liquid handlers, or specialized screening platforms. The system emits instructions (e.g., in JSON, CSV, or a domain-specific command format) to the robots, which handle pipetting, plating cells, reagent additions, or performing measurements like optical density or fluorescence. Online data capture occurs as the robots execute tasks, with sensors or integrated instruments producing intermediate readouts such as transduction efficiency from a fluorescent plate reader, cell viability from a real-time imaging station, and reagent usage logs. The system automatically ingests these data streams into the knowledge graph or ephemeral subgraphs for time-labeled storage (consistent with spatio-temporal integration from prior embodiments). Real-time monitoring is handled by the Federation Manager or the “ROS2/ANML layer” which tracks job statuses from each robotic device. If any anomalies occur (e.g., pipetting error, insufficient reagent volume), the system can pause or adjust the next steps accordingly. For

iterative or next-step re-planning, once the robotic step completes, results are posted back to the system's HPC pipelines for analysis, and the knowledge graph is updated. The system reevaluates the experiment design in a closed-loop manner—possibly adjusting MOI, reaction times, or CRISPR design parameters for subsequent steps.

[0122] The integration with ROS2 & ANML incorporates ROS2 (Robot Operating System 2), which provides a robust pub-sub messaging layer for real-time robot control and sensor feedback. Each lab device or station can be exposed as a ROS2 node. Our system publishes “task instructions” (like “pipette 20 μL reagent X to well #4”) to relevant topics, and listens to “status updates” from the device. The ANML (Action Notation Modeling Language) is used to specify high-level tasks, preconditions, resources, and effects in a domain-agnostic planning format. The system can generate or interpret ANML scripts describing the entire CRISPR workflow (e.g., “For each well in plate, pipette reagent A, wait for 30 min, measure fluorescence.”). The system may also incorporate temporal constraints (like “wash steps must happen no earlier than 10 min after transfection”). ANML scripts can then be executed by an ANML-compliant planning engine or by a bridging layer that dispatches tasks to ROS2. For Hamilton or Opentrons execution, the process begins with task decomposition, where the LLM Agent breaks a CRISPR knockout protocol into atomic steps (pipetting, mixing, incubation, measurement), encoded as an ANML or planning domain definition language (PDDL)-like plan. Translation to robot-specific commands is handled by a Tool Provider or “Lab Robot Service” that transforms high-level steps into G-code-like or Python-based scripts for the chosen robot (Opentrons uses Python protocols, Hamilton has specialized macros). During runtime, the system monitors each step, and if the robot logs an error or if the measured volumes deviate, the plan can be paused or re-planned. Adaptive re-planning is implemented when real-time data indicate suboptimal results—like unexpectedly low transduction efficiency, poor cell viability, or reagent depletion—the system automatically re-plans the next steps. This dynamic adaptation surpasses typical CRISPR-GPT workflows, which do not do iterative re-planning with real-time data from HPC logs or lab sensors.

[0123] In yet another embodiment, the federated system interfaces with automated laboratory robotics platforms (e.g., Opentrons, Hamilton robots) in a closed-loop manner. The system's multi-temporal analysis framework coordinates real-time data collection from laboratory instruments (plate readers, cell imagers, flow cytometers) and merges these sensor logs into ephemeral subgraphs in the knowledge graph. The LLM-driven or ANML or PDDL or alternative planning agent (or agent collective), co-located within the federation manager, then orchestrates updated CRISPR protocols or cell-culture instructions based on immediate feedback from ongoing experiments.

[0124] A specialized ROS2-ANML (Robot Operating System 2-Action Notation Modeling Language) bridge handles command and control. It receives high-level instructions, such as “transfer 20 μL reagent to wells 1-12,” from the LLM agent's ANML script. It then converts these tasks into robot-specific commands (e.g., Opentrons Python protocols) and monitors real-time progress, including pipetting logs, reagent volumes, and sensor outputs. After each robotic step, HPC instrumentation logs are automatically processed to update the ephemeral subgraph, which triggers new or revised tasks if the results deviate significantly from predictions (e.g., unexpectedly low transduction efficiency).

[0125] Additionally, HPC pipelines run advanced planning algorithms such as Monte Carlo Tree Search (MCTS) or reinforcement learning for iterative “experiment states,” choosing next steps to reduce uncertainty or optimize editing efficiency. The system uses ephemeral enclaves to isolate sensitive IP or patient-derived data even on-site. For example, a biotech collaborator can define enclaving policies to protect proprietary small-molecule reagents or novel prime editors. During subsequent rounds, only aggregated or anonymized results are shared across institutions, preventing any leak of unique lab processes or IP-protected reagent compositions.

[0126] For real-time readouts & HPC instrument logs, instrument logs might indicate “transduction

efficiency=15%, below the 30% threshold." The knowledge graph ephemeral subgraph for "Timepoint #1" records that result. The system's HPC pipeline runs immediate analysis—e.g., checking potential reasons for low efficiency (the chosen lentiviral MOI might be too low, or cells might be confluent).

[0127] By directly integrating HPC orchestration and knowledge graphs with real-time lab robotics, the system supports an adaptive, closed-loop experimental design. Researchers can rapidly iterate experimental conditions—e.g., altering CRISPR construct concentrations or vector delivery methods in mid-study—while maintaining secure multi-node collaboration. This significantly extends beyond typical, static CRISPR-GPT-style design-only workflows, enabling dynamic re-planning and scaled automation for high-throughput in-vitro or in-vivo experiments.

[0128] For automated next-step decisions, the system can utilize advanced search or planning algorithms including UTC (Upper Confidence bound for Trees) with super-exponential regret bounds and MCTS+RL (Monte Carlo Tree Search+Reinforcement Learning). A typical lab domain might have transitions and uncertain outcomes, so an RL or MCTS approach can explore different "actions" (like adjusting viral titer or plating density). Alternatively, the system can rely on a hierarchical task network (HTN) or PDDL-based domain model extended with the ANML approach, but to handle dynamic re-planning, we incorporate MCTS+RL or UTC style exploration for better adaptive performance. Human-machine teaming relies on iterative or recursive in vivo and in silico experimentation. The planning engine tries to reduce epistemic uncertainty. The system can propose an update: "Based on the low efficiency, let's double the viral MOI or change to a polybrene concentration from 4 μg/mL to 8 μg/mL." A human operator can confirm or override, with the knowledge graph recording each decision for future reference. The information-theoretic approach allows the system to incorporate an information theory metric to maximize theoretical epistemic uncertainty reduction in the downstream model. For example, if multiple CRISPR conditions are uncertain, the system chooses the next step that yields the greatest expected information gain. This approach can unify HPC-driven simulations (in silico modeling of gene-editing outcomes) with in-lab actions (in vivo validation).

[0129] For continual fine-tuning, domain adaptation, and retrieval augmented generation (RAG), we store new observations in the knowledge corpora, continuously refining domain-specific LLM parameters or RAG contexts. The next iteration of CRISPR-GPT can incorporate these curated updates, improving accuracy or domain coverage. In an example scenario, Round 1 involves the system designing a CRISPR prime editing approach for a certain set of genes in a 96-well plate, with robots performing the protocol and measurement on Day 2. When observation shows 70% wells <10% editing, HPC logs may reveal those wells used a particular reagent batch with questionable quality. For adaptive re-planning, the system decides to reorder a new reagent batch or adjust prime editor concentration, automatically updating the protocol steps in ANML or PDDL, generating new instructions for the lab robot, and re-executing an improved experiment. Through human-machine teaming, a human verifies the proposed changes, fostering iterative/recursive data-driven refinement.

[0130] The implementation layers encompass several key components: The Federation Manager & HPC orchestrates scheduling for lab robot tasks and HPC analysis tasks while maintaining ephemeral knowledge graph subgraphs for each round/timepoint. The ROS2-ANML Bridge manages real-time bridging between high-level planning and low-level robot command messages, subscribing to sensor streams and publishing updated progress or errors. The LLM Agent with MCTS+RL handles complicated multi-step scenarios with unknown yield through tree search or RL to find the best sequence of actions, with user override capabilities. UTC with Super-Exponential Regret provides another advanced approach for handling uncertain multi-armed bandit style decisions. The Information-Theoretic Maximization calculates expected uncertainty reduction in CRISPR-omics models for each potential action. For privacy & security, ephemeral enclaves can be used for sensitive data or HPC-level logs, ensuring no large sequences or personally identifiable

genomic data get exposed outside local bounds.
[0131] Compared to standard CRISPR-GPT, Physical Execution enables active execution via integrated robotics rather than mere instruction provision; Real-Time Data Loop allows ingestion of real-time lab data, HPC logs, and ephemeral subgraph updates for automatic re-planning; Advanced Planning incorporates ANML for action modeling plus MCTS+RL or UTC with advanced regret bounds; Human-Machine Teaming enables user oversight and intervention; and Epistemic Uncertainty Minimization systematically chooses experiments to reduce knowledge gaps. This embodiment thus extends the CRISPR-GPT approach into a fully automated, closed-loop lab environment, delivering iterative and adaptive gene-editing experimentation with integrated robotics, HPC pipelines, advanced planning, and knowledge graph-driven synergy.
[0132] Communication protocols may be implemented through extensible frameworks that could accommodate emerging network technologies and communication patterns. These frameworks might support various approaches to incorporating new communication methods while maintaining security requirements. The system architecture may support different strategies for extending communication capabilities while preserving privacy guarantees across new protocols.
[0133] One or more different aspects may be described in the present application. Further, for one or more of the aspects described herein, numerous alternative arrangements may be described; it should be appreciated that these are presented for illustrative purposes only and are not limiting of the aspects contained herein or the claims presented herein in any way. One or more of the arrangements may be widely applicable to numerous aspects, as may be readily apparent from the disclosure. In general, arrangements are described in sufficient detail to enable those skilled in the art to practice one or more of the aspects, and it should be appreciated that other arrangements may be utilized and that structural, logical, software, electrical and other changes may be made without departing from the scope of the particular aspects. Particular features of one or more of the aspects described herein may be described with reference to one or more particular aspects or figures that form a part of the present disclosure, and in which are shown, by way of illustration, specific arrangements of one or more of the aspects. It should be appreciated, however, that such features are not limited to usage in the one or more particular aspects or figures with reference to which they are described. The present disclosure is neither a literal description of all arrangements of one or more of the aspects nor a listing of features of one or more of the aspects that must be present in all arrangements.
[0134] Headings of sections provided in this patent application and the title of this patent application are for convenience only, and are not to be taken as limiting the disclosure in any way.
[0135] Devices that are in communication with each other need not be in continuous communication with each other, unless expressly specified otherwise. In addition, devices that are in communication with each other may communicate directly or indirectly through one or more communication means or intermediaries, logical or physical.
[0136] A description of an aspect with several components in communication with each other does not imply that all such components are required. To the contrary, a variety of optional components may be described to illustrate a wide variety of possible aspects and in order to more fully illustrate one or more aspects. Similarly, although process steps, method steps, algorithms or the like may be described in a sequential order, such processes, methods and algorithms may generally be configured to work in alternate orders, unless specifically stated to the contrary. In other words, any sequence or order of steps that may be described in this patent application does not, in and of itself, indicate a requirement that the steps be performed in that order. The steps of described processes may be performed in any order practical. Further, some steps may be performed simultaneously despite being described or implied as occurring non-simultaneously (e.g., because one step is described after the other step). Moreover, the illustration of a process by its depiction in a drawing does not imply that the illustrated process is exclusive of other variations and modifications thereto, does not imply that the illustrated process or any of its steps are necessary to one or more of the

aspects, and does not imply that the illustrated process is preferred. Also, steps are generally described once per aspect, but this does not mean they must occur once, or that they may only occur once each time a process, method, or algorithm is carried out or executed. Some steps may be omitted in some aspects or some occurrences, or some steps may be executed more than once in a given aspect or occurrence.

[0137] When a single device or article is described herein, it will be readily apparent that more than one device or article may be used in place of a single device or article. Similarly, where more than one device or article is described herein, it will be readily apparent that a single device or article may be used in place of the more than one device or article.

[0138] The functionality or the features of a device may be alternatively embodied by one or more other devices that are not explicitly described as having such functionality or features. Thus, other aspects need not include the device itself.

[0139] Techniques and mechanisms described or referenced herein will sometimes be described in singular form for clarity. However, it should be appreciated that particular aspects may include multiple iterations of a technique or multiple instantiations of a mechanism unless noted otherwise. Process descriptions or blocks in figures should be understood as representing modules, segments, or portions of code which include one or more executable instructions for implementing specific logical functions or steps in the process. Alternate implementations are included within the scope of various aspects in which, for example, functions may be executed out of order from that shown or discussed, including substantially concurrently or in reverse order, depending on the functionality involved, as would be understood by those having ordinary skill in the art.

Definitions

[0140] As used herein, “federated distributed computational graph” refers to a computational architecture that enables coordinated distributed computing across multiple nodes while maintaining security boundaries and privacy controls between participating entities.

[0141] As used herein, “federation manager” refers to any system component or collection of components that coordinates operations, resources, and communications across multiple computational nodes in a federated system while maintaining prescribed security protocols.

[0142] As used herein, “computational node” refers to any computing resource or collection of computing resources capable of performing biological data processing operations while maintaining prescribed security and privacy controls within the federated system.

[0143] As used herein, “privacy preservation system” refers to any combination of hardware and software components that implement security controls, encryption, access management, or other mechanisms to protect sensitive data during processing and transmission across federated operations.

[0144] As used herein, “knowledge integration component” refers to any system element or collection of elements that manages the organization, storage, retrieval, and relationship mapping of biological data across the federated system while maintaining security boundaries.

[0145] As used herein, “multi-temporal analysis” refers to any approach or methodology for analyzing biological data across multiple time scales while maintaining temporal consistency and enabling dynamic feedback incorporation throughout federated operations.

[0146] As used herein, “genome-scale editing” refers to any process or collection of processes for coordinating and validating genetic modifications across multiple genetic loci while maintaining security controls and privacy requirements.

[0147] As used herein, “biological data” refers to any information related to biological systems, including but not limited to genomic data, protein structures, metabolic pathways, cellular processes, tissue-level interactions, and organism-scale characteristics that may be processed within the federated system.

[0148] As used herein, “secure cross-institutional collaboration” refers to any process or methodology that enables multiple institutions to work together on biological research while

maintaining control over their sensitive data and proprietary methods through privacy-preserving protocols. To bolster cross-institutional data sharing without compromising privacy, the system includes an Advanced Synthetic Data Generation Engine employing copula-based transferable models, variational autoencoders, and diffusion-style generative methods. This engine resides either in the federation manager or as dedicated microservices, ingesting high-dimensional biological data (e.g., gene expression, single-cell multi-omics, epidemiological time-series) across nodes. The system applies advanced transformations—such as Bayesian hierarchical modeling or differential privacy to ensure no sensitive raw data can be reconstructed from the synthetic outputs. During the synthetic data generation pipeline, the knowledge graph engine also contributes topological and ontological constraints. For example, if certain gene pairs are known to co-express or certain metabolic pathways must remain consistent, the generative model enforces these relationships in the synthetic datasets. The ephemeral enclaves at each node optionally participate in cryptographic subroutines that aggregate local parameters without revealing them. Once aggregated, the system trains or fine-tunes generative models and disseminates only the anonymized, synthetic data to collaborator nodes for secondary analyses or machine learning tasks. Institutions can thus engage in robust multi-institutional calibration, using synthetic data to standardize pipeline configurations (e.g., compare off-target detection algorithms) or warm-start machine learning models before final training on local real data. Combining the generative engine with real-time HPC logs further refines the synthetic data to reflect institution-specific HPC usage or error modes. This approach is particularly valuable where data volumes vary widely among partners, ensuring smaller labs or clinics can leverage the system's global model knowledge in a secure, privacy-preserving manner. Such advanced synthetic data generation not only mitigates confidentiality risks but also increases the reproducibility and consistency of distributed studies. Collaborators gain a unified, representative dataset for method benchmarking or pilot exploration without any single entity relinquishing raw, sensitive genomic or phenotypic records. This fosters deeper cross-domain synergy, enabling more reliable, faster progress toward clinically or commercially relevant discoveries.

[0149] As used herein, “synthetic data generation” refers to a sophisticated, multi-layered process or methodology for creating representative data that maintains statistical properties, spatio-temporal relationships, and domain-specific constraints of real data while optionally preserving privacy of source information (or models) and enabling secure collaborative analysis. This process encompasses several key technical approaches and probabilistic, fuzzy, or existential guarantees: At its foundation, the process leverages advanced generative models including diffusion models, variational autoencoders (VAEs), foundation models, and specialized language models fine-tuned on aggregated biological data. These models are integrated with probabilistic programming frameworks that enable the specification of complex generative processes, incorporating priors, likelihoods, and sophisticated sampling schemes that can represent hierarchical models and Bayesian networks. The approach also employs copula-based transferable models that allow the separation of marginal distributions from underlying dependency structures, enabling the transfer of structural relationships from data-rich sources to data-limited target domains. Following the guidelines of a user defined anonymization policy model, a privacy assessment model runs on this representative dataset to identify and analyze data that must be removed, masked, or replaced for final use, and remove any private information. This removal process may include simple deletions of fields or columns where it will not affect the overall data synthesis, or may instead substitute synthetic private data following the same guidelines of maintaining distributions and dependency structures of underlying data, which may require (for situations such as health data) generating a representative population distribution of synthetic patients, with medical history and profiles. The data generation process is enhanced through integration with various knowledge representation systems. This includes spatio-temporal knowledge graphs that capture location-specific constraints, temporal progression, and event-based relationships in biological systems. The knowledge graphs

support advanced reasoning tasks through extended logic engines like Vadalog and Graph Neural Network (GNN)-based inference for multi-dimensional data streams and event-based knowledge graphs for process monitoring and decision support. These knowledge structures enable the synthetic data to maintain complex relationships across temporal, spatial, and event-based dimensions while preserving domain-specific constraints and ontological relationships. Privacy preservation is achieved through multiple complementary mechanisms. The system employs differential privacy techniques during model training, federated learning protocols that ensure raw data never leaves local custody, and homomorphic encryption-based aggregation for secure multi-party computation. Ephemeral enclaves provide additional security by creating temporary, isolated computational environments for sensitive operations. The system implements systems such as membership inference defenses, k-anonymity strategies, and graph-structured privacy protections to prevent reconstruction of individual records or sensitive sequences. The generation process incorporates biological plausibility through multiple validation layers. Domain-specific constraints ensure that synthetic gene sequences respect codon usage frequencies, that epidemiological time-series remain statistically valid while anonymized, and that protein-protein interactions follow established biochemical rules. The system maintains ontological relationships and multi-modal data integration, allowing synthetic data to reflect complex dependencies across molecular, cellular, and population-wide scales. This approach particularly excels at generating synthetic data for challenging scenarios, including rare or underrepresented cases, multi-timepoint experimental designs, and complex multi-omics relationships that may be difficult to obtain from real data alone. The system can generate synthetic populations that reflect realistic socio-demographic or domain-specific distributions, particularly valuable for specialized machine learning training or augmenting small data domains. The synthetic data supports a wide range of downstream applications, including model training, cross-institutional collaboration, and knowledge discovery. It enables institutions to share the statistical essence of their datasets without exposing private information, supports multi-lab synergy, and allows for iterative refinement of models and knowledge bases. The system can produce synthetic data at different scales and granularities, from individual molecular interactions to population-level epidemiological patterns, while maintaining statistical fidelity and causal relationships present in the source data. Importantly, the synthetic data generation process ensures that no individual records, sensitive sequences, proprietary experimental details, or personally identifiable information can be reverse-engineered from the synthetic outputs. This is achieved through careful control of information flow, multiple privacy validation layers, and sophisticated anonymization techniques that preserve utility while protecting sensitive information. The system also supports continuous adaptation and improvement through mechanisms for quality assessment, validation, and refinement. This includes evaluation metrics for synthetic data quality, structural validity checks, and the ability to incorporate new knowledge or constraints as they become available. The process can be dynamically adjusted to meet varying privacy requirements, regulatory constraints, and domain-specific needs while maintaining the fundamental goal of enabling secure, privacy-preserving collaborative analysis in biological and biomedical research contexts.

Conceptual Architecture

[0150] FIG. **1** is a block diagram illustrating exemplary architecture of federated distributed computational graph (FDCG) for biological system engineering and analysis **100**. The federated distributed computational graph architecture described represents one implementation of system **100**, as various alternative arrangements and configurations remain possible while maintaining core system functionality. Subsystems **200**-**600** may be implemented through different technical approaches or combined in alternative configurations based on specific institutional requirements and operational constraints. For example, multi-scale integration framework subsystem **200** and knowledge integration subsystem **400** could be combined into a single processing unit in some implementations, or federation manager subsystem **300** could be distributed across multiple

coordinating nodes rather than operating as a centralized manager. Similarly, genome-scale editing protocol subsystem **500** and multi-temporal analysis framework subsystem **600** may be implemented as separate dedicated hardware units or as software processes running on shared computational infrastructure. This modularity enables system **100** to be adapted for varying computational requirements, security needs, and institutional configurations while preserving the core capabilities of secure cross-institutional collaboration and privacy-preserving data analysis.

[0151] System **100** receives biological data **101** through multi-scale integration framework subsystem **200**, which processes incoming data across molecular, cellular, tissue, and organism levels. Multi-scale integration framework subsystem **200** connects bidirectionally with federation manager subsystem **300**, which coordinates distributed computation and maintains data privacy across system **100**.

[0152] Federation manager subsystem **300** interfaces with knowledge integration subsystem **400**, maintaining data relationships and provenance tracking throughout system **100**. Knowledge integration subsystem **400** provides feedback **130** to multi-scale integration framework subsystem **200**, enabling continuous refinement of data integration processes based on accumulated knowledge.

[0153] System **100** includes two specialized processing subsystems: genome-scale editing protocol subsystem **500** and multi-temporal analysis framework subsystem **600**. These subsystems receive processed data from federation manager subsystem **300** and operate in parallel to perform specific analytical functions. Genome-scale editing protocol subsystem **500** coordinates editing operations and produces genomic analysis output **102**, while providing feedback **110** to federation manager subsystem **300** for real-time validation and optimization. Multi-temporal analysis framework subsystem **600** processes temporal aspects of biological data and generates temporal analysis output **103**, with feedback **120** returning to federation manager subsystem **300** for dynamic adaptation of processing strategies.

[0154] Federation manager subsystem **300** maintains operational coordination across all subsystems while implementing blind execution protocols to preserve data privacy between participating institutions. Knowledge integration subsystem **400** enriches data processing throughout system **100** by maintaining distributed knowledge graphs and vector databases that track relationships between biological entities across multiple scales.

[0155] In another embodiment, the federation manager subsystem implements a Dynamically Partitioned Federated Enclave Framework that creates on-demand enclaves, cither within a single computational node or spanning multiple nodes, to handle specific workflows with heightened confidentiality requirements. Each enclave is an isolated, secure zone formed via trusted execution environments (e.g., Intel SGX, AMD SEV) or lightweight virtualization layers (e.g., microVM hypervisors). The federation manager coordinates secure key exchanges for enclave instantiation and enforces ephemeral lifecycles: enclaves exist only for the duration of the assigned task (e.g., large-scale genome assembly, multi-omic integrative analysis) and then dissolve after final validation.

[0156] When a participating institution requests an enclave for a high-sensitivity workflow—such as analyzing proprietary gene-editing protocols or performing a multi-party computation on patient-derived iPSC data—the blind execution coordinator dynamically segments relevant data and computation graphs. It routes only minimal, tokenized references or irreversibly hashed identifiers into the enclave, thereby preventing infiltration of nonessential or sensitive data. All in-enclave memory, checkpointing, and storage buffers remain cryptographically isolated from other processes. An ephemeral key pair is used to secure enclaved computations, with cryptographic proofs confirming correct task execution to outside nodes.

[0157] This fine-grained enclaving extends beyond static node-level compartmentalization by providing targeted enclaves that can “fork” or subdivide at runtime. For instance, a single HPC node with multiple GPU cores could simultaneously host multiple enclaves in parallel for different

institutional workflows, ensuring each remains cryptographically walled off. Alternatively, enclaves may be chained across node boundaries under a controlled communication policy, enabling partial-homomorphic data calculations that require distributed, ephemeral enclaves. Upon completion, the federation manager triggers a secure teardown procedure, wiping ephemeral keys and overwriting memory buffers to guarantee no residual data is accessible.

[0158] This dynamically partitioned enclave paradigm is particularly advantageous for cross-institutional collaborations requiring variable security levels, such as joint CRISPR design and toxicology assessment. Some participants might require ultra-strict enclaving of partial data subsets, while others can operate under lighter enclaving. By matching enclaving policies to evolving security requirements in real time, the system provides a flexible yet robust approach to data confidentiality in federated multi-institutional computations.

[0159] The interconnected feedback loops **110**, **120**, and **130** enable system **100** to continuously optimize its operations based on accumulated knowledge and analysis results while maintaining security protocols and institutional boundaries. This architecture supports secure cross-institutional collaboration for biological system engineering and analysis through coordinated data processing and privacy-preserving protocols.

[0160] Biological data **101** enters system **100** through multi-scale integration framework subsystem **200**, which processes and standardizes data across molecular, cellular, tissue, and organism levels. Processed data flows from multi-scale integration framework subsystem **200** to federation manager subsystem **300**, which coordinates distribution of computational tasks while maintaining privacy through blind execution protocols. Federation manager subsystem **300** interfaces with knowledge integration subsystem **400** to enrich data processing with contextual relationships and maintain data provenance tracking.

[0161] Federation manager subsystem **300** directs processed data to specialized subsystems based on analysis requirements. For genomic analysis, data flows to genome-scale editing protocol subsystem **500**, which coordinates editing operations and generates genomic analysis output **102**. For temporal analysis, data flows to multi-temporal analysis framework subsystem **600**, which processes time-based aspects of biological data and produces temporal analysis output **103**.

[0162] System **100** incorporates three feedback paths that enable continuous optimization. Feedback **110** flows from genome-scale editing protocol subsystem **500** to federation manager subsystem **300**, providing real-time validation of editing operations. Feedback **120** flows from multi-temporal analysis framework subsystem **600** to federation manager subsystem **300**, enabling dynamic adaptation of processing strategies. Feedback **130** flows from knowledge integration subsystem **400** to multi-scale integration framework subsystem **200**, refining data integration processes based on accumulated knowledge.

[0163] Throughout data processing, federation manager subsystem **300** maintains security protocols and institutional boundaries while coordinating operations across all subsystems. This coordinated data flow for data in motion and as persisted along with provenance information for data, models, and processes along with software and hardware bills of materials enables secure cross-institutional collaboration while preserving data privacy requirements and enables better science with more reproducibility and traceability.

[0164] FIG. **2** is a block diagram illustrating exemplary architecture of multi-scale integration framework **200**. Multi-scale integration framework **200** comprises several interconnected subsystems for processing biological data across multiple scales. Multi-scale integration framework **200** may implement a comprehensive biological data processing architecture through coordinated operation of specialized subsystems. The framework may process biological data across multiple scales of organization while maintaining consistency and enabling dynamic adaptation.

[0165] Molecular processing engine subsystem **210** handles integration of protein, RNA, and metabolite data, processing incoming molecular-level information and coordinating with cellular system coordinator subsystem **220**. Molecular processing engine subsystem **210** may implement

sophisticated molecular data integration through various analytical approaches. For example, it may process protein structural data using advanced folding algorithms, analyze RNA expression patterns through statistical methods, and integrate metabolite profiles using pathway mapping techniques. The subsystem may, for instance, employ machine learning models trained on molecular interaction data to identify patterns and predict relationships between different molecular components. These capabilities may be enhanced through real-time analysis of molecular dynamics and interaction networks.

[0166] Cellular system coordinator subsystem **220** manages cell-level data and pathway analysis, bridging molecular and tissue-scale information processing. Cellular system coordinator subsystem **220** may bridge molecular and tissue-scale processing through multi-level data integration approaches. The subsystem may, for example, analyze cellular pathways using graph-based algorithms while maintaining connections to both molecular-scale interactions and tissue-level effects. It may implement adaptive processing workflows that can adjust to varying cellular conditions and experimental protocols.

[0167] Tissue integration layer subsystem **230** coordinates tissue-level data processing, working in conjunction with organism scale manager subsystem **240** to maintain consistency across biological scales. Tissue integration layer subsystem **230** may coordinate processing of tissue-level biological data through various analytical frameworks. For example, it may analyze tissue organization patterns, process inter-cellular communication networks, and maintain tissue-scale mathematical models. The subsystem may implement specialized algorithms for handling three-dimensional tissue structures and analyzing spatial relationships between different cell types.

[0168] Organism scale manager subsystem **240** handles organism-level data integration, ensuring cohesive analysis across all biological levels. Organism scale manager subsystem **240** may maintain cohesive analysis across biological scales through sophisticated coordination protocols. It may, for instance, implement hierarchical data models that preserve relationships between tissue-level observations and organism-wide effects. The subsystem may employ adaptive scaling mechanisms that adjust analysis parameters based on organism-specific characteristics.

[0169] Cross-scale synchronization subsystem **250** maintains consistency between these different scales of biological organization, implementing machine learning models to identify patterns and relationships across scales. Cross-scale synchronization subsystem **250** may implement advanced pattern recognition capabilities through various machine learning approaches. For example, it may employ neural networks trained on multi-scale biological data to identify relationships between molecular events and organism-level outcomes. The subsystem may maintain dynamic models that adapt to new patterns as they emerge across different scales of biological organization.

[0170] Temporal resolution handler subsystem **260** manages different time scales across biological processes, coordinating with data stream integration subsystem **270** to process real-time inputs across scales. Temporal resolution handler subsystem **260** may process biological events across multiple time scales through sophisticated synchronization protocols. For example, it may coordinate analysis of rapid molecular interactions alongside slower developmental processes, implementing adaptive sampling strategies that maintain temporal coherence across scales.

[0171] Data stream integration subsystem **270** coordinates incoming data streams from various sources, ensuring proper temporal alignment and scale-appropriate processing. Data stream integration subsystem **270** may manage incoming biological data through various processing pipelines optimized for different data types and temporal scales. The subsystem may, for instance, implement real-time data validation, normalization, and integration protocols while maintaining scale-appropriate processing parameters. It may employ adaptive filtering mechanisms that adjust to varying data quality and sampling rates.

[0172] Through these coordinated mechanisms, multi-scale integration framework **200** may enable comprehensive analysis of biological systems across multiple scales of organization while maintaining consistency and enabling dynamic adaptation to changing experimental conditions.

[0173] Multi-scale integration framework **200** receives biological data **101** through data stream integration subsystem **270**, which distributes incoming data to appropriate scale-specific processing subsystems. Processed data flows through cross-scale synchronization subsystem **250**, which maintains consistency across all processing layers. Framework **200** interfaces with federation manager subsystem **300** for coordinated processing across system **100**, while receiving feedback **130** from knowledge integration subsystem **400** to refine integration processes based on accumulated knowledge.

[0174] This architecture enables coordinated processing of biological data across multiple scales while maintaining temporal consistency and proper relationships between different levels of biological organization. Implementation of machine learning models throughout framework **200** supports pattern recognition and cross-scale relationship identification, particularly within molecular processing engine subsystem **210** and cross-scale synchronization subsystem **250**.

[0175] In multi-scale integration framework **200**, machine learning models are implemented primarily within molecular processing engine subsystem **210** and cross-scale synchronization subsystem **250**. Molecular processing engine subsystem **210** utilizes deep learning models trained on molecular interaction data to identify patterns and predict interactions between proteins, RNA molecules, and metabolites. These models employ convolutional neural networks for processing structural data and transformer architectures for sequence analysis, trained using standardized molecular datasets while maintaining privacy through federated learning approaches.

[0176] Cross-scale synchronization subsystem **250** implements transfer learning techniques to apply knowledge gained at one biological scale to others. This subsystem employs hierarchical neural networks trained on multi-scale biological data, enabling pattern recognition across different levels of biological organization. Training occurs through a distributed process coordinated by federation manager subsystem **300**, allowing multiple institutions to contribute to model improvement while preserving data privacy.

[0177] Implementation of these machine learning components occurs through distributed tensor processing units integrated within framework **200**'s computational infrastructure. Models in molecular processing engine subsystem **210** operate on incoming molecular data streams, generating predictions and pattern analyses that flow to cellular system coordinator subsystem **220**. Cross-scale synchronization subsystem **250** continuously processes outputs from all scale-specific subsystems, using transfer learning to maintain consistency and identify relationships across scales.

[0178] Model training procedures incorporate privacy-preserving techniques such as differential privacy and secure aggregation, enabling collaborative improvement of model performance without exposing sensitive institutional data. Regular model updates occur through federated averaging protocols coordinated by federation manager subsystem **300**, ensuring consistent performance across distributed deployments while maintaining security boundaries.

[0179] Framework **200** requires data validation protocols at each processing level to maintain data integrity across scales. Input validation occurs at data stream integration subsystem **270**, which implements format checking and data quality assessment before distribution to scale-specific processing subsystems. Each scale-specific subsystem incorporates error detection and correction mechanisms to handle inconsistencies in biological data processing.

[0180] Resource management capabilities within framework **200** enable dynamic allocation of computational resources based on processing demands. This includes load balancing across processing units and prioritization of critical analytical pathways. Framework **200** maintains processing queues for each scale-specific subsystem, coordinating workload distribution through cross-scale synchronization subsystem **250**.

[0181] State management and recovery mechanisms ensure operational continuity during processing interruptions or failures. Each subsystem maintains state information enabling recovery from interruptions without data loss. Checkpoint systems within cross-scale synchronization subsystem **250** preserve processing state across multiple scales, facilitating recovery of multi-scale

analyses.
[0182] Integration with external reference databases occurs through molecular processing engine subsystem **210** and organism scale manager subsystem **240**, enabling validation against established biological knowledge. These connections operate through secure protocols coordinated by federation manager subsystem **300** to maintain system security.
[0183] Data versioning capabilities track changes and updates across all processing scales, enabling reproducibility of analyses and maintaining audit trails. This versioning system operates across all subsystems, coordinated through cross-scale synchronization subsystem **250**.
[0184] In multi-scale integration framework **200**, data flows through interconnected processing paths designed to enable comprehensive biological analysis across scales. Biological data **101** enters through data stream integration subsystem **270**, which directs incoming data to molecular processing engine subsystem **210**. Data then progresses linearly through scale-specific processing, flowing from molecular processing engine subsystem **210** to cellular system coordinator subsystem **220**, then to tissue integration layer subsystem **230**, and finally to organism scale manager subsystem **240**. Each scale-specific subsystem additionally sends its processed data to cross-scale synchronization subsystem **250**, which implements transfer learning to identify patterns and relationships across biological scales. Cross-scale synchronization subsystem **250** coordinates with temporal resolution handler subsystem **260** to maintain temporal consistency before sending integrated results to federation manager subsystem **300**. Knowledge integration subsystem **400** provides feedback **130** to cross-scale synchronization subsystem **250**, enabling continuous refinement of cross-scale pattern recognition and analysis capabilities.
[0185] FIG. **3** is a block diagram illustrating exemplary architecture of federation manager subsystem **300**. Federation manager subsystem **300** receives biological data through multi-scale integration framework subsystem **200** and coordinates processing across system **100** through several interconnected components while maintaining security protocols and data privacy requirements. The architecture illustrated in **300** implements the core federated distributed computational graph (FDCG) that forms the foundation of the system. In this graph structure, each node comprises a complete system **100** implementation, serving as a vertex in the computational graph. The federation manager subsystem **300** establishes and manages edges between these vertices through node communication subsystem **350**, creating a dynamic graph topology that enables secure distributed computation. These edges represent both data flows and computational relationships between nodes, with the blind execution coordinator subsystem **320** and distributed task scheduler subsystem **330** working in concert to route computations through the resulting graph structure. The federation manager subsystem **300** maintains this graph topology through resource tracking subsystem **310**, which monitors the capabilities and availability of each vertex, and security protocol engine subsystem **340**, which ensures secure communication along graph edges. This FDCG architecture enables flexible scaling and reconfiguration, as new vertices can be dynamically added to the graph through the establishment of new system **100** implementations, with the federation manager subsystem **300** automatically incorporating these new nodes into the existing graph structure while maintaining security protocols and institutional boundaries. The recursive nature of this architecture, where each vertex represents a complete system implementation capable of independent operation, creates a robust and adaptable computational graph that can efficiently coordinate distributed biological data analysis while preserving data privacy and operational autonomy.
[0186] Federation manager subsystem **300** coordinates operations between multiple implementations of system **100**, each operating as a distinct computational entity within the federated architecture. Each system **100** implementation contains its complete suite of subsystems, enabling autonomous operation while participating in federated processing through coordination between their respective federation manager subsystems **300**.
[0187] When federation manager subsystem **300** distributes computational tasks, it communicates

with federation manager subsystems **300** of other system **100** implementations through their respective node communication subsystems **350**. This enables secure collaboration while maintaining institutional boundaries, as each system **100** implementation maintains control over its local resources and data through its own multi-scale integration framework subsystem **200**, knowledge integration subsystem **400**, genome-scale editing protocol subsystem **500**, and multi-temporal analysis framework subsystem **600**.

[0188] Resource tracking subsystem **310** monitors available computational resources across participating system **100** implementations, while blind execution coordinator subsystem **320** manages secure distributed processing operations between them. Distributed task scheduler subsystem **330** coordinates workflow execution across multiple system **100** implementations, with security protocol engine subsystem **340** maintaining privacy boundaries between distinct system **100** instances.

[0189] This architectural approach enables flexible federation patterns, as each system **100** implementation may participate in multiple collaborative relationships while maintaining operational independence. The recursive nature of the architecture, where each computational node is a complete system **100** implementation, provides consistent capabilities and interfaces across the federation while preserving institutional autonomy and security requirements.

[0190] Through this coordinated interaction between system **100** implementations, federation manager subsystem **300** enables secure cross-institutional collaboration while maintaining data privacy and operational independence. Each system **100** implementation may contribute its computational resources and specialized capabilities to federated operations while maintaining control over its sensitive data and proprietary methods. Federation manager subsystem **300** may implement the federated distributed computational graph through coordinated operation of its core components. The graph structure may, for example, represent a dynamic network where each vertex may serve as a complete system **100** implementation, and edges may represent secure communication channels for data exchange and computational coordination.

[0191] Resource tracking subsystem **310** monitors computational resources and node capabilities across system **100**, maintaining real-time status information and resource availability. Resource tracking subsystem **310** interfaces with blind execution coordinator subsystem **320**, providing resource allocation data for secure distributed processing operations. Resource tracking subsystem **310** may maintain the graph topology through various monitoring and update cycles. For example, it may implement a distributed state management protocol that can track each vertex's status, potentially including current processing load, available specialized capabilities, and operational state. When system state changes occur, such as the addition of new computational capabilities or changes in resource availability, resource tracking subsystem **310** may update the graph topology accordingly. This subsystem may, for instance, maintain a distributed registry of vertex capabilities that enables efficient task routing and resource allocation across the federation.

[0192] Blind execution coordinator subsystem **320** implements privacy-preserving computation protocols that enable collaborative analysis while maintaining data privacy between participating nodes. Blind execution coordinator subsystem **320** works in conjunction with distributed task scheduler subsystem **330** to coordinate secure processing operations across institutional boundaries. Blind execution coordinator subsystem **320** may transform computational operations to enable secure processing across graph edges while maintaining vertex autonomy. When coordinating cross-institutional computation, it may, for example, implement a multi-phase protocol: First, it may analyze the computational requirements and data sensitivity levels. Then, it may generate privacy-preserving transformation patterns that can enable collaborative computation without exposing sensitive data between vertices. The system may, for instance, establish secure execution contexts that maintain isolation between participating system **100** implementations while enabling coordinated processing.

[0193] Distributed task scheduler subsystem **330** manages workflow orchestration and task

distribution across computational nodes based on resource availability and processing requirements. Distributed task scheduler subsystem **330** interfaces with security protocol engine subsystem **340** to ensure task execution maintains prescribed security policies. Distributed task scheduler subsystem **330** may implement graph-aware task distribution through various scheduling protocols. For example, it may analyze both the graph topology and current vertex states to determine optimal task routing paths. The scheduler may maintain multiple concurrent execution contexts, each potentially representing a distributed computation spanning multiple vertices. These contexts may, for instance, track task dependencies, resource requirements, and security constraints across the graph structure. When new tasks enter the system, the scheduler may analyze the graph topology to identify suitable execution paths that can satisfy both computational and security requirements.

[0194] Security protocol engine subsystem **340** enforces access controls and privacy policies across federated operations, working with node communication subsystem **350** to maintain secure information exchange between participating nodes. Security protocol engine subsystem **340** implements encryption protocols for data protection during processing and transmission. Security protocol engine subsystem **340** may establish and maintain secure graph edges through various security management approaches. It may, for instance, implement distributed security protocols that ensure inter-vertex communications maintain prescribed privacy requirements. The protocols may include, for example, validation of security credentials, monitoring of communication patterns, and re-establishment of secure channels if security parameters change.

[0195] Node communication subsystem **350** handles messaging and synchronization between computational nodes, enabling secure information exchange while maintaining institutional boundaries. Node communication subsystem **350** implements standardized protocols for data transmission and operational coordination across system **100**. Node communication subsystem **350** may maintain the implementation of graph edges through various communication channels. It may, for instance, implement messaging protocols that ensure delivery of both control messages and data across graph edges. Such protocols may include, for example, channel encryption, message validation, and acknowledgment mechanisms that maintain communication integrity across the federation.

[0196] Through these mechanisms, federation manager subsystem **300** may maintain a graph structure that enables secure collaborative computation while preserving the operational independence of each vertex. The system may continuously adapt the graph topology to reflect changing computational requirements and security constraints, enabling efficient cross-institutional collaboration while maintaining privacy boundaries.

[0197] Federation manager subsystem **300** coordinates with knowledge integration subsystem **400** for tracking data relationships and provenance, genome-scale editing protocol subsystem **500** for coordinating editing operations, and multi-temporal analysis framework subsystem **600** for temporal data processing. These interactions occur through defined interfaces while maintaining security protocols and privacy requirements.

[0198] Through coordination of these components, federation manager subsystem **300** enables secure collaborative computation across institutional boundaries while preserving data privacy and maintaining operational efficiency. Federation manager subsystem **300** provides centralized coordination while enabling distributed processing through computational nodes operating within prescribed security boundaries.

[0199] Federation manager subsystem **300** incorporates machine learning capabilities within resource tracking subsystem **310** and blind execution coordinator subsystem **320** to enhance system performance and security. Resource tracking subsystem **310** implements gradient-boosted decision tree models trained on historical resource utilization data to predict computational requirements and optimize allocation across nodes. These models process features including CPU utilization, memory consumption, network bandwidth, and task completion times to forecast resource needs

and detect potential bottlenecks.
[0200] Blind execution coordinator subsystem **320** employs federated learning techniques through distributed neural networks that enable collaborative model training while maintaining data privacy. These models implement secure aggregation protocols during training, allowing nodes to contribute to model improvement without exposing sensitive institutional data. Training occurs through iterative model updates using encrypted gradients, with model parameters aggregated securely through multi-party computation protocols.
[0201] Resource tracking subsystem **310** maintains separate prediction models for different types of biological computations, including genomic analysis, protein folding, and pathway modeling. These models are continuously refined through online learning approaches as new performance data becomes available, enabling adaptive resource optimization based on evolving computational patterns.
[0202] The machine learning implementations within federation manager subsystem **300** operate through distributed tensor processing units integrated within system **100**'s computational infrastructure. Model training procedures incorporate differential privacy techniques to prevent information leakage during collaborative learning processes. Regular model updates occur through secure aggregation protocols that maintain privacy while enabling continuous improvement of system performance.
[0203] Federation manager subsystem **300** coordinates model deployment across computational nodes through standardized interfaces that abstract underlying implementation details. This enables consistent performance across heterogeneous hardware configurations while maintaining security boundaries during model execution and training operations.
[0204] Through these machine learning capabilities, federation manager subsystem **300** achieves efficient resource utilization and secure collaborative computation while preserving institutional data privacy requirements. The combination of predictive resource optimization and privacy-preserving learning techniques enables effective cross-institutional collaboration within prescribed security constraints.
[0205] The machine learning models within federation manager subsystem **300** may be trained through various approaches using different types of data. For example, resource tracking subsystem **310** may train its predictive models on historical system performance data, which may include CPU and memory utilization patterns, network bandwidth consumption, task completion times, and resource allocation histories. This training data may be collected during system operation and may be used to continuously refine prediction accuracy.
[0206] Training procedures for blind execution coordinator subsystem **320** may implement federated learning approaches where model updates may occur without centralizing sensitive data. For example, each participating node may compute model updates locally, and these updates may be aggregated securely through encryption protocols that preserve data privacy while enabling model improvement.
[0207] The training data may incorporate various biological computation patterns. For example, models may learn from genomic analysis workflows, protein structure predictions, or pathway modeling tasks. These diverse training examples may help models adapt to different types of computational requirements and resource utilization patterns.
[0208] Models may also be trained on synthetic data generated through privacy-preserving techniques. For example, generative models may create representative computational patterns that maintain statistical properties of real workloads while protecting sensitive information. This synthetic training data may enable robust model development without exposing institutional data.
[0209] The training process may implement transfer learning approaches where knowledge gained from one type of biological computation may be applied to others. For example, models trained on protein folding workflows may transfer relevant features to RNA structure prediction tasks, potentially improving performance across different types of analyses.

[0210] Model training may occur through distributed optimization procedures that maintain security boundaries. For example, secure aggregation protocols may enable collaborative model improvement while preventing any single institution from accessing sensitive data from others. These protocols may implement differential privacy techniques to prevent information leakage during training.

[0211] Federation manager subsystem **300** may implement comprehensive scaling, state management, and recovery mechanisms to maintain operational reliability. Resource scaling capabilities may include dynamic adjustment of computational resources based on processing demands and node availability. For example, federation manager subsystem **300** may automatically scale processing capacity by activating additional nodes during periods of high demand, while maintaining security protocols across scaling operations.

[0212] State management capabilities may include distributed checkpointing mechanisms that track computation progress across federated operations. For example, federation manager subsystem **300** may maintain state information through secure snapshot protocols that enable workflow recovery without compromising privacy requirements. These snapshots may capture essential operational parameters while excluding sensitive data, enabling secure state restoration across institutional boundaries.

[0213] Error handling and recovery mechanisms may incorporate multiple layers of fault detection and response protocols. For example, federation manager subsystem **300** may implement heartbeat monitoring systems that detect node failures or communication interruptions. Recovery procedures may include automatic failover mechanisms that redistribute processing tasks while maintaining security boundaries and data privacy requirements.

[0214] The system may implement transaction management protocols that maintain consistency during distributed operations. For example, federation manager subsystem **300** may coordinate two-phase commit procedures across participating nodes to ensure atomic operations complete successfully or roll back without compromising system integrity. These protocols may enable reliable distributed processing while preserving security requirements during recovery operations.

[0215] Federation manager subsystem **300** may maintain operational continuity through redundant processing pathways. For example, critical computational tasks may be replicated across multiple nodes with secure verification protocols ensuring consistent results. This redundancy may enable continuous operation during node failures while maintaining prescribed security protocols and privacy requirements.

[0216] These capabilities may work in concert to enable reliable operation of federation manager subsystem **300** across varying computational loads and potential system disruptions. The combination of dynamic resource scaling, secure state management, and robust error recovery may support consistent performance while maintaining security boundaries during normal operation and recovery scenarios.

[0217] Federation manager subsystem **300** processes data through coordinated flows across its component subsystems, in various embodiments. Initial data enters federation manager subsystem **300** from multi-scale integration framework subsystem **200**, where it is first received by resource tracking subsystem **310** for workload analysis and resource allocation.

[0218] Resource tracking subsystem **310** processes the incoming data to determine computational requirements, utilizing predictive models to assess resource needs. This processed resource allocation data flows to blind execution coordinator subsystem **320**, which partitions the computational tasks into secure processing units while maintaining data privacy requirements.

[0219] From blind execution coordinator subsystem **320**, the partitioned tasks flow to distributed task scheduler subsystem **330**, which coordinates task distribution across available computational nodes **399** based on resource availability and processing requirements. The scheduled tasks then pass through security protocol engine subsystem **340**, where they are encrypted and prepared for secure transmission.

[0220] Node communication subsystem **350** receives the secured tasks from security protocol engine subsystem **340** and manages their distribution to appropriate computational nodes. Results from node processing flow back through node communication subsystem **350**, where they are validated by security protocol engine subsystem **340** before being aggregated by blind execution coordinator subsystem **320**.

[0221] The aggregated results flow through established interfaces to knowledge integration subsystem **400** for relationship tracking, genome-scale editing protocol subsystem **500** for editing operations, and multi-temporal analysis framework subsystem **600** for temporal processing. Feedback from these subsystems returns through node communication subsystem **350**, enabling continuous optimization of processing operations.

[0222] Throughout these data flows, federation manager subsystem **300** maintains secure channels and privacy boundaries while enabling efficient distributed computation across institutional boundaries. The coordinated flow of data through these subsystems enables collaborative biological analysis while preserving security requirements and operational efficiency.

[0223] FIG. **4** is a block diagram illustrating exemplary architecture of knowledge integration subsystem **400**. Knowledge integration subsystem **400** processes biological data through coordinated operation of specialized components designed to maintain data relationships while preserving security protocols. Knowledge integration subsystem **400** may implement a comprehensive biological knowledge management architecture through coordinated operation of specialized components, in various embodiments. The subsystem may process and integrate biological data while maintaining security protocols and enabling cross-institutional collaboration.

[0224] Vector database subsystem **410** implements efficient storage and retrieval of biological data through specialized indexing structures optimized for high-dimensional data types. Vector database subsystem **410** interfaces with knowledge graph engine subsystem **420**, enabling relationship tracking across biological entities while maintaining data privacy requirements. Vector database subsystem **410** may implement advanced data storage and retrieval capabilities through various specialized indexing approaches. For example, it may utilize high-dimensional indexing structures optimized for biological data types such as protein sequences, metabolic profiles, and gene expression patterns. The subsystem may, for instance, employ locality-sensitive hashing techniques that enable efficient similarity searches while maintaining privacy constraints. These indexing structures may adapt dynamically to accommodate new biological data types and changing query patterns.

[0225] Knowledge graph engine subsystem **420** maintains distributed graph databases that track relationships between biological entities across multiple scales. Knowledge graph engine subsystem **420** coordinates with temporal versioning subsystem **430** to track changes in biological relationships over time while preserving data lineage. Knowledge graph engine subsystem **420** may maintain distributed biological relationship networks through sophisticated graph database implementations. The subsystem may, for example, represent molecular interactions, cellular pathways, and organism-level relationships as interconnected graph structures that preserve biological context. It may implement distributed consensus protocols that enable collaborative graph updates while maintaining data sovereignty across institutional boundaries. The engine may employ advanced graph algorithms that can identify complex relationship patterns across multiple biological scales.

[0226] Temporal versioning subsystem **430** implements version control for biological data, maintaining historical records of changes while enabling reproducible analysis. Temporal versioning subsystem **430** works in conjunction with provenance tracking subsystem **440** to maintain complete data lineage across federated operations. Temporal versioning subsystem **430** may implement comprehensive version control mechanisms through various temporal management approaches. For example, it may maintain complete histories of biological relationship changes while enabling reproducible analysis across different time points. The subsystem may, for instance,

implement branching and merging protocols that allow parallel development of biological models while maintaining consistency. These versioning capabilities may include sophisticated diff algorithms optimized for biological data types.
[0227] Provenance tracking subsystem **440** records data sources and transformations throughout processing operations, ensuring traceability while maintaining security protocols. Provenance tracking subsystem **440** interfaces with ontology management subsystem **450** to maintain consistent terminology across institutional boundaries. Provenance tracking subsystem **440** may maintain complete data lineage through various tracking mechanisms designed for biological data workflows. The subsystem may, for example, record transformation operations, data sources, and processing parameters while preserving security protocols. It may implement distributed provenance protocols that maintain consistency across federated operations while enabling secure auditing capabilities. The tracking system may employ cryptographic techniques that ensure provenance records cannot be altered without detection.
[0228] Ontology management subsystem **450** implements standardized biological terminology and relationship definitions, enabling consistent interpretation across federated operations. Ontology management subsystem **450** coordinates with query processing subsystem **460** to enable standardized data retrieval across distributed storage systems. Ontology management subsystem **450** may implement biological terminology standardization through sophisticated semantic frameworks. For example, it may maintain mappings between institutional terminologies and standard references while preserving local naming conventions. The subsystem may, for instance, employ machine learning approaches that can suggest terminology alignments based on context and usage patterns. These capabilities may include automated consistency checking and conflict resolution mechanisms.
[0229] Query processing subsystem **460** handles distributed data retrieval operations while maintaining security protocols and privacy requirements. Query processing subsystem **460** implements secure search capabilities across vector database subsystem **410** and knowledge graph engine subsystem **420**, enabling efficient data access while preserving privacy constraints. Query processing subsystem **460** may handle distributed data retrieval through various secure search implementations. The subsystem may, for example, implement federated query protocols that maintain privacy while enabling comprehensive search across distributed resources. It may employ advanced query optimization techniques that consider both computational efficiency and security constraints. The processing engine may implement various access control mechanisms that enforce institutional policies while enabling collaborative analysis.
[0230] Through these coordinated mechanisms, knowledge integration subsystem **400** may enable sophisticated biological knowledge management while preserving security requirements and enabling efficient cross-institutional collaboration. The system may continuously adapt to changing data types, relationship patterns, and security requirements while maintaining consistent operation across federated environments.
[0231] Knowledge integration subsystem **400** receives processed data from federation manager subsystem **300** through established interfaces while maintaining feedback loop **130** to multi-scale integration framework subsystem **200**. This architecture enables secure knowledge integration across institutional boundaries while preserving data privacy and maintaining operational efficiency through coordinated component operation.
[0232] Through these interconnected subsystems, knowledge integration subsystem **400** maintains comprehensive biological data relationships while enabling secure cross-institutional collaboration. Coordinated operation of these components supports efficient data storage, relationship tracking, and secure retrieval operations while preserving privacy requirements and security protocols across federated operations.
[0233] Knowledge integration subsystem **400** incorporates machine learning capabilities throughout its components to enable sophisticated data analysis and relationship modeling.

Knowledge graph engine subsystem **420** may implement graph neural networks trained on biological interaction data to analyze and predict relationships between entities. These models may process features including protein-protein interactions, metabolic pathways, and gene regulatory networks to identify complex biological relationships across different scales.

[0234] Query processing subsystem **460** may employ natural language processing models to standardize and interpret biological terminology across institutional boundaries. These models may be trained on curated biological ontologies and literature databases, enabling consistent query interpretation while maintaining privacy requirements. Training may incorporate transfer learning approaches where knowledge gained from public datasets may be applied to institution-specific terminology.

[0235] Vector database subsystem **410** may utilize embedding models to represent biological entities in high-dimensional space, enabling efficient similarity searches while preserving privacy. These models may learn representations from various biological data types, including protein sequences, molecular structures, and pathway information. Training procedures may implement privacy-preserving techniques that enable model improvement without exposing sensitive institutional data.

[0236] The machine learning implementations within knowledge integration subsystem **400** may operate through distributed tensor processing units integrated within system **100**'s computational infrastructure. Model training procedures may incorporate differential privacy techniques to prevent information leakage during collaborative learning processes. Regular model updates may occur through secure aggregation protocols that maintain privacy while enabling continuous improvement of system performance.

[0237] Knowledge graph engine subsystem **420** may maintain separate prediction models for different types of biological relationships, including molecular interactions, cellular pathways, and organism-level associations. These models may be continuously refined through online learning approaches as new relationship data becomes available, enabling adaptive optimization based on emerging biological patterns.

[0238] Through these machine learning capabilities, knowledge integration subsystem **400** may achieve sophisticated relationship analysis and efficient data organization while preserving institutional data privacy requirements. The combination of graph neural networks, natural language processing, and embedding models may enable effective biological knowledge integration within prescribed security constraints.

[0239] Knowledge integration subsystem **400** processes data through coordinated flows across its component subsystems. Initial data enters from federation manager subsystem **300**, flowing first to vector database subsystem **410** for embedding and storage. Vector database subsystem **410** processes incoming data to create high-dimensional representations, passing these to knowledge graph engine subsystem **420** for relationship analysis and graph structure integration. Knowledge graph engine subsystem **420** coordinates with temporal versioning subsystem **430** and provenance tracking subsystem **440** to maintain data history and lineage throughout processing operations. As data flows through these subsystems, ontology management subsystem **450** ensures consistent terminology mapping, while query processing subsystem **460** handles data retrieval requests from other parts of system **100**. Processed data flows back to multi-scale integration framework subsystem **200** through feedback loop **130**, enabling continuous refinement of integration processes. Throughout these operations, each subsystem maintains secure processing protocols while enabling efficient data access and relationship tracking across institutional boundaries.

[0240] FIG. **5** is a block diagram illustrating exemplary architecture of genome-scale editing protocol subsystem **500**. Genome-scale editing protocol subsystem **500** coordinates genetic modification operations through interconnected components designed to maintain precision and security across editing operations. In accordance with various embodiments, genome-scale editing protocol subsystem **500** may implement different architectural configurations while maintaining

core editing and security capabilities. For example, some implementations may combine validation engine subsystem **520** and safety verification subsystem **570** into a unified validation framework, while others may maintain them as separate components. Similarly, off-target analysis subsystem **530** and repair pathway predictor subsystem **540** may be implemented either as distinct subsystems or as an integrated prediction engine, depending on specific institutional requirements and operational constraints.

[0241] The modular nature of genome-scale editing protocol subsystem **500** enables flexible adaptation to different operational environments while preserving essential security protocols and editing capabilities. Some implementations may incorporate additional specialized components beyond those described, while others may implement streamlined architectures that combine multiple functions within unified processing units. This architectural flexibility enables institutions to implement configurations that align with their specific requirements while maintaining consistent security protocols and editing capabilities across different deployment patterns.

[0242] These variations in component organization and implementation demonstrate the adaptability of genome-scale editing protocol subsystem **500** while preserving its fundamental capabilities for secure genetic modification operations. The system architecture supports multiple implementation patterns while maintaining essential security protocols and operational efficiency across different configurations.

[0243] CRISPR design coordinator subsystem **510** manages edit design across multiple genetic loci through pattern recognition and optimization algorithms. This subsystem processes sequence data to identify optimal guide RNA configurations, incorporating chromatin accessibility data and structural predictions to maximize editing efficiency. CRISPR design coordinator subsystem **510** interfaces with validation engine subsystem **520** to verify proposed edits before execution, transmitting both guide RNA designs and predicted efficiency metrics.

[0244] Validation engine subsystem **520** performs real-time verification of editing operations through analysis of modification outcomes and safety parameters. This subsystem implements multi-stage validation protocols that assess both computational predictions and experimental results, incorporating feedback from previous editing operations to refine validation criteria. Validation engine subsystem **520** coordinates with off-target analysis subsystem **530** to monitor potential unintended effects during editing processes, maintaining continuous assessment throughout execution.

[0245] Off-target analysis subsystem **530** predicts and tracks effects beyond intended edit sites through computational modeling and pattern analysis. This subsystem employs genome-wide sequence similarity scanning and chromatin state analysis to identify potential off-target locations, generating comprehensive risk assessments for each proposed edit. Off-target analysis subsystem **530** works in conjunction with repair pathway predictor subsystem **540** to model DNA repair mechanisms and outcomes, enabling integrated assessment of both immediate and long-term effects.

[0246] Repair pathway predictor subsystem **540** models cellular repair responses to genetic modifications through analysis of repair mechanism patterns. This subsystem incorporates cell-type specific factors and environmental conditions to predict repair outcomes, generating probability distributions for different repair pathways. Repair pathway predictor subsystem **540** interfaces with database integration subsystem **550** to incorporate reference data into prediction models, enabling continuous refinement of repair forecasting capabilities.

[0247] Database integration subsystem **550** connects with genomic databases while maintaining security protocols and privacy requirements. This subsystem implements secure query interfaces and data transformation protocols, enabling reference data access while preserving institutional privacy boundaries. Database integration subsystem **550** coordinates with edit orchestration subsystem **560** to provide reference data for editing operations, supporting real-time decision-making during execution.

[0248] Edit orchestration subsystem **560** coordinates parallel editing operations across multiple genetic loci while maintaining process consistency. This subsystem implements sophisticated scheduling algorithms that optimize editing efficiency while managing resource utilization and maintaining data privacy across operations. Edit orchestration subsystem **560** interfaces with safety verification subsystem **570** to ensure compliance with security protocols, enabling secure execution of complex editing patterns.

[0249] Safety verification subsystem **570** monitors editing operations for compliance with safety requirements and institutional protocols. This subsystem implements real-time monitoring capabilities that track both individual edits and cumulative effects, maintaining comprehensive safety assessments throughout execution. Safety verification subsystem **570** works with result integration subsystem **580** to maintain security during result aggregation, ensuring privacy preservation during outcome analysis.

[0250] Result integration subsystem **580** combines and analyzes outcomes from multiple editing operations while preserving data privacy. This subsystem implements secure aggregation protocols that enable comprehensive analysis while maintaining institutional boundaries and data privacy requirements. Result integration subsystem **580** provides feedback through loop **110** to federation manager subsystem **300**, enabling real-time optimization of editing processes through secure communication channels. Genome-scale editing protocol subsystem **500** coordinates with federation manager subsystem **300** through established interfaces while maintaining feedback loop **110** for continuous process refinement. This architecture enables precise genetic modification operations while preserving security protocols and privacy requirements through coordinated component operation.

[0251] Genome-scale editing protocol subsystem **500** incorporates machine learning capabilities across several key components. CRISPR design coordinator subsystem **510** may implement deep neural networks trained on genomic sequence data to predict editing efficiency and optimize guide RNA design. These models may process features including sequence composition, chromatin accessibility, and structural properties to identify optimal editing sites. Training data may incorporate results from previous editing operations while maintaining privacy through federated learning approaches.

[0252] Off-target analysis subsystem **530** may employ convolutional neural networks trained on genome-wide sequence data to predict potential unintended editing effects. These models may analyze sequence similarity patterns and chromatin state information to identify possible off-target sites. Training may utilize public genomic databases combined with secured institutional data, enabling robust prediction while preserving data privacy.

[0253] Repair pathway predictor subsystem **540** may implement probabilistic graphical models to forecast DNA repair outcomes following editing operations. These models may learn from observed repair patterns across multiple cell types and editing conditions, incorporating both sequence context and cellular state information. Training procedures may employ bayesian approaches to handle uncertainty in repair pathway selection.

[0254] The machine learning implementations within genome-scale editing protocol subsystem **500** may operate through distributed tensor processing units integrated within system **100**'s computational infrastructure. Model training procedures may incorporate differential privacy techniques to prevent information leakage during collaborative learning processes. Regular model updates may occur through secure aggregation protocols that maintain privacy while enabling continuous improvement of editing accuracy.

[0255] Edit orchestration subsystem **560** may utilize reinforcement learning approaches to optimize parallel editing operations, learning from successful editing patterns while maintaining security protocols. These models may adapt to varying cellular conditions and editing requirements through online learning mechanisms that preserve institutional privacy boundaries.

[0256] Through these machine learning capabilities, genome-scale editing protocol subsystem **500**

may achieve precise genetic modifications while preserving data privacy requirements. The combination of deep learning, probabilistic modeling, and reinforcement learning may enable effective editing operations within prescribed security constraints.
[0257] Genome-scale editing protocol subsystem **500** may implement comprehensive error handling and recovery mechanisms to maintain operational reliability. For example, fault detection protocols may identify various types of editing failures, including guide RNA mismatches, insufficient editing efficiency, or validation errors. Recovery procedures may include automated rollback mechanisms that restore editing operations to previous known-good states while maintaining security protocols.
[0258] State management capabilities within genome-scale editing protocol subsystem **500** may include distributed checkpointing mechanisms that track editing progress across multiple genetic loci. For example, edit orchestration subsystem **560** may maintain secure state snapshots that capture editing parameters, validation results, and safety verification status. These snapshots may enable secure recovery without compromising editing precision or data privacy.
[0259] The system may implement transaction management protocols that maintain consistency during distributed editing operations. For example, edit orchestration subsystem **560** may coordinate two-phase commit procedures across editing operations to ensure modifications complete successfully or roll back without compromising genome integrity. These protocols may enable reliable editing operations while preserving security requirements during recovery scenarios.
[0260] Genome-scale editing protocol subsystem **500** may maintain operational continuity through redundant validation pathways. For example, critical editing operations may undergo parallel validation through multiple instances of validation engine subsystem **520**, with secure verification protocols ensuring consistent results. This redundancy may enable continuous operation during component failures while maintaining prescribed security protocols and privacy requirements.
[0261] These capabilities may work together to enable reliable operation of genome-scale editing protocol subsystem **500** across varying editing loads and potential system disruptions. The combination of robust error handling, secure state management, and comprehensive recovery protocols may support consistent editing performance while maintaining security boundaries during both normal operation and recovery scenarios.
[0262] Genome-scale editing protocol subsystem **500** processes data through coordinated flows across its component subsystems. Initial data enters from federation manager subsystem **300** through CRISPR design coordinator subsystem **510**, which analyzes sequence information and generates edit designs. These designs flow to validation engine subsystem **520** for initial verification before proceeding to parallel analysis paths.
[0263] From validation engine subsystem **520**, data flows simultaneously to off-target analysis subsystem **530** and repair pathway predictor subsystem **540**. Off-target analysis subsystem **530** examines potential unintended effects, while repair pathway predictor subsystem **540** forecasts repair outcomes. Both subsystems interface with database integration subsystem **550** to incorporate reference data into their analyses.
[0264] Results from these analyses converge at edit orchestration subsystem **560**, which coordinates execution of verified editing operations. Edit orchestration subsystem **560** sends execution data to safety verification subsystem **570** for compliance monitoring. Safety verification subsystem **570** passes verified results to result integration subsystem **580**, which aggregates outcomes and generates feedback.
[0265] Result integration subsystem **580** sends processed data through feedback loop **110** to federation manager subsystem **300**, enabling continuous optimization of editing processes. Throughout these operations, each subsystem maintains secure processing protocols while enabling efficient coordination of editing operations across multiple genetic loci.
[0266] Database integration subsystem **550** provides reference data flows to multiple subsystems

simultaneously, supporting operations of CRISPR design coordinator subsystem **510**, validation engine subsystem **520**, off-target analysis subsystem **530**, and repair pathway predictor subsystem **540**. These coordinated data flows enable comprehensive analysis while maintaining security protocols and privacy requirements across editing operations.

[0267] FIG. **6** is a block diagram illustrating exemplary architecture of multi-temporal analysis framework subsystem **600**. Multi-temporal analysis framework subsystem **600** processes biological data across multiple time scales through coordinated operation of specialized components designed to maintain temporal consistency while enabling dynamic adaptation. In accordance with various embodiments, multi-temporal analysis framework subsystem **600** may implement different architectural configurations while maintaining core temporal analysis and security capabilities. For example, some implementations may combine temporal scale manager subsystem **610** and temporal synchronization subsystem **640** into a unified temporal coordination framework, while others may maintain them as separate components. Similarly, rhythm analysis subsystem **650** and scale translation subsystem **660** may be implemented either as distinct subsystems or as an integrated pattern analysis engine, depending on specific institutional requirements and operational constraints. The modular nature of multi-temporal analysis framework subsystem **600** enables flexible adaptation to different operational environments while preserving essential security protocols and analytical capabilities. Some implementations may incorporate additional specialized components beyond those described, while others may implement streamlined architectures that combine multiple functions within unified processing units. This architectural flexibility enables institutions to implement configurations that align with their specific requirements while maintaining consistent security protocols and temporal analysis capabilities across different deployment patterns.

[0268] Temporal scale manager subsystem **610** coordinates analysis across different time domains through synchronization of temporal data streams. For example, this subsystem may process data ranging from millisecond-scale molecular interactions to day-scale organism responses, implementing adaptive sampling rates to maintain temporal resolution across scales. Temporal scale manager subsystem **610** may include specialized timing protocols that enable coherent analysis across multiple time domains while preserving causal relationships. This subsystem interfaces with feedback integration subsystem **620** to incorporate dynamic updates into temporal models, potentially enabling real-time adaptation of temporal analysis strategies.

[0269] Feedback integration subsystem **620** handles real-time model updating through continuous processing of analytical results. This subsystem may implement sliding window analyses that incorporate new data while maintaining historical context, for example, adjusting model parameters based on emerging temporal patterns. Feedback integration subsystem **620** may include adaptive learning mechanisms that enable dynamic response to changing biological conditions. This subsystem coordinates with cross-node validation subsystem **630** to verify temporal consistency across distributed operations, potentially implementing secure validation protocols.

[0270] Cross-node validation subsystem **630** verifies analysis results through comparison of temporal patterns across computational nodes. For example, this subsystem may implement consensus protocols that ensure consistent temporal interpretation across distributed analyses while maintaining privacy boundaries. Cross-node validation subsystem **630** may include pattern matching algorithms that identify and resolve temporal inconsistencies. This subsystem works in conjunction with temporal synchronization subsystem **640** to maintain time-based consistency across operations.

[0271] Temporal synchronization subsystem **640** maintains consistency between different time scales through coordinated timing protocols. This subsystem may implement hierarchical synchronization mechanisms that align analyses across multiple temporal resolutions while preserving causal relationships. For example, temporal synchronization subsystem **640** may include phase-locking algorithms that maintain temporal coherence across distributed operations. This

subsystem interfaces with rhythm analysis subsystem **650** to process biological cycles and periodic patterns while maintaining temporal alignment.

[0272] Rhythm analysis subsystem **650** processes biological rhythms and cycles through pattern recognition and temporal modeling. This subsystem may implement spectral analysis techniques that identify periodic patterns across multiple time scales, for example, detecting circadian rhythms alongside faster metabolic oscillations. Rhythm analysis subsystem **650** may include wavelet analysis capabilities that enable multi-scale decomposition of temporal patterns. This subsystem coordinates with scale translation subsystem **660** to enable coherent analysis across different temporal scales.

[0273] Scale translation subsystem **660** converts between different time scales through mathematical transformation and pattern matching. For example, this subsystem may implement adaptive resampling algorithms that maintain signal fidelity across temporal transformations while preserving essential biological patterns. Scale translation subsystem **660** may include interpolation mechanisms that enable smooth transitions between different temporal resolutions. This subsystem interfaces with historical data manager subsystem **670** to incorporate past observations into current analyses while maintaining temporal consistency.

[0274] Historical data manager subsystem **670** maintains temporal data archives while preserving security protocols and privacy requirements. This subsystem may implement secure compression algorithms that enable efficient storage of temporal data while maintaining accessibility for analysis. For example, historical data manager subsystem **670** may include versioning mechanisms that track changes in temporal patterns over extended periods. This subsystem coordinates with prediction subsystem **680** to support forecasting operations through secure access to historical data.

[0275] Prediction subsystem **680** models future states based on temporal patterns through analysis of historical trends and current conditions. This subsystem may implement ensemble forecasting methods that combine multiple prediction models to improve accuracy while maintaining uncertainty estimates. For example, prediction subsystem **680** may include adaptive forecasting algorithms that adjust prediction horizons based on data quality and pattern stability. This subsystem provides feedback through loop **120** to federation manager subsystem **300**, potentially enabling continuous refinement of temporal analysis processes through secure communication channels.

[0276] Multi-temporal analysis framework subsystem **600** coordinates with federation manager subsystem **300** through established interfaces while maintaining feedback loop **120** for process optimization. This architecture enables comprehensive temporal analysis while preserving security protocols and privacy requirements through coordinated component operation.

[0277] Multi-temporal analysis framework subsystem **600** incorporates machine learning capabilities throughout its components. Prediction subsystem **680** may implement recurrent neural networks trained on temporal biological data to forecast system behavior across multiple time scales. These models may process features including gene expression patterns, metabolic fluctuations, and cellular state transitions to identify temporal dependencies. Training data may incorporate both historical observations and real-time measurements while maintaining privacy through federated learning approaches.

[0278] Scale translation subsystem **660** may employ transformer models trained on multi-scale temporal data to enable conversion between different time domains. These models may analyze patterns across molecular, cellular, and organism-level timescales to identify relationships between temporal processes. Training may utilize synchronized temporal data streams while preserving institutional privacy through secure aggregation protocols.

[0279] Rhythm analysis subsystem **650** may implement specialized time series models to characterize biological rhythms and periodic patterns. These models may learn from observed biological cycles across multiple scales, incorporating both frequency domain and time domain features. Training procedures may employ ensemble methods to handle varying cycle lengths and

phase relationships while maintaining security requirements.
[0280] The machine learning implementations within multi-temporal analysis framework subsystem **600** may operate through distributed tensor processing units integrated within system **100**'s computational infrastructure. Model training procedures may incorporate differential privacy techniques to prevent information leakage during collaborative learning processes. Regular model updates may occur through secure aggregation protocols that maintain privacy while enabling continuous improvement of temporal analysis accuracy.
[0281] Temporal synchronization subsystem **640** may utilize attention mechanisms to identify relevant temporal relationships across different time scales. These models may adapt to varying temporal resolutions and sampling rates through online learning mechanisms that preserve institutional privacy boundaries.
[0282] Through these machine learning capabilities, multi-temporal analysis framework subsystem **600** may achieve sophisticated temporal analysis while preserving data privacy requirements. The combination of recurrent networks, transformer models, and specialized time series analysis may enable effective temporal modeling within prescribed security constraints.
[0283] Multi-temporal analysis framework subsystem **600** processes data through coordinated flows across its component subsystems. Initial data enters from federation manager subsystem **300** through temporal scale manager subsystem **610**, which coordinates temporal alignment and processing across different time domains.
[0284] From temporal scale manager subsystem **610**, data flows to feedback integration subsystem **620** for incorporation of dynamic updates and real-time adjustments. Feedback integration subsystem **620** sends processed data to cross-node validation subsystem **630**, which verifies temporal consistency across distributed operations.
[0285] Cross-node validation subsystem **630** coordinates with temporal synchronization subsystem **640** to maintain time-based consistency across scales. Temporal synchronization subsystem **640** directs synchronized data to rhythm analysis subsystem **650** for processing of biological cycles and periodic patterns.
[0286] Rhythm analysis subsystem **650** sends identified patterns to scale translation subsystem **660**, which converts analyses between different temporal scales. Scale translation subsystem **660** coordinates with historical data manager subsystem **670** to incorporate past observations into current analyses.
[0287] Historical data manager subsystem **670** provides archived temporal data to prediction subsystem **680**, which generates forecasts and future state predictions. Prediction subsystem **680** sends processed results through feedback loop **120** to federation manager subsystem **300**, enabling continuous refinement of temporal analysis processes.
[0288] Throughout these operations, each subsystem maintains secure processing protocols while enabling efficient coordination of temporal analyses across multiple time scales. Temporal synchronization subsystem **640** provides timing coordination to all subsystems simultaneously, ensuring consistent temporal alignment across all processing operations while maintaining security protocols and privacy requirements.
[0289] This coordinated data flow enables comprehensive temporal analysis while preserving security boundaries between system components and participating institutions. Each connection represents secure data transmission channels between subsystems, supporting sophisticated temporal analysis while maintaining prescribed security protocols.
[0290] FIG. **7** is a method diagram illustrating the initial node federation process, in an embodiment. A new computational node is activated and broadcasts its presence to federation manager subsystem **300** via node communication subsystem **350**, initiating the secure federation protocol **701**. Resource tracking subsystem **310** validates the new node's hardware specifications, computational capabilities, and security protocols through standardized verification procedures that assess processing power, memory allocation, and network bandwidth capabilities **702**. Security

protocol engine **340** establishes an encrypted communication channel with the new node and performs initial security handshake operations to verify node authenticity through multi-factor cryptographic validation **703**. The new node's local privacy preservation subsystem transmits its privacy requirements and data handling policies to federation manager subsystem **300** for validation against federation-wide security standards and institutional compliance requirements **704**. Blind execution coordinator **320** configures secure computation protocols between the new node and existing federation members based on validated privacy policies, establishing encrypted channels for future collaborative processing **705**. Federation manager subsystem **300** updates its distributed resource inventory through resource tracking subsystem **310** to include the new node's capabilities and constraints, enabling efficient task allocation and resource optimization across the federation **706**. Knowledge integration subsystem **400** establishes secure connections with the new node's local knowledge components to enable privacy-preserving data relationship mapping while maintaining institutional boundaries and data sovereignty **707**. Distributed task scheduler **330** incorporates the new node into its task allocation framework based on the node's registered capabilities and security boundaries, preparing the node for participation in federated computations **708**. Federation manager subsystem **300** finalizes node integration by broadcasting updated federation topology to all nodes and activating the new node for distributed computation, completing the secure federation process **709**.

[0291] FIG. **8** is a method diagram illustrating distributed computation workflow in system **100**, in an embodiment. A biological analysis task is received by federation manager subsystem **300** through node communication subsystem **350** and validated by security protocol engine **340** for processing requirements and privacy constraints, initiating the secure distributed computation process **801**. Blind execution coordinator **320** decomposes the analysis task into discrete computational units while preserving data privacy through selective information masking and encryption, ensuring that sensitive biological data remains protected throughout processing **802**. Resource tracking subsystem **310** evaluates current federation capabilities and node availability to determine optimal task distribution patterns across the computational graph, considering factors such as processing capacity, specialized capabilities, and historical performance metrics **803**. Distributed task scheduler **330** assigns computational units to specific nodes based on their capabilities, current workload, and security boundaries while maintaining privacy requirements and ensuring efficient resource utilization across the federation **804**. Multi-scale integration framework subsystem **200** at each participating node processes its assigned computational units through molecular processing engine subsystem **210** and cellular system coordinator subsystem **220**, applying specialized algorithms while maintaining data isolation **805**. Knowledge integration subsystem **400** securely aggregates intermediate results through vector database subsystem **410** and knowledge graph engine subsystem **420** while maintaining data privacy and tracking provenance across distributed operations **806**. Cross-node validation protocols verify computational integrity across participating nodes through secure multi-party computation mechanisms, ensuring consistent and accurate processing while preserving institutional boundaries **807**. Result integration subsystem **580** combines validated results while preserving privacy constraints through secure aggregation protocols that enable comprehensive analysis without exposing sensitive data **808**. Federation manager subsystem **300** returns final analysis results to the requesting node and updates distributed knowledge repositories with privacy-preserving insights, completing the secure distributed computation workflow **809**.

[0292] FIG. **9** is a method diagram illustrating knowledge integration process in system **100**, in an embodiment. Knowledge integration subsystem **400** receives biological data through federation manager subsystem **300** and initiates secure integration protocols through vector database subsystem **410**, establishing secure channels for cross-institutional data processing **901**. Vector database subsystem **410** processes incoming biological data into high-dimensional representations while maintaining privacy through differential privacy mechanisms, enabling efficient similarity

searches without exposing sensitive information **902**. Knowledge graph engine subsystem **420** analyzes data relationships and updates its distributed graph structure while preserving institutional boundaries, implementing secure graph operations that maintain data sovereignty across participating nodes **903**. Temporal versioning subsystem **430** establishes versioning controls and maintains temporal consistency across newly integrated data relationships, ensuring reproducibility while preserving historical context of biological relationships **904**. Provenance tracking subsystem **440** records data lineage and transformation histories while ensuring compliance with privacy requirements, maintaining comprehensive audit trails without exposing sensitive institutional information **905**. Ontology management subsystem **450** aligns biological terminology and relationships across institutional boundaries through standardized mapping protocols, enabling consistent interpretation while preserving institutional terminologies **906**. Query processing subsystem **460** validates integration results through secure distributed queries across participating nodes, verifying relationship consistency while maintaining privacy controls **907**. Cross-node knowledge synchronization is performed through secure consensus protocols while maintaining privacy boundaries, ensuring consistent biological relationship representations across the federation **908**. Knowledge integration subsystem **400** transmits integration status through feedback loop **130** to multi-scale integration framework subsystem **200** for continuous refinement, enabling adaptive optimization of integration processes **909**.

[0293] FIG. **10** is a method diagram illustrating multi-temporal analysis workflow in system **100**, in an embodiment. Multi-temporal analysis framework subsystem **600** receives biological data through federation manager subsystem **300** for processing across multiple time scales via temporal scale manager subsystem **610**, initiating secure temporal analysis protocols **1001**. Temporal scale manager subsystem **610** coordinates temporal domain synchronization across distributed nodes while maintaining privacy boundaries through secure timing protocols, establishing coherent time-based processing frameworks across the federation **1002**. Feedback integration subsystem **620** incorporates real-time processing results into temporal models through dynamic feedback mechanisms, enabling adaptive refinement of temporal analyses while preserving data privacy **1003**. Cross-node validation subsystem **630** verifies temporal consistency across distributed operations through secure validation protocols, ensuring synchronized analysis across institutional boundaries **1004**. Temporal synchronization subsystem **640** aligns analyses across multiple temporal resolutions while preserving causal relationships between biological events, maintaining coherent temporal relationships from molecular to organism-level timescales **1005**. Rhythm analysis subsystem **650** identifies biological cycles and periodic patterns through secure pattern recognition algorithms, detecting temporal regularities while maintaining privacy controls **1006**. Scale translation subsystem **660** performs secure conversions between different temporal scales while maintaining pattern fidelity, enabling comprehensive analysis across diverse biological rhythms and frequencies **1007**. Historical data manager subsystem **670** securely integrates archived temporal data with current analyses through privacy-preserving access protocols, incorporating historical context while maintaining data security **1008**. Prediction subsystem **680** generates forecasts through ensemble learning approaches and transmits results through feedback loop **120** to federation manager subsystem **300**, completing the temporal analysis workflow with privacy-preserved predictions **1009**.

[0294] FIG. **11** is a method diagram illustrating genome-scale editing process in system **100**, in an embodiment. Genome-scale editing protocol subsystem **500** receives editing requests through federation manager subsystem **300** and initiates secure editing protocols via CRISPR design coordinator subsystem **510**, establishing privacy-preserved channels for cross-node editing operations **1101**. CRISPR design coordinator subsystem **510** analyzes sequence data and generates optimized guide RNA designs while maintaining privacy through secure computation protocols, incorporating chromatin accessibility data and structural predictions to maximize editing efficiency **1102**. Validation engine subsystem **520** performs initial verification of proposed edits through

multi-stage validation protocols across distributed nodes, implementing real-time assessment of computational predictions and experimental parameters **1103**. Off-target analysis subsystem **530** conducts comprehensive risk assessment through secure genome-wide analysis of potential unintended effects, employing machine learning models to predict off-target probabilities while maintaining data privacy **1104**. Repair pathway predictor subsystem **540** forecasts cellular repair outcomes through privacy-preserving machine learning models, incorporating cell-type specific factors and environmental conditions to generate repair probability distributions **1105**. Database integration subsystem **550** securely incorporates reference data into editing analyses while maintaining institutional boundaries, enabling validated comparisons without compromising sensitive information **1106**. Edit orchestration subsystem **560** coordinates parallel editing operations across multiple genetic loci through secure scheduling protocols, optimizing editing efficiency while preserving privacy requirements **1107**. Safety verification subsystem **570** monitors editing operations for compliance with security and safety requirements across the federation, tracking both individual modifications and cumulative effects **1108**. Result integration subsystem **580** aggregates editing outcomes through secure protocols and transmits results via feedback loop **110** to federation manager subsystem **300**, completing the editing workflow while maintaining privacy boundaries **1109**.

[0295] In a non-limiting use case example of an embodiment of federated distributed computational graph (FDCG) for biological system engineering and analysis **100**, three research institutions collaborate on analyzing drug resistance patterns in bacterial populations while maintaining privacy of their proprietary strain collections and experimental data. Each institution operates as a computational node within system **100**, with federation manager subsystem **300** coordinating secure analysis across institutional boundaries.

[0296] The first institution contributes genomic sequencing data from antibiotic-resistant bacterial strains, the second institution provides historical antibiotic effectiveness data, and the third institution contributes protein structure data for relevant resistance mechanisms. Federation manager subsystem **300** decomposes the analysis task through blind execution coordinator **320**, enabling each institution to process portions of the analysis without accessing other institutions' sensitive data.

[0297] Multi-scale integration framework subsystem **200** processes data across molecular, cellular, and population scales, while knowledge integration subsystem **400** securely maps relationships between resistance mechanisms, genetic markers, and treatment outcomes. Multi-temporal analysis framework subsystem **600** analyzes the evolution of resistance patterns over time, identifying emerging trends while maintaining institutional privacy.

[0298] Through this federated collaboration, the institutions successfully identify novel resistance patterns and potential therapeutic targets without compromising their proprietary data. The resulting insights are securely shared through federation manager subsystem **300**, with each institution maintaining control over their contribution level to subsequent research efforts.

[0299] In another non-limiting use case example, system **100** enables secure collaboration between a biotechnology company and multiple academic institutions studying cellular aging mechanisms. The biotechnology company operates a primary node containing proprietary data about cellular rejuvenation factors, while academic partners maintain nodes with specialized aging research data from various model organisms.

[0300] Federation manager subsystem **300** establishes secure processing channels that allow analysis of aging pathways across species while protecting the company's intellectual property and the institutions' unpublished research data. Multi-scale integration framework subsystem **200** correlates molecular markers of aging across different organisms, while knowledge integration subsystem **400** builds secure relationship maps between aging mechanisms and potential interventions.

[0301] Multi-temporal analysis framework subsystem **600** processes longitudinal aging data across

different time scales, from rapid cellular responses to long-term organismal changes. The system's privacy-preserving protocols enable identification of conserved aging mechanisms without exposing sensitive experimental methods or proprietary compounds.

[0302] In a third non-limiting example, system **100** facilitates collaboration between medical research centers studying rare genetic disorders. Each center maintains a node containing sensitive patient genetic data and clinical histories. Federation manager subsystem **300** coordinates privacy-preserving analysis across these nodes, enabling pattern recognition in disease progression without compromising patient privacy.

[0303] Genome-scale editing protocol subsystem **500** evaluates potential therapeutic strategies across multiple genetic loci, while multi-temporal analysis framework subsystem **600** tracks disease progression patterns. Knowledge integration subsystem **400** securely maps relationships between genetic variations and clinical outcomes, enabling insights that would be impossible for any single institution to derive independently.

[0304] In another non-limiting use case example of an embodiment of federated distributed computational graph (FDCG) for biological system engineering and analysis **100**, a network of research institutions studies protein interaction networks across multiple organisms. The computational graph initially consists of five nodes, each representing a complete system **100** implementation at different institutions. Federation manager subsystem **300** establishes edges between these nodes based on their computational capabilities and security protocols, creating a dynamic graph topology for distributed analysis.

[0305] When processing protein interaction data, federation manager subsystem **300** decomposes analysis tasks into subgraphs of computational operations. For example, when analyzing a specific protein pathway, one edge in the graph carries structural analysis tasks between two nodes with specialized molecular modeling capabilities, while another edge routes interaction prediction tasks between nodes with advanced machine learning implementations. Blind execution coordinator **320** ensures that these graph edges maintain data privacy during computation.

[0306] As analysis demands increase, three additional institutions join the federation, causing federation manager subsystem **300** to dynamically reconfigure the computational graph. New edges are established based on the incoming nodes' capabilities, creating additional parallel processing paths while maintaining security boundaries. The resulting expanded graph enables more efficient distribution of computational tasks while preserving the privacy guarantees essential for cross-institutional collaboration.

[0307] These use case examples demonstrate how the FDCG architecture adapts its graph topology to optimize biological data analysis across a growing network of institutional nodes while maintaining secure edges for privacy-preserving computation.

[0308] The potential applications of system **100** extend well beyond biological research and engineering. The federated distributed computational graph architecture could be adapted for any domain requiring secure cross-institutional collaboration and privacy-preserving distributed computation. For instance, the system could enable secure collaboration in fields such as healthcare analytics, drug development, materials science, environmental monitoring, or financial modeling. The fundamental capabilities of maintaining data privacy while enabling sophisticated distributed analysis could support research ranging from climate modeling to quantum systems. Similarly, the system's ability to coordinate multi-scale and temporal analyses while preserving institutional boundaries could benefit applications in fields like sustainable energy development, advanced manufacturing, or predictive maintenance. The modular nature of the architecture allows for adaptation to various computational requirements while maintaining essential security protocols. These examples are provided for illustration only and should not be construed as limiting the scope or applicability of the system's fundamental architecture and capabilities.

Exemplary Computing Environment

[0309] FIG. **12** illustrates an exemplary computing environment on which an embodiment

described herein may be implemented, in full or in part. This exemplary computing environment describes computer-related components and processes supporting enabling disclosure of computer-implemented embodiments. Inclusion in this exemplary computing environment of well-known processes and computer components, if any, is not a suggestion or admission that any embodiment is no more than an aggregation of such processes or components. Rather, implementation of an embodiment using processes and components described in this exemplary computing environment will involve programming or configuration of such processes and components resulting in a machine specially programmed or configured for such implementation. The exemplary computing environment described herein is only one example of such an environment and other configurations of the components and processes are possible, including other relationships between and among components, and/or absence of some processes or components described. Further, the exemplary computing environment described herein is not intended to suggest any limitation as to the scope of use or functionality of any embodiment implemented, in whole or in part, on components or processes described herein.

[0310] The exemplary computing environment described herein comprises a computing device **10** (further comprising a system bus **11**, one or more processors **20**, a system memory **30**, one or more interfaces **40**, one or more non-volatile data storage devices **50**), external peripherals and accessories **60**, external communication devices **70**, remote computing devices **80**, and cloud-based services **90**.

[0311] System bus **11** couples the various system components, coordinating operation of and data transmission between those various system components. System bus **11** represents one or more of any type or combination of types of wired or wireless bus structures including, but not limited to, memory busses or memory controllers, point-to-point connections, switching fabrics, peripheral busses, accelerated graphics ports, and local busses using any of a variety of bus architectures. By way of example, such architectures include, but are not limited to, Industry Standard Architecture (ISA) busses, Micro Channel Architecture (MCA) busses, Enhanced ISA (EISA) busses, Video Electronics Standards Association (VESA) local busses, a Peripheral Component Interconnects (PCI) busses also known as a Mezzanine busses, or any selection of, or combination of, such busses. Depending on the specific physical implementation, one or more of the processors **20**, system memory **30** and other components of the computing device **10** can be physically co-located or integrated into a single physical component, such as on a single chip. In such a case, some or all of system bus **11** can be electrical pathways within a single chip structure.

[0312] Computing device may further comprise externally-accessible data input and storage devices **12** such as compact disc read-only memory (CD-ROM) drives, digital versatile discs (DVD), or other optical disc storage for reading and/or writing optical discs **62**; magnetic cassettes, magnetic tape, magnetic disk storage, or other magnetic storage devices; or any other medium which can be used to store the desired content and which can be accessed by the computing device **10**. Computing device may further comprise externally-accessible data ports or connections **12** such as serial ports, parallel ports, universal serial bus (USB) ports, and infrared ports and/or transmitter/receivers. Computing device may further comprise hardware for wireless communication with external devices such as IEEE 1394 (“Firewire”) interfaces, IEEE 802.11 wireless interfaces, BLUETOOTH® wireless interfaces, and so forth. Such ports and interfaces may be used to connect any number of external peripherals and accessories **60** such as visual displays, monitors, and touch-sensitive screens **61**, USB solid state memory data storage drives (commonly known as “flash drives” or “thumb drives”) **63**, printers **64**, pointers and manipulators such as mice **65**, keyboards **66**, and other devices **67** such as joysticks and gaming pads, touchpads, additional displays and monitors, and external hard drives (whether solid state or disc-based), microphones, speakers, cameras, and optical scanners.

[0313] Processors **20** are logic circuitry capable of receiving programming instructions and processing (or executing) those instructions to perform computer operations such as retrieving data,

storing data, and performing mathematical calculations. Processors **20** are not limited by the materials from which they are formed or the processing mechanisms employed therein, but are typically comprised of semiconductor materials into which many transistors are formed together into logic gates on a chip (i.e., an integrated circuit or IC). The term processor includes any device capable of receiving and processing instructions including, but not limited to, processors operating on the basis of quantum computing, optical computing, mechanical computing (e.g., using nanotechnology entities to transfer data), and so forth. Depending on configuration, computing device **10** may comprise more than one processor. For example, computing device **10** may comprise one or more central processing units (CPUs) **21**, each of which itself has multiple processors or multiple processing cores, each capable of independently or semi-independently processing programming instructions based on technologies like complex instruction set computer (CISC) or reduced instruction set computer (RISC). Further, computing device **10** may comprise one or more specialized processors such as a graphics processing unit (GPU) **22** configured to accelerate processing of computer graphics and images via a large array of specialized processing cores arranged in parallel. Further computing device **10** may be comprised of one or more specialized processes such as Intelligent Processing Units, field-programmable gate arrays or application-specific integrated circuits for specific tasks or types of tasks. The term processor may further include: neural processing units (NPUs) or neural computing units optimized for machine learning and artificial intelligence workloads using specialized architectures and data paths; tensor processing units (TPUs) designed to efficiently perform matrix multiplication and convolution operations used heavily in neural networks and deep learning applications; application-specific integrated circuits (ASICs) implementing custom logic for domain-specific tasks; application-specific instruction set processors (ASIPs) with instruction sets tailored for particular applications; field-programmable gate arrays (FPGAs) providing reconfigurable logic fabric that can be customized for specific processing tasks; processors operating on emerging computing paradigms such as quantum computing, optical computing, mechanical computing (e.g., using nanotechnology entities to transfer data), and so forth. Depending on configuration, computing device **10** may comprise one or more of any of the above types of processors in order to efficiently handle a variety of general purpose and specialized computing tasks. The specific processor configuration may be selected based on performance, power, cost, or other design constraints relevant to the intended application of computing device **10**.

[0314] System memory **30** is processor-accessible data storage in the form of volatile and/or nonvolatile memory. System memory **30** may be either or both of two types: non-volatile memory and volatile memory. Non-volatile memory **30***a* is not erased when power to the memory is removed, and includes memory types such as read only memory (ROM), electronically-erasable programmable memory (EEPROM), and rewritable solid state memory (commonly known as "flash memory"). Non-volatile memory **30***a* is typically used for long-term storage of a basic input/output system (BIOS) **31**, containing the basic instructions, typically loaded during computer startup, for transfer of information between components within computing device, or a unified extensible firmware interface (UEFI), which is a modern replacement for BIOS that supports larger hard drives, faster boot times, more security features, and provides native support for graphics and mouse cursors. Non-volatile memory **30***a* may also be used to store firmware comprising a complete operating system **35** and applications **36** for operating computer-controlled devices. The firmware approach is often used for purpose-specific computer-controlled devices such as appliances and Internet-of-Things (IoT) devices where processing power and data storage space is limited. Volatile memory **30***b* is erased when power to the memory is removed and is typically used for short-term storage of data for processing. Volatile memory **30***b* includes memory types such as random-access memory (RAM), and is normally the primary operating memory into which the operating system **35**, applications **36**, program modules **37**, and application data **38** are loaded for execution by processors **20**. Volatile memory **30***b* is generally faster than non-volatile memory **30***a*

due to its electrical characteristics and is directly accessible to processors **20** for processing of instructions and data storage and retrieval. Volatile memory **30***b* may comprise one or more smaller cache memories which operate at a higher clock speed and are typically placed on the same IC as the processors to improve performance.

[0315] There are several types of computer memory, each with its own characteristics and use cases. System memory **30** may be configured in one or more of the several types described herein, including high bandwidth memory (HBM) and advanced packaging technologies like chip-on-wafer-on-substrate (CoWoS). Static random access memory (SRAM) provides fast, low-latency memory used for cache memory in processors, but is more expensive and consumes more power compared to dynamic random access memory (DRAM). SRAM retains data as long as power is supplied. DRAM is the main memory in most computer systems and is slower than SRAM but cheaper and more dense. DRAM requires periodic refresh to retain data. NAND flash is a type of non-volatile memory used for storage in solid state drives (SSDs) and mobile devices and provides high density and lower cost per bit compared to DRAM with the trade-off of slower write speeds and limited write endurance. HBM is an emerging memory technology that provides high bandwidth and low power consumption which stacks multiple DRAM dies vertically, connected by through-silicon vias (TSVs). HBM offers much higher bandwidth (up to 1 TB/s) compared to traditional DRAM and may be used in high-performance graphics cards, AI accelerators, and edge computing devices. Advanced packaging and CoWoS are technologies that enable the integration of multiple chips or dies into a single package. CoWoS is a 2.5D packaging technology that interconnects multiple dies side-by-side on a silicon interposer and allows for higher bandwidth, lower latency, and reduced power consumption compared to traditional PCB-based packaging. This technology enables the integration of heterogeneous dies (e.g., CPU, GPU, HBM) in a single package and may be used in high-performance computing, AI accelerators, and edge computing devices.

[0316] Interfaces **40** may include, but are not limited to, storage media interfaces **41**, network interfaces **42**, display interfaces **43**, and input/output interfaces **44**. Storage media interface **41** provides the necessary hardware interface for loading data from non-volatile data storage devices **50** into system memory **30** and storage data from system memory **30** to non-volatile data storage device **50**. Network interface **42** provides the necessary hardware interface for computing device **10** to communicate with remote computing devices **80** and cloud-based services **90** via one or more external communication devices **70**. Display interface **43** allows for connection of displays **61**, monitors, touchscreens, and other visual input/output devices. Display interface **43** may include a graphics card for processing graphics-intensive calculations and for handling demanding display requirements. Typically, a graphics card includes a graphics processing unit (GPU) and video RAM (VRAM) to accelerate display of graphics. In some high-performance computing systems, multiple GPUs may be connected using NVLink bridges, which provide high-bandwidth, low-latency interconnects between GPUs. NVLink bridges enable faster data transfer between GPUs, allowing for more efficient parallel processing and improved performance in applications such as machine learning, scientific simulations, and graphics rendering. One or more input/output (I/O) interfaces **44** provide the necessary support for communications between computing device **10** and any external peripherals and accessories **60**. For wireless communications, the necessary radio-frequency hardware and firmware may be connected to I/O interface **44** or may be integrated into I/O interface **44**. Network interface **42** may support various communication standards and protocols, such as Ethernet and Small Form-Factor Pluggable (SFP). Ethernet is a widely used wired networking technology that enables local area network (LAN) communication. Ethernet interfaces typically usc RJ45 connectors and support data rates ranging from 10 Mbps to 100 Gbps, with common speeds being 100 Mbps, 1 Gbps, 10 Gbps, 25 Gbps, 40 Gbps, and 100 Gbps. Ethernet is known for its reliability, low latency, and cost-effectiveness, making it a popular choice for home, office, and data center networks. SFP is a compact, hot-pluggable transceiver used for

both telecommunication and data communications applications. SFP interfaces provide a modular and flexible solution for connecting network devices, such as switches and routers, to fiber optic or copper networking cables. SFP transceivers support various data rates, ranging from 100 Mbps to 100 Gbps, and can be easily replaced or upgraded without the need to replace the entire network interface card. This modularity allows for network scalability and adaptability to different network requirements and fiber types, such as single-mode or multi-mode fiber.

[0317] Non-volatile data storage devices **50** are typically used for long-term storage of data. Data on non-volatile data storage devices **50** is not erased when power to the non-volatile data storage devices **50** is removed. Non-volatile data storage devices **50** may be implemented using any technology for non-volatile storage of content including, but not limited to, CD-ROM drives, digital versatile discs (DVD), or other optical disc storage; magnetic cassettes, magnetic tape, magnetic disc storage, or other magnetic storage devices; solid state memory technologies such as EEPROM or flash memory; or other memory technology or any other medium which can be used to store data without requiring power to retain the data after it is written. Non-volatile data storage devices **50** may be non-removable from computing device **10** as in the case of internal hard drives, removable from computing device **10** as in the case of external USB hard drives, or a combination thereof, but computing device will typically comprise one or more internal, non-removable hard drives using either magnetic disc or solid state memory technology. Non-volatile data storage devices **50** may be implemented using various technologies, including hard disk drives (HDDs) and solid-state drives (SSDs). HDDs use spinning magnetic platters and read/write heads to store and retrieve data, while SSDs use NAND flash memory. SSDs offer faster read/write speeds, lower latency, and better durability due to the lack of moving parts, while HDDs typically provide higher storage capacities and lower cost per gigabyte. NAND flash memory comes in different types, such as Single-Level Cell (SLC), Multi-Level Cell (MLC), Triple-Level Cell (TLC), and Quad-Level Cell (QLC), each with trade-offs between performance, endurance, and cost. Storage devices connect to the computing device **10** through various interfaces, such as SATA, NVMe, and PCIe. SATA is the traditional interface for HDDs and SATA SSDs, while NVMe (Non-Volatile Memory Express) is a newer, high-performance protocol designed for SSDs connected via PCIe. PCIe SSDs offer the highest performance due to the direct connection to the PCIe bus, bypassing the limitations of the SATA interface. Other storage form factors include M.2 SSDs, which are compact storage devices that connect directly to the motherboard using the M.2 slot, supporting both SATA and NVMe interfaces. Additionally, technologies like Intel Optane memory combine 3D XPoint technology with NAND flash to provide high-performance storage and caching solutions. Non-volatile data storage devices **50** may be non-removable from computing device **10**, as in the case of internal hard drives, removable from computing device **10**, as in the case of external USB hard drives, or a combination thereof. However, computing devices will typically comprise one or more internal, non-removable hard drives using either magnetic disc or solid-state memory technology. Non-volatile data storage devices **50** may store any type of data including, but not limited to, an operating system **51** for providing low-level and mid-level functionality of computing device **10**, applications **52** for providing high-level functionality of computing device **10**, program modules **53** such as containerized programs or applications, or other modular content or modular programming, application data **54**, and databases **55** such as relational databases, non-relational databases, object oriented databases, NoSQL databases, vector databases, knowledge graph databases, key-value databases, document oriented data stores, and graph databases.

[0318] Applications (also known as computer software or software applications) are sets of programming instructions designed to perform specific tasks or provide specific functionality on a computer or other computing devices. Applications are typically written in high-level programming languages such as C, C++, Scala, Erlang, GoLang, Java, Scala, Rust, and Python, which are then either interpreted at runtime or compiled into low-level, binary, processor-executable instructions operable on processors **20**. Applications may be containerized so that they can be run on any

computer hardware running any known operating system. Containerization of computer software is a method of packaging and deploying applications along with their operating system dependencies into self-contained, isolated units known as containers. Containers provide a lightweight and consistent runtime environment that allows applications to run reliably across different computing environments, such as development, testing, and production systems facilitated by specifications such as containerd.

[0319] The memories and non-volatile data storage devices described herein do not include communication media. Communication media are means of transmission of information such as modulated electromagnetic waves or modulated data signals configured to transmit, not store, information. By way of example, and not limitation, communication media includes wired communications such as sound signals transmitted to a speaker via a speaker wire, and wireless communications such as acoustic waves, radio frequency (RF) transmissions, infrared emissions, and other wireless media.

[0320] External communication devices **70** are devices that facilitate communications between computing devices and either remote computing devices **80**, or cloud-based services **90**, or both. External communication devices **70** include, but are not limited to, data modems **71** which facilitate data transmission between computing device and the Internet **75** via a common carrier such as a telephone company or internet service provider (ISP), routers **72** which facilitate data transmission between computing device and other devices, and switches **73** which provide direct data communications between devices on a network or optical transmitters (e.g., lasers). Here, modem **71** is shown connecting computing device **10** to both remote computing devices **80** and cloud-based services **90** via the Internet **75**. While modem **71**, router **72**, and switch **73** are shown here as being connected to network interface **42**, many different network configurations using external communication devices **70** are possible. Using external communication devices **70**, networks may be configured as local area networks (LANs) for a single location, building, or campus, wide area networks (WANs) comprising data networks that extend over a larger geographical area, and virtual private networks (VPNs) which can be of any size but connect computers via encrypted communications over public networks such as the Internet **75**. As just one exemplary network configuration, network interface **42** may be connected to switch **73** which is connected to router **72** which is connected to modem **71** which provides access for computing device **10** to the Internet **75**. Further, any combination of wired **77** or wireless **76** communications between and among computing devices **10**, external communication devices **70**, remote computing devices **80**, and cloud-based services **90** may be used. Remote computing devices **80**, for example, may communicate with computing devices through a variety of communication channels **74** such as through switch **73** via a wired **77** connection, through router **72** via a wireless connection **76**, or through modem **71** via the Internet **75**. Furthermore, while not shown here, other hardware that is specifically designed for servers or networking functions may be employed. For example, secure socket layer (SSL) acceleration cards can be used to offload SSL encryption computations, and transmission control protocol/internet protocol (TCP/IP) offload hardware and/or packet classifiers on network interfaces **42** may be installed and used at server devices or intermediate networking equipment (e.g., for deep packet inspection).

[0321] In a networked environment, certain components of computing device **10** may be fully or partially implemented on remote computing devices **80** or cloud-based services **90**. Data stored in non-volatile data storage device **50** may be received from, shared with, duplicated on, or offloaded to a non-volatile data storage device on one or more remote computing devices **80** or in a cloud computing service **92**. Processing by processors **20** may be received from, shared with, duplicated on, or offloaded to processors of one or more remote computing devices **80** or in a distributed computing service **93**. By way of example, data may reside on a cloud computing service **92**, but may be usable or otherwise accessible for use by computing device **10**. Also, certain processing subtasks may be sent to a microservice **91** for processing with the result being transmitted to

computing device **10** for incorporation into a larger processing task. Also, while components and processes of the exemplary computing environment are illustrated herein as discrete units (e.g., OS **51** being stored on non-volatile data storage device **51** and loaded into system memory **35** for use) such processes and components may reside or be processed at various times in different components of computing device **10**, remote computing devices **80**, and/or cloud-based services **90**. Also, certain processing subtasks may be sent to a microservice **91** for processing with the result being transmitted to computing device **10** for incorporation into a larger processing task. Infrastructure as Code (IaaC) tools like Terraform can be used to manage and provision computing resources across multiple cloud providers or hyperscalers. This allows for workload balancing based on factors such as cost, performance, and availability. For example, Terraform can be used to automatically provision and scale resources on AWS spot instances during periods of high demand, such as for surge rendering tasks, to take advantage of lower costs while maintaining the required performance levels. In the context of rendering, tools like Blender can be used for object rendering of specific elements, such as a car, bike, or house. These elements can be approximated and roughed in using techniques like bounding box approximation or low-poly modeling to reduce the computational resources required for initial rendering passes. The rendered elements can then be integrated into the larger scene or environment as needed, with the option to replace the approximated elements with higher-fidelity models as the rendering process progresses.

[0322] In an implementation, the disclosed systems and methods may utilize, at least in part, containerization techniques to execute one or more processes and/or steps disclosed herein. Containerization is a lightweight and efficient virtualization technique that allows you to package and run applications and their dependencies in isolated environments called containers. One of the most popular containerization platforms is containerd, which is widely used in software development and deployment. Containerization, particularly with open-source technologies like containerd and container orchestration systems like Kubernetes, is a common approach for deploying and managing applications. Containers are created from images, which are lightweight, standalone, and executable packages that include application code, libraries, dependencies, and runtime. Images are often built from a containerfile or similar, which contains instructions for assembling the image. Containerfiles are configuration files that specify how to build a container image. Systems like Kubernetes natively support containerd as a container runtime. They include commands for installing dependencies, copying files, setting environment variables, and defining runtime configurations. Container images can be stored in repositories, which can be public or private. Organizations often set up private registries for security and version control using tools such as Harbor, JFrog Artifactory and Bintray, GitLab Container Registry, or other container registries. Containers can communicate with each other and the external world through networking. Containerd provides a default network namespace, but can be used with custom network plugins. Containers within the same network can communicate using container names or IP addresses.

[0323] Remote computing devices **80** are any computing devices not part of computing device **10**. Remote computing devices **80** include, but are not limited to, personal computers, server computers, thin clients, thick clients, personal digital assistants (PDAs), mobile telephones, watches, tablet computers, laptop computers, multiprocessor systems, microprocessor based systems, set-top boxes, programmable consumer electronics, video game machines, game consoles, portable or handheld gaming units, network terminals, desktop personal computers (PCs), minicomputers, mainframe computers, network nodes, virtual reality or augmented reality devices and wearables, and distributed or multi-processing computing environments. While remote computing devices **80** are shown for clarity as being separate from cloud-based services **90**, cloud-based services **90** are implemented on collections of networked remote computing devices **80**.

[0324] Cloud-based services **90** are Internet-accessible services implemented on collections of networked remote computing devices **80**. Cloud-based services are typically accessed via application programming interfaces (APIs) which are software interfaces which provide access to

computing services within the cloud-based service via API calls, which are pre-defined protocols for requesting a computing service and receiving the results of that computing service. While cloud-based services may comprise any type of computer processing or storage, three common categories of cloud-based services **90** are serverless logic apps, microservices **91**, cloud computing services **92**, and distributed computing services **93**.

[0325] Microservices **91** are collections of small, loosely coupled, and independently deployable computing services. Each microservice represents a specific computing functionality and runs as a separate process or container. Microservices promote the decomposition of complex applications into smaller, manageable services that can be developed, deployed, and scaled independently. These services communicate with each other through well-defined application programming interfaces (APIs), typically using lightweight protocols like HTTP, protobuffers, gRPC or message queues such as Kafka. Microservices **91** can be combined to perform more complex or distributed processing tasks. In an embodiment, Kubernetes clusters with containerized resources are used for operational packaging of system.

[0326] Cloud computing services **92** are delivery of computing resources and services over the Internet **75** from a remote location. Cloud computing services **92** provide additional computer hardware and storage on as-needed or subscription basis. Cloud computing services **92** can provide large amounts of scalable data storage, access to sophisticated software and powerful server-based processing, or entire computing infrastructures and platforms. For example, cloud computing services can provide virtualized computing resources such as virtual machines, storage, and networks, platforms for developing, running, and managing applications without the complexity of infrastructure management, and complete software applications over public or private networks or the Internet on a subscription or alternative licensing basis, or consumption or ad-hoc marketplace basis, or combination thereof.

[0327] Distributed computing services **93** provide large-scale processing using multiple interconnected computers or nodes to solve computational problems or perform tasks collectively. In distributed computing, the processing and storage capabilities of multiple machines are leveraged to work together as a unified system. Distributed computing services are designed to address problems that cannot be efficiently solved by a single computer or that require large-scale computational power or support for highly dynamic compute, transport or storage resource variance or uncertainty over time requiring scaling up and down of constituent system resources. These services enable parallel processing, fault tolerance, and scalability by distributing tasks across multiple nodes.

[0328] While the disclosed inventions core federated distributed computational architecture enhancements targets CRISPR-based and bridge RNA based multi-omics and biological research, the disclosure's modular design supports a broader range of scientific and engineering domains in particular across proteins, computational chemistry, biology and multi-omics. Specifically, the federation manager's scheduling and ephemeral enclaving capabilities extend seamlessly to other highly regulated scenarios—e.g., clinical data analytics for patient-reported outcomes, pathology and imaging sharing, or real-time financial risk modeling that demands isolation of sensitive data. By abstracting the knowledge integration subsystem to handle domain-specific ontologies (not limited to biological terminologies), the invention's secure graph foundation can be repurposed for medical imaging, advanced industrial process monitoring, or multi-center robotics development.

[0329] The system's dynamic resource allocation accommodates HPC and XaaS (e.g. GPU and TPU) bursts in diverse verticals such as proteomics, physical phenomena modeling (e.g. electric, magnetic, fluid, structure, plastic, FSI), or materials science and engineering simulations. Load balancing logic within the resource tracking subsystem automatically scales to handle surges in data volume or complex multi-temporal analyses (like daily, monthly, annual climate patterns). Similarly, ephemeral enclaves, blind execution protocols, and homomorphic encryption modules all remain applicable for any domain requiring fine-grained privacy, from healthcare compliance to

trade-secret-focused industrial R&D.
[0330] Further expansions exploit multi-cloud or hybrid HPC arrangements. Institutions can register ephemeral GPU-backed cloud instances through Terraform or Kubernetes orchestration. The federation manager treats these instances as ephemeral nodes that join the graph only when demand spikes, further optimizing cost and efficiency. Meanwhile, the knowledge integration subsystem updates cross-cloud knowledge graphs, ensuring distributed, real-time data integrity. The net effect is a scalable, domain-agnostic infrastructure, enabling any consortium of research or commercial partners to accelerate innovation without risking data exposure.
[0331] The invention's architecture thus excels as a universal, future-proof approach for secure and distributed computation, uniting advanced CRISPR analytics with many other fields. By combining multi-temporal modeling, ephemeral enclaves, HPC-lab automation, and dynamic resource scheduling, the system provides a blueprint for next-generation collaborative data science—ensuring confidentiality, adaptability, and high-throughput performance.
[0332] Although described above as a physical device, computing device **10** can be a virtual computing device, in which case the functionality of the physical components herein described, such as processors **20**, system memory **30**, network interfaces **40**, NVLink or other GPU-to-GPU high bandwidth communications links and other like components can be provided by computer-executable instructions. Such computer-executable instructions can execute on a single physical computing device, or can be distributed across multiple physical computing devices, including being distributed across multiple physical computing devices in a dynamic manner such that the specific, physical computing devices hosting such computer-executable instructions can dynamically change over time depending upon need and availability. In the situation where computing device **10** is a virtualized device, the underlying physical computing devices hosting such a virtualized computing device can, themselves, comprise physical components analogous to those described above, and operating in a like manner. Furthermore, virtual computing devices can be utilized in multiple layers with one virtual computing device executing within the construct of another virtual computing device. Thus, computing device **10** may be either a physical computing device or a virtualized computing device within which computer-executable instructions can be executed in a manner consistent with their execution by a physical computing device. Similarly, terms referring to physical components of the computing device, as utilized herein, mean either those physical components or virtualizations thereof performing the same or equivalent functions.
[0333] The skilled person will be aware of a range of possible modifications of the various aspects described above. Accordingly, the present invention is defined by the claims and their equivalents.

Claims

1. A federated distributed computational system comprising: a plurality of computational nodes distributed across multiple institutions; and a federation manager coupled to the plurality of computational nodes and configured to enforce institutional governance protocols, wherein each computational node comprises: a local computational engine configured to process biological data across multiple temporal and spatial scales; a privacy preservation subsystem implementing multi-layer security protocols including blind execution protocols and ephemeral enclaves; a knowledge integration component configured to orchestrate multiple specialized databases including relational, NoSQL, time-series, columnar, and vector databases while maintaining cross-institutional privacy boundaries; and a communication interface configured to enable secure cross-institutional data exchange; wherein the federation manager coordinates real-time distributed computation across the plurality of nodes while maintaining data privacy between institutions and dynamically adapting resource allocation based on computational demands.
2. The system of claim 1, wherein the local computational engine comprises: a distributed computational graph processor configured to perform multi-scale analysis across molecular,

cellular, tissue, and organisms levels; a resource optimization module that dynamically allocates computational resources across multiple time domains from milliseconds to weeks; and a real-time monitoring system that enables adaptive feedback across different biological scales.

3. The system of claim 1, wherein the privacy preservation subsystem comprises: blind execution protocols that enable collaborative computation while maintaining node privacy; ephemeral enclaves that provide temporary, isolated computational environments for sensitive operations; differential privacy mechanisms for secure data aggregation; and federated learning protocols that ensure raw data never leaves local custody.

4. The system of claim 1, wherein the knowledge integration component comprises: a distributed knowledge graph implementing spatio-temporal and event-based relationships; a vector database configured for high-dimensional biological data storage and retrieval; neurosymbolic reasoning capabilities combining logical constraints with machine learning inference; and provenance tracking systems that maintain data lineage across federated operations.

5. The system of claim 1, wherein the federation manager comprises: a synthetic data generation module implementing copula-based transferable models; probabilistic programming frameworks for complex generative processes; privacy-preserving validation layers for synthetic data quality assessment; and adaptive optimization mechanisms for cross-domain knowledge transfer.

6. The system of claim 1, further comprising a multi-temporal modeling framework configured to: analyze biological data across multiple time scales simultaneously; enable dynamic feedback incorporation from real-time experimental results; coordinate data ingestion and monitoring across different temporal resolutions; and reallocate computational resources based on temporal analysis requirements.

7. The system of claim 1, wherein each computational node comprises a genome-scale editing module configured to: coordinate multi-locus editing operations with real-time validation; implement privacy-preserving protocols for sensitive genomic data; maintain audit trails of editing operations while preserving institutional boundaries; and enable secure collaborative validation of editing outcomes.

8. A method for federated distributed computation comprising: establishing a plurality of computational nodes distributed across multiple institutions; implementing a federation manager coupled to the plurality of nodes and configured to enforce institutional governance protocols; at each computational node: processing biological data using a local computational engine configured for multi-scale analysis; preserving data privacy through multi-layer security protocols including blind execution and ephemeral enclaves; integrating knowledge components across multiple specialized database types while maintaining institutional boundaries; maintaining secure cross-institutional communications; coordinating real-time distributed computation across the plurality of nodes while maintaining data privacy between institutions; and dynamically adapting resource allocation based on computational demands.

9. The method of claim 8, wherein processing biological data comprises: implementing a distributed computational graph for integrated multi-scale analysis; performing dynamic resource optimization across multiple time domains; and enabling adaptive feedback across different biological scales.

10. The method of claim 8, wherein preserving data privacy comprises: executing blind protocols that enable collaborative computation; implementing ephemeral enclaves for sensitive operations; applying differential privacy mechanisms for data aggregations; and utilizing federated learning protocols to maintain local data custody.

11. The method of claim 8, wherein integrating knowledge components comprises: maintaining a distributed knowledge graph with spatio-temporal relationships; implementing vector storage for high-dimensional biological data; enabling neurosymbolic reasoning capabilities; and tracking data provenance across federated operations.

12. The method of claim 8, wherein the federation manager generates synthetic data by:

implementing copula-based transferable models; utilizing probabilistic programming frameworks; validating synthetic data quality while preserving privacy; and optimizing cross-domain knowledge transfer mechanisms.

13. The method of claim 8, further comprising: analyzing biological data through multi-temporal modeling; incorporating dynamic feedback from real-time results; coordinating data ingestion across temporal scales; and adaptively reallocating computational resources.

14. The method of claim 8, further comprising: coordinating genome-scale editing operations with real-time validation; implementing privacy-preserving genomic data protocols; maintaining secure audit trails across institutional boundaries; and enabling collaborative validation of editing outcomes.