Security-Native Interference Sandbox

Semantic Isolation and Intrusion-Tolerant Execution in QVM–QFM

Honza Rožek
QVM / QFM Architecture

Abstract

This document specifies a security-native sandboxing framework for the Quantum Virtual Machine (QVM) operating over Quansistor Field Mathematics (QFM). The framework treats security not as an external control mechanism but as a semantic property of operator interaction, interference, and state evolution.

Instead of isolating computation through runtime boundaries or trusted environments, the proposed model enforces security by constraining how interference may propagate between execution zones. Security violations manifest as detectable semantic anomalies rather than silent breaches.

The framework enables interference-safe execution zones, blast-radius containment, cryptographically sealed state evolution, and intrusion-tolerant collapse, providing replayable and auditable security guarantees for distributed computation.

Contents

1 Motivation: Why Security Must Be Semantic

1.1 Limits of Perimeter-Based Security

Conventional security models treat computation as inherently unsafe and attempt to protect it through external controls such as access lists, runtime sandboxes, trusted execution environments, or network perimeters.

These mechanisms operate outside the semantic model of computation. They regulate who may execute code or access resources, but they do not constrain how computation itself propagates influence through shared state, dependencies, or indirect effects.

As systems become distributed, adaptive, and operator-driven, such perimeter-based security models become increasingly fragile.

1.2 Runtime Isolation Is Not Semantic Isolation

Runtime sandboxing isolates execution environments at the level of processes, memory regions, or containers. While effective against some classes of attacks, runtime isolation does not prevent semantic interaction through shared state, implicit coupling, or indirect dependencies.

Two computations may be runtime-isolated yet semantically entangled if their effects interfere through shared architectural structures. Conversely, runtime co-location does not imply semantic risk if operations are meaningfully independent.

Security that does not account for semantic interaction leaves critical attack surfaces unmodeled.

1.3 Trusted Environments and the Trust Assumption

Trusted execution environments and enclaves attempt to secure execution by narrowing the trusted computing base. However, these approaches require strong assumptions about hardware correctness, implementation fidelity, and administrative trust.

Such assumptions are difficult to justify in distributed and multi-domain deployments. Moreover, trust-based security provides limited forensic value: when a violation occurs, it is often unclear whether the failure was semantic, operational, or infrastructural.

1.4 Interference as the True Attack Surface

In QVM–QFM, the fundamental attack surface is interference: the ability of one operation to influence another through semantic coupling of state.

Attacks manifest not only as unauthorized access but as unintended propagation of influence across execution boundaries. This includes:

- leakage of information through shared constraints;
- manipulation of outcomes via indirect dependencies;

- amplification of effects through uncontrolled coupling.

Any security model that does not explicitly constrain interference cannot provide complete protection.

1.5 Security as a Property of Execution Semantics

To address these limitations, security must be elevated from a runtime control problem to a semantic property of execution.

In a semantic security model:

- permitted interactions are defined explicitly;
- interference paths are constrained by design;
- violations manifest as detectable semantic anomalies;
- security guarantees are replayable and auditable.

This approach aligns security with the same principles that govern determinism, causality, and cost accounting in QVM–QFM.

1.6 Design Position of This Work

This work introduces a security-native interference sandbox for QVM–QFM, in which security zones are defined by semantic interference constraints rather than by runtime boundaries.

Instead of attempting to prevent all intrusion, the framework ensures that intrusion effects are bounded, visible, and reconstructible.

Security becomes a property of computation itself, not an external assumption imposed upon it.

2 Interference as an Attack Surface

2.1 From Unauthorized Access to Unauthorized Influence

Traditional security models focus on preventing unauthorized access to code, memory, or data. In distributed and operator-based systems, this focus is insufficient.

An attacker may comply fully with access controls while still exerting unauthorized influence over computation outcomes through semantic interference. Such influence does not require direct access to protected resources.

Security failures therefore manifest not only as access violations but as violations of intended influence boundaries.

2.2 Semantic Interference Channels

A semantic interference channel exists whenever the behavior or outcome of one computation depends on the state or evolution of another in a way that is not explicitly declared or constrained.

Examples include:

- shared normalization or stabilization constraints;
- indirect coupling through global invariants;
- feedback effects introduced by adaptive operators;
- accumulation of interference prior to collapse.

These channels are invisible to runtime isolation mechanisms and are not addressed by access control lists.

2.3 Interference Amplification

Small, permitted interactions may be amplified through repeated interference. An attacker may exploit such amplification to bias results, leak information, or force premature collapse without ever violating explicit permissions.

Because amplification is semantic rather than operational, it may remain undetected until final outcomes diverge.

2.4 Distributed Interference Propagation

In distributed execution, interference may propagate across canister boundaries and administrative domains.

Latency, replication, and concurrency increase the difficulty of detecting such propagation using runtime monitoring alone. Identical attack patterns may produce different physical traces while inducing the same semantic effect.

This further demonstrates that security must be enforced at the level of semantics rather than execution artifacts.

2.5 Why Runtime Monitoring Is Insufficient

Runtime monitoring attempts to detect anomalies by observing execution behavior. However, interference attacks may follow entirely valid execution paths while producing invalid semantic influence.

Without an explicit model of permitted interference, monitoring cannot distinguish legitimate coupling from malicious manipulation.

Detection must therefore be grounded in declared semantic boundaries.

2.6 Interference as a First-Class Security Concept

By treating interference as a first-class security concept, QVM–QFM reframes security analysis from access control to influence control.

The question shifts from:

Who executed what?

to:

Which operations were allowed to influence which others, and how?

This reframing enables precise specification, enforcement, and forensic analysis of security boundaries.

2.7 Summary

Interference constitutes the primary attack surface in operator-based distributed systems. Attacks exploit unauthorized influence rather than unauthorized access. Effective security therefore requires explicit modeling and control of interference paths within execution semantics.

3 Security-Native Sandbox Zones

3.1 Sandbox Zones as Semantic Constructs

In QVM–QFM, a sandbox zone is not defined by process boundaries, memory isolation, or execution context. Instead, a sandbox zone is a semantic construct that constrains how interference may propagate between operators and state components.

A sandbox zone specifies which forms of influence are permitted, which are forbidden, and how violations are detected and handled.

Sandboxing is therefore expressed in terms of *interference permissions*, not runtime containment.

3.2 Definition of a Sandbox Zone

Formally, a sandbox zone Z is defined as a tuple:

$$Z = (\Psi_Z, \mathcal{O}_Z, \mathcal{I}_Z),$$

where:

- Ψ_Z is the subset of architectural state governed by the zone;
- $\mathcal{O}_Z$ is the set of operators permitted to act within the zone;
- $\mathcal{I}_Z$ defines the allowed interference relations between Ψ_Z and external state.

Any operator application violating $\mathcal{I}_Z$ constitutes a semantic security violation.

3.3 Explicit Interference Permissions

Interference permissions are explicit and declarative. A sandbox zone must specify:

- which external state components may influence Ψ_Z;
- whether influence is read-only, write-only, or bidirectional;
- whether influence is reversible or may trigger collapse;
- any quantitative limits on interference strength.

Implicit or inferred interference permissions are forbidden.

3.4 Isolation by Interference Constraints

Isolation in QVM–QFM is achieved by constraining interference, not by preventing execution.

Two computations may execute concurrently and even share physical resources while remaining securely isolated if their interference relations are disjoint or explicitly bounded.

Conversely, two computations may be runtime-isolated yet considered non-isolated if their semantic interference is unconstrained.

3.5 Nested and Composable Sandbox Zones

Sandbox zones may be nested and composed. A zone may contain sub-zones with stricter interference constraints.

Interference permissions compose monotonically: a sub-zone may further restrict interference but may not weaken the constraints of its parent zone.

This enables fine-grained security policies without global coordination.

3.6 Dynamic Zone Creation and Modification

Sandbox zones may be created or modified dynamically through explicit zone-management operators.

Such operators:

- declare the scope of the zone;
- define interference permissions;
- specify default violation handling behavior.

Zone modification itself is subject to replay, audit, and security policy enforcement.

3.7 Detection of Sandbox Violations

A sandbox violation occurs when an operator induces interference not permitted by the zone's interference specification.

Violations are detected semantically during operator application rather than retrospectively through monitoring.

Detection does not rely on anomaly heuristics or pattern recognition; it is a direct consequence of semantic mismatch.

3.8 Visibility and Forensic Traceability

Sandbox zone boundaries, permissions, and violations are recorded in the execution trace.

During replay or forensic analysis, a verifier can reconstruct:

- the active sandbox zones at each logical time;
- permitted interference relations;
- the exact point and nature of any violation.

This ensures that security decisions are reproducible and legally meaningful.

3.9 Summary

Security-native sandbox zones in QVM–QFM enforce isolation by constraining semantic interference. By defining sandboxing as a property of operator interaction rather than runtime containment, the framework provides precise, replayable, and auditable security boundaries for distributed computation.

4 Blast-Radius Containment and Isolation

4.1 From Binary Security to Bounded Impact

Conventional security models implicitly assume a binary outcome: either an attack is prevented, or the system is compromised. In complex distributed systems, this assumption is unrealistic.

QVM–QFM adopts a different position: intrusion may occur, but its impact must be bounded, visible, and semantically contained. Security is therefore defined not by absolute prevention, but by controlled blast radius.

4.2 Definition of Blast Radius

The blast radius of an operation or intrusion is defined as the maximal set of architectural state components that may be semantically influenced as a consequence of that operation.

Formally, for an operator application $\mathcal{O}$ within a sandbox zone Z, the blast radius $B(\mathcal{O}, Z)$ is the transitive closure of permitted interference relations originating from Ψ_Z.

Any influence outside this set constitutes a sandbox violation.

4.3 Blast-Radius Constraints

Blast-radius constraints are declared as part of sandbox zone specification. Such constraints may include:

- explicit limits on which state components may be affected;
- quantitative bounds on interference strength;
- prohibitions on cross-zone propagation;
- requirements for explicit collapse before propagation.

These constraints are semantic and independent of runtime topology or resource allocation.

4.4 Containment of Malicious or Faulty Behavior

If an operator behaves maliciously or incorrectly within a sandbox zone, its effects are confined to the blast radius defined for that zone.

Even in the presence of adversarial behavior, the framework guarantees that:

- no undeclared state outside the blast radius is affected;
- no hidden interference paths are introduced;
- all effects remain visible and replayable.

This containment applies equally to accidental faults and intentional attacks.

4.5 Isolation Across Zones

Sandbox zones may be configured to enforce strict blast-radius isolation. In such configurations, interference between zones is prohibited unless explicitly mediated through boundary operators.

Boundary operators define:

- which effects may cross zone boundaries;
- how such effects are transformed or attenuated;
- whether crossing requires explicit collapse or authorization.

Absent such operators, blast radius does not extend beyond zone boundaries.

4.6 Dynamic Adjustment of Blast Radius

Blast-radius constraints may be adjusted dynamically through explicit policy operators. Such adjustments:

- are themselves subject to sandbox permissions;
- are recorded in the execution trace;
- take effect at a specific logical time.

This enables adaptive security policies without sacrificing determinism or auditability.

4.7 Forensic Analysis of Blast Radius

During replay or forensic analysis, a verifier can reconstruct the blast radius associated with any operation or intrusion.

This enables precise answers to questions such as:

- which state components were at risk;
- which components were actually affected;
- whether any effect exceeded declared limits.

Such analysis is essential for legal accountability and regulatory review.

4.8 Summary

Blast-radius containment in QVM–QFM ensures that security incidents have bounded, explicit, and reconstructible impact. By defining isolation in terms of semantic influence rather than runtime barriers, the framework provides intrusion tolerance without sacrificing transparency or determinism.

5 Cryptographic Sealing of State Evolution

5.1 Purpose of State Sealing

While sandbox zones and blast-radius constraints limit semantic interference, cryptographic sealing ensures that state evolution itself cannot be modified, reordered, or selectively concealed without detection.

State sealing does not protect secrecy by default. Its primary purpose is integrity, continuity, and forensic verifiability of execution history.

Sealing binds semantic evolution to cryptographic evidence.

5.2 Sealed State Transitions

Each committed state transition

$$\Psi_t \to \Psi_{t+1}$$

is cryptographically sealed by computing a seal value σ_{t+1} derived from:

- the prior seal σ_t;
- the operator identifier and parameters;
- the affected state scope;
- the resulting state hash.

Formally, a seal is defined as:

$$\sigma_{t+1} = H(\sigma_t \,\|\, \mathcal{O}_t \,\|\, \text{scope}_t \,\|\, \text{hash}(\Psi_{t+1})),$$

where H is a cryptographic hash function.

5.3 Chain of Semantic Integrity

Seals form a hash-linked chain aligned with logical time. Any attempt to alter, remove, reorder, or inject state transitions invalidates all subsequent seals.

Because seals are computed from semantic descriptors rather than runtime artifacts, integrity verification is independent of execution environment or deployment topology.

5.4 Zone-Aware Sealing

Sandbox zones may define independent sealing domains. In such configurations:

- each zone maintains its own seal chain;
- boundary operators introduce explicit seal linkage between zones;
- cross-zone propagation is cryptographically visible.

This enables compartmentalized verification while preserving global auditability.

5.5 Sealing of Violations and Intrusions

Sandbox violations, policy breaches, and intrusion-related events are sealed explicitly as part of state evolution.

An intrusion cannot be concealed by rollback, partial disclosure, or selective replay. The cryptographic seal guarantees that evidence of the event persists in the execution history.

5.6 Replay and Independent Verification

During deterministic replay, seal values are recomputed and compared against recorded seals.

Verification requires no trusted runtime, hardware enclave, or network attestation. Any party possessing the execution trace can independently verify integrity and ordering.

5.7 Relationship to Deterministic Replay

State sealing complements deterministic replay by providing cryptographic evidence that the replayed execution corresponds exactly to the original.

Replay establishes semantic equivalence; sealing establishes integrity and non-repudiation of execution history.

5.8 Failure Semantics

Failure to compute or verify a seal constitutes a fatal integrity error. Execution must halt or transition into a failure-handling state defined by security policy.

Sealing failures cannot be ignored, deferred, or repaired heuristically.

5.9 Summary

Cryptographic sealing binds semantic state evolution to verifiable integrity guarantees. By sealing operator application, scope, and resulting state, QVM–QFM ensures that security-relevant execution history is immutable, replayable, and forensically trustworthy.

6 Intrusion-Tolerant Collapse and Recovery

6.1 From Intrusion Prevention to Intrusion Tolerance

Absolute intrusion prevention is not achievable in complex distributed systems. QVM–QFM therefore adopts an intrusion-tolerant security model in which intrusions are expected, detected semantically, and handled through controlled state transitions.

Security is defined not by the impossibility of intrusion, but by the ability to contain, record, and recover from it without loss of determinism or accountability.

6.2 Definition of Intrusion Events

An intrusion event is defined as any operator application or state transition that violates declared sandbox zone constraints, interference permissions, or blast-radius limits.

Intrusion events are semantic violations, not behavioral anomalies. They are detected deterministically during operator application rather than through post hoc monitoring.

6.3 Intrusion-Triggered Collapse

Upon detection of an intrusion event, the execution environment may invoke an intrusion-triggered collapse.

An intrusion-triggered collapse is a deterministic operator that:

- terminates or isolates the affected execution context;
- resolves accumulated interference within declared limits;
- commits the system to a secure and stable post-intrusion state.

The collapse is explicit, replayable, and recorded in the execution trace.

6.4 Isolation and Containment During Collapse

During intrusion-triggered collapse, blast-radius constraints are enforced strictly. Effects of the intrusion cannot propagate beyond the declared sandbox zone or permitted boundaries.

Any attempt by malicious operators to amplify influence during collapse is blocked by semantic constraints rather than runtime controls.

6.5 Preservation of Evidence

Intrusion-triggered collapse preserves forensic evidence. The state prior to collapse, the violation details, and the resulting post-collapse state are all sealed cryptographically.

No rollback or erasure of intrusion evidence is permitted as part of recovery.

6.6 Deterministic Recovery Paths

Following collapse, recovery proceeds along deterministic paths defined by security policy. Possible recovery actions include:

- resumption from a clean pre-intrusion checkpoint;
- continuation within a restricted sandbox zone;
- escalation to human or external governance.

Recovery actions are explicit, replayable, and subject to the same causality and accounting rules as ordinary execution.

6.7 Replay and Auditability

During deterministic replay, intrusion-triggered collapse and recovery are reconstructed identically. A verifier can determine:

- the exact point of intrusion detection;
- the scope and nature of the violation;

- the collapse operator applied;
- the resulting system state.

This enables independent validation of security handling.

6.8 Relationship to Energy and Entropy Accounting

Intrusion-triggered collapse incurs explicit cost. Collapse energy and entropy reduction are accounted for according to the QFM accounting framework.

This prevents attackers from inducing collapse without economic or semantic consequence and aligns security response with system-wide cost controls.

6.9 Summary

Intrusion-tolerant collapse enables QVM–QFM to handle security violations deterministically and transparently. By treating intrusion as a semantic event that triggers controlled collapse and recovery, the framework provides resilience without sacrificing auditability or replayability.

7 Security Guarantees, Failure Modes, and Non-Claims

7.1 Scope of Security Guarantees

The Security-Native Interference Sandbox provides security guarantees at the level of execution semantics rather than at the level of runtime, hardware, or network infrastructure.

The primary guarantee is interference integrity: ensuring that semantic influence between operators and state components occurs only through declared and constrained interference relations.

The framework guarantees detectability, containment, and forensic reconstructibility of security-relevant events.

7.2 Guaranteed Properties

An execution conforming to this framework guarantees that:

- sandbox zone boundaries are semantically enforced;
- interference propagation is limited to declared permissions;
- blast radius of any operation or intrusion is bounded and explicit;
- state evolution integrity is cryptographically sealed;
- intrusion-triggered collapse is deterministic and replayable.

Any deviation from these properties constitutes a detectable violation of the security model.

7.3 Failure Modes

The framework explicitly tolerates certain failure modes, including:

- successful intrusion attempts within a sandbox zone;
- malicious or faulty operator behavior;
- partial execution failure or forced collapse;
- network partitions or canister unavailability.

Such failures do not invalidate the security guarantees. Instead, they are represented as explicit semantic events with bounded impact and preserved evidence.

7.4 What the Framework Does Not Guarantee

The Security-Native Interference Sandbox does not claim to:

- prevent all intrusions or attacks;
- guarantee confidentiality or data secrecy;
- ensure availability or liveness under all conditions;
- protect against denial-of-service attacks;
- enforce trustworthiness of operators or users.

The framework guarantees accountability and containment, not absolute protection.

7.5 Explicit Non-Claims

To avoid overreach, the following claims are explicitly excluded:

- no reliance on trusted execution environments or hardware enclaves;
- no use of heuristic anomaly detection or behavioral monitoring;
- no assumption of benign execution or cooperative actors;
- no claim of artificial intelligence or autonomous security;
- no claim of real-time response guarantees.

Security is derived from semantic constraints, not from trust or heuristics.

7.6 Separation of Security and Policy

The framework separates semantic security guarantees from policy decisions. While the sandbox defines what interference is permitted, policy determines how violations are handled.

This separation ensures that security semantics remain stable and auditable even as policy evolves.

7.7 Legal and Regulatory Interpretation

For legal and regulatory purposes, compliance with this framework establishes that:

- security-relevant influence paths were explicitly constrained;
- violations were detectable and recorded;
- impact of intrusion was bounded and reconstructible.

The framework therefore provides a clear evidentiary basis for attributing responsibility in security incidents.

7.8 Summary

Security in QVM–QFM arises from explicit semantic control of interference, bounded blast radius, and cryptographically sealed state evolution. Claims are intentionally limited to detectability, containment, and auditability, ensuring robustness under adversarial, legal, and regulatory scrutiny.

8 Conclusion

This document has introduced a security-native interference sandbox for the Quantum Virtual Machine (QVM) operating over Quansistor Field Mathematics (QFM). The framework redefines security as a semantic property of execution rather than as an external runtime control or trust assumption.

By treating interference as the primary attack surface, the proposed model constrains how influence may propagate between operators and state components through explicit sandbox zones, blast-radius limits, and boundary semantics. Security violations manifest as detectable semantic events rather than as silent breaches.

Cryptographic sealing of state evolution ensures integrity and non-repudiation of execution history, while intrusion-triggered collapse provides a deterministic and auditable mechanism for containment and recovery. Security incidents are therefore bounded, visible, and reconstructible without sacrificing determinism or replayability.

The framework deliberately avoids claims of absolute protection, confidentiality, or availability. Instead, it establishes accountable security through explicit semantic constraints, deterministic handling of violations, and preserved forensic evidence.

Taken together, the Security-Native Interference Sandbox provides a foundational security model for distributed computation in which intrusion tolerance, containment, and legal verifiability are inherent properties of execution semantics rather than after-the-fact controls.

A Formal Sandbox and Interference Definitions

A.1 Normative Scope

This appendix specifies formal definitions for sandbox zones, interference permissions, blast radius, and intrusion events used by the Security-Native Interference Sandbox framework. The definitions herein are normative. Any execution claiming compliance with this framework *must* satisfy these definitions.

A.2 Architectural State

Let Ψ denote the complete architectural state of a QVM–QFM execution. Ψ is composed of a finite set of typed state components governed by QFM semantics.

At logical time t, the architectural state is denoted Ψ_t.

A.3 Operators

An operator $\mathcal{O}$ is a deterministic semantic transformation:

$$\mathcal{O} : \Psi_t \to \Psi_{t+1}.$$

Operator semantics include declared scope, interference properties, and permission requirements.

A.4 Interference Relation

An interference relation is a directed semantic influence from one state component to another.

Formally, an interference relation is a tuple:

$$(\psi_i \to \psi_j),$$

indicating that changes in ψ_i may influence the evolution of ψ_j under declared semantics.

Interference relations may be reversible or irreversible and may carry quantitative bounds.

A.5 Sandbox Zone

A sandbox zone Z is defined as a triple:

$$Z = (\Psi_Z, \mathcal{O}_Z, \mathcal{I}_Z),$$

where:

- $\Psi_Z \subseteq \Psi$ is the subset of state governed by the zone;

- $\mathcal{O}_Z$ is the set of operators permitted within the zone;
- $\mathcal{I}_Z$ is the set of permitted interference relations involving Ψ_Z.

Any operator application affecting Ψ_Z must satisfy $\mathcal{I}_Z$.

A.6 Interference Permission

An interference permission explicitly authorizes a specific interference relation. Permissions may specify:

- direction of influence;
- scope of affected state;
- reversibility or collapse requirement;
- quantitative strength limits.

All interference permissions must be explicitly declared. Implicit permissions are forbidden.

A.7 Blast Radius

The blast radius of an operator application $\mathcal{O}$ within a sandbox zone Z is defined as:

$$B(\mathcal{O}, Z) = \text{closure}(\mathcal{I}_Z, \text{scope}(\mathcal{O})),$$

where closure denotes the transitive closure of permitted interference relations reachable from the operator's scope.

Any semantic effect outside $B(\mathcal{O}, Z)$ constitutes a sandbox violation.

A.8 Sandbox Violation

A sandbox violation occurs when an operator induces an interference relation not contained in $\mathcal{I}_Z$ or exceeds declared blast radius limits.

Sandbox violations are semantic errors detected during operator application.

A.9 Intrusion Event

An intrusion event is defined as any sandbox violation, whether caused by malicious intent, faulty operator behavior, or misconfiguration.

Intrusion events are explicit state transition events and are recorded in the execution trace.

A.10 Intrusion-Triggered Collapse

An intrusion-triggered collapse is a deterministic operator invoked in response to an intrusion event. The collapse:

- terminates or isolates the violating execution context;
- resolves accumulated interference;
- commits the system to a stable post-intrusion state.

Collapse operators are subject to replay and cryptographic sealing.

A.11 Zone Composition and Nesting

Sandbox zones may be nested. For zones Z_1 and Z_2 where $Z_2 \subseteq Z_1$, interference permissions must satisfy:

$$\mathcal{I}_{Z_2} \subseteq \mathcal{I}_{Z_1}.$$

Sub-zones may impose stricter constraints but may not weaken parent-zone permissions.

A.12 Replay Consistency

A replay execution is compliant if and only if it reconstructs:

- identical sandbox zone definitions;
- identical interference permissions;
- identical detection of violations;
- identical intrusion-triggered collapse behavior.

Any divergence constitutes a violation of sandbox semantics.

A.13 Summary

This appendix provides formal definitions for sandbox zones, interference permissions, blast radius, and intrusion handling in QVM–QFM. These definitions establish security as an explicit, enforceable, and replayable semantic property of execution.

B Security Contract

B.1 Purpose of the Contract

This appendix defines the Security Contract governing all executions claiming compliance with the Security-Native Interference Sandbox for QVM–QFM.

The Security Contract specifies mandatory obligations under which sandboxing, interference control, blast-radius containment, and intrusion handling are considered valid, deterministic, replayable, and legally verifiable.

Any execution violating this contract is non-compliant by definition, regardless of observed outcomes.

B.2 Scope of the Contract

The Security Contract applies to:

- definition and enforcement of sandbox zones;
- declaration and evaluation of interference permissions;
- blast-radius containment and boundary semantics;
- cryptographic sealing of state evolution;
- detection, handling, and recovery from intrusion events.

The contract is independent of runtime implementation, hardware platform, and deployment topology.

B.3 Sandbox Zone Obligations

An execution *must* satisfy the following sandbox zone obligations:

- all sandbox zones shall be explicitly declared;
- each zone shall define its governed state scope, permitted operators, and interference permissions;
- no operator may affect zone-governed state outside declared permissions.

Implicit or inferred sandbox boundaries are prohibited.

B.4 Interference Permission Obligations

All interference relations *must* satisfy:

- explicit declaration of direction, scope, and limits;
- semantic evaluation during operator application;
- prohibition of undeclared or transitive implicit interference.

Any interference not explicitly permitted constitutes a sandbox violation.

B.5 Blast-Radius Obligations

Blast radius *must* be bounded by declared interference permissions. The execution environment shall ensure that:

- no semantic effect propagates beyond the declared blast radius;
- boundary crossings occur only through explicit boundary operators;
- blast-radius constraints are enforced deterministically.

Runtime containment mechanisms may not substitute for semantic enforcement.

B.6 Violation Detection Obligations

Sandbox violations *must* be detected deterministically at the time of operator application.

Heuristic detection, probabilistic monitoring, or post hoc inference is forbidden as a substitute for semantic violation detection.

B.7 Intrusion Handling Obligations

Upon detection of an intrusion event, the execution environment *must*:

- record the intrusion as an explicit semantic event;
- invoke an intrusion-triggered collapse or isolation operator defined by security policy;
- enforce blast-radius containment during collapse.

Silent continuation after intrusion is prohibited.

B.8 State Sealing Obligations

All security-relevant state transitions *must* be cryptographically sealed.

The sealing mechanism shall:

- bind each state transition to prior sealed state;
- include operator identity, scope, and resulting state hash;
- prevent undetected reordering, removal, or modification.

Failure to seal or verify seals constitutes a fatal integrity error.

B.9 Replay and Verification Obligations

A compliant execution *must* support deterministic replay that reconstructs:

- identical sandbox zone definitions;
- identical interference permissions;
- identical blast-radius constraints;
- identical intrusion detection and collapse behavior;
- identical cryptographic seal chain.

Any divergence constitutes a Security Contract violation.

B.10 Failure Semantics

Failures, including execution interruption or partial collapse, shall not invalidate the Security Contract.

Failures must be represented as explicit state transitions subject to sandbox constraints, sealing, and replay.

Implicit retries, silent rollback, or evidence erasure are prohibited.

B.11 Non-Substitutability of Trust

Compliance with this contract shall not rely on trust assumptions about:

- operator correctness;
- execution environment integrity;
- network reliability;
- administrative control.

Security guarantees arise solely from semantic enforcement and cryptographic sealing.

B.12 Legal and Regulatory Interpretation

For legal and regulatory purposes, violation of this Security Contract implies that:

- sandbox boundaries were not semantically enforced;
- intrusion impact cannot be reliably bounded;
- execution history lacks admissible integrity guarantees.

The Security Contract therefore provides a binary criterion for admissibility of security-related execution evidence.

B.13 Summary

This Security Contract establishes enforceable obligations for security-native sandboxing in QVM–QFM. By defining security as a contractual property of execution semantics, the framework ensures deterministic containment, replayable intrusion handling, and legally robust forensic evidence.

C Claim-Ready Summary (US / EPC)

C.1 Overview

This appendix provides a claim-ready summary of the inventive concepts disclosed in this document. The invention concerns a security-native sandboxing framework for distributed computation executed by a Quantum Virtual Machine (QVM) operating over Quansistor Field Mathematics (QFM).

The framework defines security as a semantic property of operator interaction and interference, rather than as a runtime containment or trust-based mechanism.

C.2 Independent Claim (Core Invention)

Claim 1 (Independent). A computer-implemented method for enforcing security in distributed computation, the method comprising:

- executing instructions as deterministic operators acting on an explicit architectural state field governed by Quansistor Field Mathematics;
- defining one or more sandbox zones, each sandbox zone specifying governed state scope, permitted operators, and explicit interference permissions;
- evaluating operator application semantically to determine whether induced interference is permitted within an active sandbox zone;
- bounding semantic influence of operator application by a declared blast radius derived from permitted interference relations;
- detecting sandbox violations deterministically as semantic events;
- cryptographically sealing state evolution to preserve integrity and ordering of security-relevant events;
- invoking an intrusion-triggered collapse upon detection of a sandbox violation to contain impact and commit to a stable post-intrusion state;
- whereby security is enforced through semantic interference constraints rather than runtime isolation or trust assumptions.

C.3 Dependent Claims

Claim 2. The method of Claim 1, wherein interference permissions explicitly specify direction, scope, reversibility, or quantitative limits of influence.

Claim 3. The method of Claim 1, wherein sandbox zones are nested and composable, such that sub-zones impose stricter interference constraints than parent zones.

Claim 4. The method of Claim 1, wherein the blast radius is defined as the transitive closure of permitted interference relations reachable from an operator's scope.

Claim 5. The method of Claim 1, wherein boundary operators are required to permit any semantic influence crossing sandbox zone boundaries.

Claim 6. The method of Claim 1, wherein intrusion-triggered collapse preserves forensic evidence and prohibits rollback or erasure of intrusion events.

Claim 7. The method of Claim 1, wherein cryptographic sealing links each state transition to prior sealed state using operator identity, scope, and resulting state hash.

Claim 8. The method of Claim 1, wherein deterministic replay reconstructs identical sandbox zones, interference permissions, violation detection, collapse behavior, and cryptographic seal chain.

C.4 System Claims

Claim 9. A distributed computational system comprising:

- a Quantum Virtual Machine configured to execute deterministic QFM operators;
- a sandbox management mechanism defining semantic sandbox zones and interference permissions;
- an interference evaluation mechanism enforcing blast-radius containment;
- a cryptographic sealing mechanism binding semantic state evolution to verifiable integrity;
- an intrusion-handling mechanism invoking deterministic collapse and recovery;

wherein the system provides replayable and legally verifiable security semantics independent of runtime isolation or trusted hardware.

C.5 Use-Case and Policy Claims

Claim 10. The system of Claim 9, wherein security policies determine recovery or escalation actions following intrusion-triggered collapse without altering semantic enforcement.

Claim 11. The system of Claim 9, wherein sandbox violations and blast-radius impact are reconstructed for forensic, legal, or regulatory analysis.

C.6 Non-Claims and Scope Clarification

The invention does not claim:

- prevention of all intrusions or attacks;
- confidentiality or secrecy of computation or data;
- availability or liveness guarantees;
- denial-of-service protection;
- reliance on trusted execution environments or hardware enclaves.

The invention claims only deterministic, explicit, and reconstructible semantic security enforcement through interference control.

C.7 EPC-Oriented Reformulation Notes

For European Patent Convention filings, Claim 1 may be reformulated as a computer-implemented method producing the technical effects of:

- bounded and explicit impact of security incidents;
- elimination of implicit or timing-based security assumptions;
- verifiable integrity and ordering of security-relevant execution events in distributed systems.

C.8 Summary

This claim set defines a security-native interference sandbox for distributed computation. By constraining semantic influence, bounding blast radius, sealing state evolution, and handling intrusion through deterministic collapse, the invention establishes accountable, replayable, and legally robust security semantics at scale.