

Certified Confinement: Kernel-Checked Validation of Untrusted Quantum-Cloud Transpiler Output

Ali Hamza Malik

University of Massachusetts Amherst
Amherst, Massachusetts, USA
ahmalik@umass.edu

Muhammad Taqi Raza

University of Massachusetts Amherst
Amherst, Massachusetts, USA
taqi@umass.edu

Abstract

Cloud providers are adopting spatial multi-tenancy to improve quantum processing unit (QPU) utilization. However, this introduces cross-tenant vulnerabilities: otherwise hardware-legal circuits may place operations adjacent to co-tenant qubits, creating a leakage channel via physical coupling. Because tenants cannot safely trust cloud transpilers, we introduce an efficient client-side validator for buffered gate-support confinement. Given a device coupling graph, a forbidden region, and a buffer radius k , the validator rejects circuits containing non-identity gates within the forbidden zone or its k -hop neighborhood. We formally establish this structural guarantee via a Lean soundness theorem. In QASMBench experiments, we find that up to 71% of stock transpiler outputs violate this buffered policy despite being hardware-legal. Our validator catches these violations with zero false rejections when the allocation correctly reserves the buffer, yielding kernel-checked verdicts across 11 IBM device topologies. To empirically justify the buffer radius, we characterized two IBM Heron r2 QPUs. Under a 5σ distinguishing criterion, nearest-neighbor (one-hop) leakage is detectable in a median of just 186 shots, whereas two-hop leakage requires an estimated 1.3×10^5 to 1.9×10^7 shots, remaining undetectable under tested conditions. Together, our hardware measurements justify graph-buffered allocation against nearest-neighbor channels, while our formal certificate provides a rigorous guarantee of gate-support confinement.

CCS Concepts

• **Security and privacy** → **Logic and verification**; *Virtualization and security*; • **Software and its engineering** → *Formal software verification*.

Keywords

quantum cloud security, multi-tenant confinement, quantum compilation, translation validation, post-transpilation validation, side channels, proof assistants

ACM Reference Format:

Ali Hamza Malik and Muhammad Taqi Raza. 2026. Certified Confinement: Kernel-Checked Validation of Untrusted Quantum-Cloud Transpiler Output. In *Post-Quantum and Quantum-based Security (PQQS '26)*, November 02–04, 2026, Milpitas, CA, USA. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3839338.3857240>



This work is licensed under a Creative Commons Attribution 4.0 International License. *PQQS '26, Milpitas, CA, USA*

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2920-1/2026/11
<https://doi.org/10.1145/3839338.3857240>

1 Introduction

Cloud quantum services let users run workloads on quantum processing units (QPUs) without operating local cryogenic infrastructure. Scarce, expensive quantum hardware motivates cloud providers to co-locate user workloads on a single chip [6, 8, 9, 37]. Under spatial multi-tenancy, mutually distrusting tenants execute quantum circuits simultaneously on disjoint physical qubit allocations of the same QPU.

Cloud providers control circuit placement and compilation. A cloud transpiler rewrites each high-level circuit to conform to chip connectivity, inserting SWAP operations to route interacting qubits onto physically coupled sites. Cloud compilation software is proprietary, complex, and untrusted. Current public cloud interfaces generally do not give tenants a provider-enforced, cryptographically bound certificate that the executed artifact respects a tenant-selected graph-buffered region.

Co-location also exposes tenants to physical interference. On superconducting quantum hardware, crosstalk and always-on ZZ couplings let operations on one qubit perturb state populations and measurement outcomes on neighboring qubits [18, 35]. Standard compilation checks establish only that an emitted circuit is *hardware-legal* (that every gate targets physically connected qubits). A hardware-legal circuit can still execute operations directly adjacent to a co-tenant's allocation, exposing sensitive quantum states to neighbor leakage.

We give tenants a client-side, kernel-checked validator that verifies physical placement and buffered gate-support confinement on emitted circuit artifacts *before* execution.

Limits of hardware legality. Hardware legality does not guarantee tenant gate-support confinement. On superconducting QPUs, physical coupling channels extend beyond the target qubits. A hardware-legal placement can leave a retained victim state one coupling edge from a co-tenant probe. On live IBM Heron hardware, that adjacent placement shifts probe outcome probabilities by $\Delta P = 0.397$ ($z = 86$, §5.4), even without active victim gates. Policies based solely on disjoint physical qubit sets miss attack surfaces that follow coupling-graph *adjacency*.

Approach. Prior approaches require tenants to trust the provider's compilation stack. Quantum compilers optimize for hardware legality and gate count, without enforcing buffered gate-support confinement. We apply *translation validation* [29, 31, 39] in the proof-carrying-code tradition [28] to check each emitted circuit artifact against a formal policy over the device coupling graph. VeriWasm provides an architectural precedent by checking native WebAssembly sandboxing after compilation to address implementation bugs in multi-tenant cloud compilers [14]. We mechanize the validator as a decision procedure in Lean [7]. The

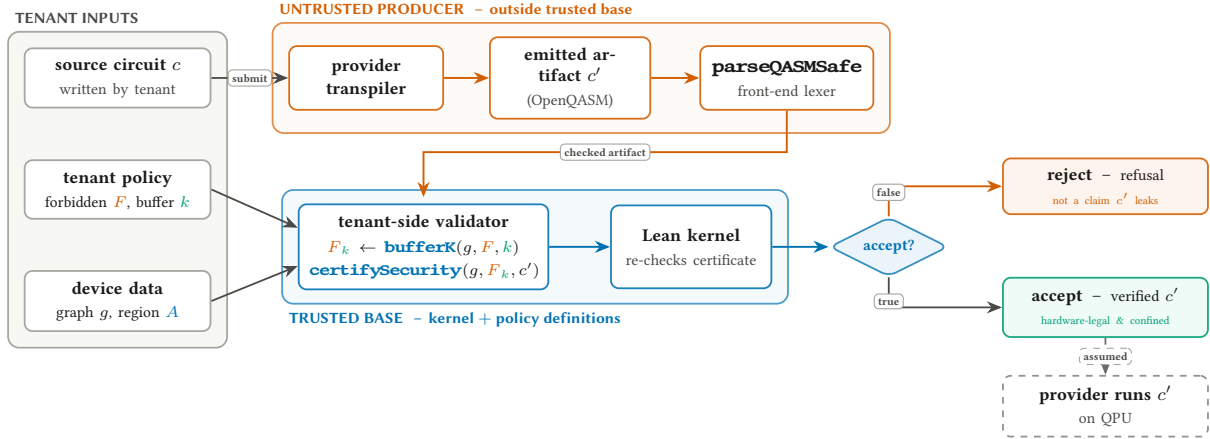


Figure 1: System overview and trust boundary. Orange components are untrusted. The validator rejects transpiler output that violates the checked placement policy (Theorem 4.1). Front-end fuzzing tests reject-safety (§3.3). The validator re-derives confinement from the emitted circuit, the published coupling graph, and the allocated region. The Lean kernel re-checks the certificate. The dashed edge represents execution binding: the provider must run the verified artifact c' (Appendix B). This deployment assumption lies outside the Lean proof.

proof kernel re-checks each generated certificate. Figure 1 shows the trust boundary.

Graph-mediated leak policy. Instruction filtering alone cannot eliminate crosstalk because physical coupling is intrinsic to the hardware. We define the security policy over the device coupling graph. Given a forbidden co-tenant region F and a safety radius of k hops, the policy expands F into a k -hop buffered region F_k . The validator requires the compiled circuit’s non-identity gate support to remain disjoint from F_k . A provider-enforced allocation within the remaining region also keeps retained states outside that buffer. Setting $k=1$ excludes the nearest-neighbor placements examined in our hardware study. Larger k values exclude more graph-near placements.

Results and scope. The validator soundly checks well-formed circuits represented in its IR against the supplied graph policy. A fuzz-tested OpenQASM front end rejects unsupported syntax before validation (§3.3). After device-graph preprocessing, validation makes a single pass over the circuit gate list. On QASMBench [21] circuits, Qiskit’s stock transpiler violates buffered gate-support confinement in 18% of outputs at optimization level 0, 53% at level 1, and 71% at levels 2 and 3. Every violating circuit is hardware-legal (§5.1). When the tenant allocation provides a free buffer, the validator incurs zero false rejections. The checker scales across 11 IBM device topology snapshots, up to the 156-qubit IBM Heron r2 chip.

On live IBM Heron r2 hardware, the one-hop policy rejected every hardware-legal placement that leaked to an emulated co-tenant. Across 23 pair-observations, the policy accepted no placement exhibiting leakage at the tested detection threshold (§6). Under a 5σ criterion for distinguishing the prepared victim states, the estimated median cost was 186 shots at one hop. At two hops, estimates ranged from 1.3×10^5 to 1.9×10^7 shots. One rejected placement was silent during initial measurement but leaked following recalibration, illustrating the policy’s conservative rejection (§5.4).

Contributions.

- **Tenant-checkable buffered gate-support confinement.** We mechanize a coupling-graph-buffered gate-support confinement policy and a kernel-checked decision procedure in Lean. The validator rejects both direct forbidden-region access and hardware-legal placements adjacent to forbidden qubits (§3, §3.2).
- **Validation of untrusted transpiler output.** We apply translation validation to quantum cloud compilation, removing the cloud transpiler from the trust boundary. Tenants verify structural placement policies on supported emitted artifacts without trusting proprietary vendor software (§3.3, §3).
- **Empirical evaluation across three domains.** We find that stock Qiskit compilation violates buffered gate-support confinement in up to 71% of optimized outputs, confirm that kernel validation scales to 156-qubit production topologies, and observe on live IBM Heron r2 hardware that the one-hop policy rejects all placements with detected leakage in our sample (§5).
- **Open-source Lean 4 proof artifact.** The complete Lean 4 mechanization, OpenQASM 2.0 front end, evaluation benchmarks, and hardware experiment logs are available as a public artifact repository.¹

2 Background and Threat Model

Multi-tenant quantum execution requires a model of coupling-graph topology and spatial attack boundaries.

2.1 Quantum programs on shared hardware

A quantum program applies a sequence of discrete operations to physical qubits and denotes a unitary matrix. Two programs are physically equivalent when their unitary matrices agree up to a *global phase* factor $e^{i\phi}$. Global phase does not change measurement outcomes, so correctness verification checks equivalence under global phase. Our formal relation **CongPhase** captures equality up

¹<https://github.com/alirafiqmalik/certified-confinement-acm-pqqqs-26-artifacts>

to a unit scalar, matching the equivalence notions resolved by industrial verification tools [4].

Cloud providers are moving toward spatial multi-tenancy to share scarce quantum processing units (QPUs) (§1). Spatial multi-tenancy co-locates mutually distrusting tenant workloads on disjoint physical qubit sets of the same chip. A hardware *coupling graph* $g = (V, E)$ represents physical qubits as nodes and multi-qubit coupling channels as edges. A compiled program is *hardware-legal* when every multi-qubit gate targets an edge in g (§3.1).

High-level circuits rarely match the coupling graph. The transpiler inserts SWAP gates to make interacting qubits adjacent, relocating tenant data across the chip.

Hardware legality ensures that multi-qubit operations target wired pairs. Buffered gate-support confinement between neighboring tenant allocations requires a separate check. On superconducting QPUs, coupling channels create parasitic interactions across adjacent graph edges. Always-on ZZ couplings and readout crosstalk let operations on one qubit perturb state populations and measurement distributions on neighboring qubits [18, 35]. Gates adjacent to a co-tenant allocation can therefore create physical leak channels in hardware-legal programs.

2.2 Threat model

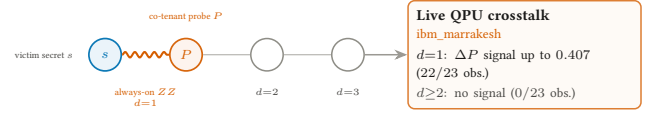
Adversary. The adversary is the *compilation path*: the cloud transpiler and intermediate passes that rewrite the circuit before execution. We grant it full operational capability: it reads the source circuit and tenant policy, may contain bugs or malicious code, and may emit arbitrary circuits targeting co-tenant physical qubits. Its goal is to induce execution of a circuit whose spatial footprint violates the tenant allocation. A passive *reading co-tenant* on neighboring physical qubits exploits this violation. It operates solely within its own allocation and reads the victim state through physical coupling channels (§5.4 emulates this role with a probe qubit). The compilation path cannot forge Lean kernel proof certificates or alter the published coupling graph or tenant policy. Our defense excludes providers executing circuits distinct from checked artifacts (*execution binding*, §6.1), physical channels beyond graph adjacency, and gate-to-pulse lowering passes that alter gate-level semantics. Table 5 in Appendix B summarizes the trust contract.

Geometry. The provider grants the tenant a physical region A_{grant} and assigns co-tenants to a separate region F . For a k -hop policy, the allowed region $A = V \setminus \text{Buf}_k(F, G)$ excludes F and its reserved halo. Deployment requires $A_{\text{grant}} \subseteq A$. Physical attack surfaces follow coupling-graph adjacency. Figure 2 relates this geometry to the physical interaction channels measured in §5.4.

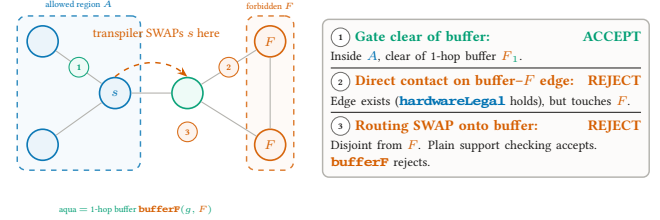
Buffered confinement. We require every compiled circuit c to satisfy

$$\text{supp}(c) \cap \text{bufferK}(g, F, k) = \emptyset,$$

so non-identity gates access only qubits outside F and its k -hop neighborhood in coupling graph g (§3.2). The support predicate $\text{supp}(c)$ excludes identity-only qubit references, matching the mechanized decision procedure `confinedb` (§3.1). Plain support confinement ($\text{supp}(c) \cap F = \emptyset$) is a weaker intermediate predicate from the soundness theorem (§4.1). Applying that theorem to the expanded region $\text{bufferK}(g, F, k)$ yields the buffered policy. Figure 2(b) shows that plain confinement accepts hardware-legal placements directly



(a) Measured physical channel: one hop leaks via ZZ crosstalk across a single shared coupling edge on Heron r2. A two-hop separation showed no detectable leak under the tested conditions (§5.4).



(b) Hardware legality and policy geometry: adjacency requires buffer confinement. In placement 2, a gate crosses the buffer- F edge. `hardwareLegal` holds despite forbidden-region access. In placement 3, a SWAP moves the secret onto the buffer qubit. A plain support check against F accepts it, whereas the buffered policy rejects it. The shaded A excludes the buffer.

Figure 2: Threat model and policy geometry.

adjacent to forbidden regions. The buffered policy excludes these adjacent gate placements. Buffered gate-support confinement constrains non-identity operation support. Full physical state isolation and result corruption analysis remain separate operational requirements (§6).

Trust assumptions. Appendix B lists trust assignments across system components. Timing channels, coherence degradation, and general information-flow channels fall outside our formal threat model [24]. Deployment requires a global property: checking one tenant circuit does not prevent an unverified co-tenant from routing into the tenant's allocation. Blocking hostile co-tenants requires providers to validate every tenant artifact against mutually disjoint buffered regions (§6.1).

Allocator knowledge of co-tenants. The mechanized check `certifySecurity` accepts any forbidden region. If the provider identifies the co-tenant region F , the tenant can submit its k -hop expansion and check buffered gate support. A tenant that knows only its granted allocation A_{grant} instead sets the forbidden input to $V \setminus A_{\text{grant}}$ and checks gate support within the grant. In that case, the provider must reserve the halo and ensure $A_{\text{grant}} \cap \text{Buf}_k(F, G) = \emptyset$. The certificate alone cannot verify that assignment. Both modes require an accurate, authenticated allocation and coupling map (§6.1).

Leak vectors. Routing or optimization passes can force secret states near co-tenant subgraphs. We consider four threat vectors: (V1) SWAP-based routing attacks [18], (V2) readout crosstalk across adjacent physical qubits [35], (V3) inverse transpilation attacks [16], and (V4) always-on nearest-neighbor ZZ state coupling. V3 concerns classical circuit disclosure independently of physical co-location (§6.2). Section 5.4 presents live hardware

measurements for V4. Section 6 details the certificate’s protection guarantees across these four vectors.

3 Design

We verify each transpiler *output* with a tenant-checked certificate. The untrusted compiler may perform arbitrary computations, provided it emits an artifact that a small, independent referee can check. This architecture follows translation validation [29, 31] and the de Bruijn criterion of independently checking a proof [3]. It also shares structural properties with verified UNSAT certificate checking [17].

Formal Mathematical Specification (Layer 1: Confinement Policy)

Let $G = (V, E)$ be an undirected coupling graph ($V = \{0, \dots, n-1\}$), $F \subseteq V$ a declared forbidden set of physical qubits, $k \in \mathbb{N}$ a buffer radius, and c a compiled circuit.

1. k -Hop Topological Buffer

$$\text{Buf}_k(F, G) \triangleq \{v \in V \mid \exists u \in F, \text{dist}_G(u, v) \leq k\}$$

2. Policy-Allowed Region

$$A \triangleq V \setminus \text{Buf}_k(F, G)$$

3. Active Gate Support

$$\text{supp}(c) \triangleq \{q \in V \mid \exists g \in c, g \text{ acts on } q \text{ and } g \neq I\}$$

4. Buffered Confinement Policy Invariant

$$\text{supp}(c) \cap \text{Buf}_k(F, G) = \emptyset \iff \text{supp}(c) \subseteq A$$

Here F denotes the co-tenant region, and A is the region allowed by this buffered policy. The provider’s granted tenant allocation A_{grant} must satisfy $A_{\text{grant}} \subseteq A$. A tenant that knows only its grant can instead check support against $V \setminus A_{\text{grant}}$ at radius zero. It relies on the provider to reserve the buffer around co-tenants. Applying another buffer to $V \setminus A_{\text{grant}}$ would unnecessarily exclude boundary qubits of the grant.

3.1 Validator inputs and decision

The validator takes three tenant-accessible inputs: the provider-published coupling graph g , a forbidden qubit set F , and the emitted circuit c . It evaluates two structural predicates over these inputs. Given F , **certifySecurity** checks **confinedb** against $\neg F$. The tenant passes either the known co-tenant region after buffer expansion or $V \setminus A_{\text{grant}}$ to confine gates to its granted allocation (§2). Both checks use the same decision procedure, but only the first derives its buffer from the supplied co-tenant region.

Hardware legality. The decidable predicate **HWF**(g, c) states that every two-qubit gate in c targets a physical edge in g . Worked examples use two hand-encoded patches: a 12-node ring (**heavyHex**, a historical name for this 2-regular cycle) and a degree-3 four-qubit heavy-hex fragment (**heavyHexFrag**). The evaluation tests 11 real device topologies, including the 156-qubit Heron r2 graph (§5.3).

Support confinement. **confinedb**(A, c) folds over the gate list and checks each gate’s qubits against the allowed set A . Identity gates are exempt because they denote I in the circuit IR. This exemption does not constrain idle-state leakage. Throughout, $\text{supp}(c)$ denotes the qubits touched by non-identity gates in c . An identity gate that references a buffer or forbidden qubit therefore satisfies this gate-support policy. Idle qubit residency without non-identity gates remains a separate modeling assumption. The physical channel measured in §5.4 operates without gate activity, so the identity

Table 1: Layered system architecture and TCB boundaries.

Layer	Trust Status	Architectural Role & Security Guarantee
Formal core	Trusted	Lean 4 kernel, specifications, IR support, decidable instances, and topology encoders, with a kernel-checked structural verdict
Sanitization	Tested	OpenQASM 2.0 front-end with a sanitized lexer satisfying tested reject-safety invariant
Outer Shell	Untrusted	Provider transpiler (Qiskit / layout passes), with output checked by referee

exemption applies only to gate support (§6). The tenant-side counterpart **routeableb**(A, c) decides whether a source circuit admits routing within A and serves as a pre-submission check.

Trusted computing base. Table 1 summarizes the trusted computing base (TCB). The Lean 4 kernel checks proofs against specifications (**HWF**, **confinedb**, **bufferK**). Auditors must also inspect the circuit IR, decidable instances, and topology encoders that give these specifications their intended meaning (§4.1). The sanitized OpenQASM front-end lexer has a tested reject-safety invariant. Provider transpilers, placement algorithms, and routing passes remain untrusted. Execution binding and allocation accuracy are separate deployment assumptions (§6.1).

A single call to **certifySecurity** produces the certificate. **certifySecurity**(g, F, c) returns a record containing **hardwareLegal** := **decide**(**HWF**(g, c)), **policyConfined** := **confinedb**($\neg F, c$), and their conjunction, **accepted**. Figure 3 details the interface. The validator evaluates finite structural predicates over the emitted gate list without simulating quantum states. Under Theorem 4.1, transpiler bugs cannot cause the validator to accept a circuit that violates these predicates. Differential fuzzing tests the front end’s reject-safety invariant for parsing errors (§3.3).

3.2 Buffered confinement policy

Buffered confinement. **bufferK**(g, F, k) expands the forbidden region by k hops in the coupling graph. **bufferF** denotes the $k=1$ instance. The policy requires compiled circuits c to satisfy:

$$\text{supp}(c) \cap \text{bufferK}(g, F, k) = \emptyset,$$

so non-identity gates target qubits outside F and its k -hop coupling-graph neighborhood. Lemma **confinedb_avoids** connects this set-based condition to **confinedb** (§4.1). Since **certifySecurity_sound** quantifies over region inputs, instantiating it with **bufferK**(g, F, k) establishes the buffered guarantee without another soundness proof (§4). Larger k values cover longer-range graph interaction channels. Other physical channels, such as readout crosstalk across shared feedlines, require custom adjacency relations in the region-parametric formulation (§6).

Physical threat model and gate-support policy. Idle physical qubits carrying state can leak to neighbors through static ZZ coupling without gate activity (§5.4). The policy-allowed region $A = V \setminus \text{Buf}_k(F, G)$ separates retained states from forbidden qubits only when the provider places the tenant’s grant within A . The provider’s allocation contract ($A_{\text{grant}} \cap \text{Buf}_k(F, G) = \emptyset$) governs idle state placement. The client-side validator certifies that transpiled

<i>Syntax</i>	
n	$\in \mathbb{N}$
g	$\in \text{Coupling } n$
F, A	$\subseteq \{0, \dots, n-1\}$
u	$::= .xi \mid .sxi \mid .idi \mid .rzi \theta i$ $\mid .czi j \mid .cxij \mid \dots$
c	$::= \cdot \mid u; c$
<i>Specification</i>	
$\text{HWF}(g, c)$	every two-qubit gate of c on an edge of g
$\text{confinedb}(A, c)$	fold: each non-identity gate's qubits in A
$\text{bufferK}(g, F, k)$	F plus its k -hop neighborhood in g
<i>Inputs</i>	
g	device coupling graph (provider-published)
F	forbidden region of physical qubits (tenant-declared)
k	buffer radius, in hops of g (tenant policy)
c	compiled circuit (untrusted transpiler output)
<i>Once per device and region</i>	
$F_k \leftarrow \text{bufferK}(g, F, k)$	
<i>Once per submission, one pass over the gates of c</i>	
$\text{certifySecurity}(g, F_k, c) \rightarrow$	$\{\text{hardwareLegal}, \text{policyConfined}, \text{accepted}\}$
$\text{hardwareLegal} := \text{decide}(\text{HWF}(g, c))$	
$\text{policyConfined} := \text{confinedb}(\neg F_k, c)$	
$\text{accepted} := \text{hardwareLegal} \wedge \text{policyConfined}$	

Figure 3: Validator syntax and interface. **accepted** is the conjunction of **hardwareLegal** and **policyConfined**: every two-qubit gate of c lies on an edge of g , and no non-identity gate of c touches F or any qubit within k hops of F (§4.1). Rejection refuses submission of c . It does not assert that c leaks.

circuits contain no non-identity gates outside the checked allowed region, enforcing the gate-support boundary during execution.

Insufficiency of support confinement. Under unbuffered support confinement, secret states can occupy sites adjacent to co-tenant boundaries without gates acting on co-tenant qubits. Because **confinedb** checks gate support alone, unbuffered policies accept adjacent placements that produce measurable crosstalk ($\Delta P = 0.397$, §5.4). On our encoded device patch, an adjacent placement likewise yields **accepted** = **true** under unbuffered checks.

The checker rejects direct contact: on the degree-3 site **heavyHexFrag** ($A = \{0, 1, 2\}$, $F = \{3\}$), it accepts **.cz 0 1**, whereas **.cz 0 3** yields **hardwareLegal** = **true** and **accepted** = **false** (both evaluated via **decide**). Hardware legality alone does not enforce tenant boundaries. Verified routers such as VOQC prove SWAP paths hardware-conformant and equivalent up to qubit permutation [10]. Neither property constrains spatial tenant boundaries.

Cost of the buffer radius. Evaluating **bufferK**(g, F, k) for every gate costs $\Theta(q^k)$ per query on a q -qubit device, or $\Theta(\# \text{gates} \cdot q^k)$ overall. We precompute the buffered region as a Boolean array in $O(k \cdot q^2)$ time. The theorem **certifySecurity_bufferArrK** proves that the array check returns identical verdicts. With $O(1)$ edge lookups, per-circuit validation remains $\Theta(\# \text{gates})$. Tenants can reuse the array across submissions because precomputation depends only on device topology and allocation boundaries (§5.2).

Policy–claim mapping. We adopt buffered confinement as an established policy framing (§7). An accepted certificate under **bufferF** guarantees that no qubit acted on by a non-identity gate of c lies in F or shares a coupling edge with F . This guarantee

covers the direct-edge geometry measured in §5.4. At $k=1$, two-hop paths through sterilized qubits remain unconstrained. Larger radii k address them. The check covers gate-bearing qubits, while idle qubit channels require physical allocation boundaries (§6).

3.3 Front end and deployment model

An untrusted front end lexes OpenQASM inputs into circuit IR terms. Its security invariant requires parsing errors to produce safe rejections. An initial lexer silently skipped unrecognized tokens, omitting support-bearing gates and causing 269 unsound acceptances during differential fuzzing. The updated lexer rejects unrecognized or ill-formed tokens and produced no unsound acceptances on the fuzzing corpus (§5.1, Appendix C.3). Input formatting defects now trigger top-level parse failures before certificate checking.

3.4 Validation and construction modes

The system has two operational modes:

- **Validation Mode (Primary Security Result):** Applies to any well-formed emitted circuit c represented in the checked IR, regardless of which transpiler produced it. The kernel-checked certificate **certifySecurity** decides hardware legality and confinement against the supplied forbidden region without trusting the compiler.
- **Construction Mode (Feasibility Evidence):** Applies to our reference route-and-optimize pipeline. Preservation theorems (§4.4) establish compilation paths that preserve confinement and semantics up to global phase.

Validation lets tenants check supported provider outputs without assuming compiler correctness. The fuzz-tested OpenQASM front end rejects unsupported syntax before IR validation (§3.3). Construction mode supplies feasibility evidence for the reference pipeline.

4 Formal Guarantees

We implement the validator as a Lean 4 decision procedure over Mathlib [7, 38]. The artifact contains the complete proof development and machine-checkable axiom provenance reports for all stated theorems (Appendix C). It also includes a 2^n denotational semantics for the circuit IR, used only as ground truth for bounded equivalence proofs outside the security validation path.

4.1 Soundness of the certificate

An **accepted** = **true** verdict implies hardware legality and spatial gate-support confinement. The decision procedure checks **HWF**(g, c) for edge legality and **confinedb**($\neg F, c$) for support confinement.

Theorem 4.1: **certifySecurity_sound.** For every coupling graph g , region F , and circuit c , if **certifySecurity**(g, F, c) reports **accepted** = **true**, then **HWF**(g, c) and **confinedb**($\neg F, c$) hold: every non-identity gate of c acts only on qubits outside F .

The auxiliary lemma **confinedb_avoids** connects the Boolean verdict to set-theoretic disjointness ($\text{supp}(c) \cap F = \emptyset$). Table 6 records **propext** alone for both **certifySecurity_sound** and **confinedb_avoids**. Theorem 4.1 uses **confinedb**, the form returned by the decision procedure. Disjointness follows without additional axioms.

Theorem 4.1 quantifies over circuit IR terms c , so it applies to any well-formed transpiler output represented in that IR (§3.4). It also quantifies over regions F : instantiating it with `bufferK(g, F, k)` proves that no non-identity gate in c touches F or its k -hop neighborhood (§3.2). Its proof is a one-line definitional unfold. Here, `decide` reflects `HWF`. Auditing the result requires inspecting `HWF`, `confinedb`, and `bufferK` over the IR's gate-support definitions. This shallow proof of Theorem 4.1 follows the de Bruijn criterion [3].

Trust base. Auditors must inspect the Lean proof kernel, policy specifications (`HWF`, `confinedb`, `bufferK`), circuit IR support definitions, decidable instances, topology encoders, and tested front-end reject-safety (§3.3). The same decision procedure supports formal reasoning in Theorem 4.1, kernel reduction via `decide` (§5.3), and executable binary evaluation for latency measurement (§5.2). Denotational semantics and phase equivalence (`CongPhase`) enter only construction-mode proofs (§4.3). All theorems have axiom audits and contain neither `sorry` nor custom axioms (Table 6, Appendix C).

4.2 Cost of a certificate

The decision procedure visits each gate once: sequential composition adds the step counts of its component circuits.

Theorem 4.2: `checkerSteps_seq`. The security check's step count is additive over sequential composition: `checkerSteps` of a sequential composition equals the sum of its component step counts. The check therefore visits each gate exactly once.

Theorem 4.2 establishes structural step-count additivity. Inline topological buffer queries still cost $\Theta(q^k)$ each on q -qubit devices. We instead precompute the region in $O(k \cdot q^2)$ time. `certifySecurity_bufferArrK` proves the array check verdict-identical. With $O(1)$ region and edge lookups, each circuit requires one linear pass over its gates, independent of device size and state-space dimension 2^n .

4.3 Symbolic routing

Construction-mode feasibility (§3.4). SWAP insertions permute physical qubit assignments. Direct comparisons of $2^n \times 2^n$ unitary matrices scale exponentially. Symbolic routing instead models SWAPs as permutations of physical qubit indices, avoiding state-vector transformations. These permutations support proofs of semantic equivalence up to global phase without dependence on 2^n . Our development uses this approach on heavy-hex topologies as constructibility evidence and verifies hardware legality along the transformation path.

VOQC proves SWAP paths hardware-conformant and equivalent up to qubit index permutations [10]. Our construction-mode proof reuses this permutation invariant for confinement: induction over the SWAP network tracks index mappings and keeps routing operations within allowed spatial bounds. Permutation semantics (`bitSwap`, `permDenote`) track qubit locations. Theorems `route_hh_congPhase` and `route_hh_HWF` establish `CongPhase(route_hh(c), c)` and hardware legality, respectively. `findPath_valid` discharges path validity. The same machinery

keeps routed gates on their paths' qubits (`routeEdge_confined`, `swapEdgeNet_confined`).

Construction-mode equivalence covers heavy-hex routing paths on stylized circuits as feasibility evidence. Translation validation (Theorem 4.1) applies separately to third-party transpiler outputs on 156-qubit production topologies (§6).

4.4 Compilation preserves confinement

Construction-mode preservation. Both rewrite stages preserve confinement. Routing maintains spatial boundaries for routable source circuits. Optimization eliminates or fuses existing operations. By induction on the circuit, `route_hh_confine` reduces routing confinement to `routeEdge_confined`. The theorem `optimize_conf`, through `conf1_fuse`, `conf_toList`, and `conf_ofList`, proves that cancellation and fusion can only shrink the qubit support.

End-to-end compilation theorem. Composing these passes, `compile_hh_confine_correct` proves that the reference pipeline produces hardware-legal, phase-equivalent, and spatially confined outputs. Its buffered counterpart, `compile_hh_confine_buffered`, states:

```
compile_hh_confine_buffered : WF(c) → routableb(¬bufferF(g, F), c) = true →
  HWF(heavyHex, optimize(route_hh(c))) ∧
  CongPhase(optimize(route_hh(c)), c) ∧
  confinedb(¬bufferF(g, F), optimize(route_hh(c))) = true
```

The theorem `compile_hh_confine_bufferedK` extends this result to arbitrary k , and `certify_compile_hh_buffered` establishes that the buffered certifier accepts our pipeline's output.

The hypotheses require well-formed source circuits (`WF(c)`) and valid initial placements (`routableb`). The hardware-legality proof uses stylized ring topologies (`heavyHex`). This construction-mode result establishes feasible confinement-preserving compilation. Translation validation verifies third-party transpiler outputs on multi-tenant QPUs.

5 Evaluation

We evaluate validator performance, scalability, and alignment with measured QPU behavior along five dimensions:

- **E1 (Policy Compliance):** How frequently does stock compilation violate buffered gate-support confinement, and does the validator accept all compliant compiled circuits? (§5.1)
- **E2 (Validator Practicality):** Does validation latency support inline submission filtering, and does region memoization maintain single-pass efficiency? (§5.2)
- **E3 (Cross-Device Scalability):** Does kernel-checked validation scale across production QPU topologies? (§5.3)
- **E4 (Physical Motivation):** Does the graph-buffered policy boundary align with crosstalk and state-leakage behavior measured on physical QPU hardware? (§5.4)
- **E5 (Capacity Trade-offs & Operability):** What chip capacity does buffer reservation consume, and can a tenant evaluate the policy using only published device metadata? (§5.5, §5.5)

Evaluation summary. Three results summarize the evaluation:

- (1) *Transpiler non-compliance and zero false rejections:* Unmodified Qiskit compilation on a 12-node ring topology violates graph-buffered confinement in 18% (optimization level 0) to 71%

(levels 2–3) of hardware-legal outputs. When tenant allocations reserve the required graph buffer, the validator makes no false confinement rejections among the tested circuits (§5.1).

- (2) *Validator efficiency and kernel scalability:* The validator makes one pass over the gate list. Unbuffered validation of a 10^4 -gate circuit takes 6 ms on the 156-qubit graph. Lean kernel reduction checks 44 placement verdicts across 11 IBM device graphs (7–156 qubits) in under 5 seconds (§5.2, §5.3).
- (3) *Physical hardware alignment:* Across 23 pair-observations on physical IBM Heron r2 hardware, the 1-hop policy rejects adjacent placements with detected leakage ($\Delta P = 0.397$, $z = 86$ in the primary run). Blind re-scoring with published coupling maps misses no measured leaks among the 46 tested distance verdicts (§5.4, §5.5).

Methodology and evidence arms. We evaluate one fixed, proved-sound policy across three independent evidence arms: (A) kernel checks on real device graphs, (B) stock provider transpiler outputs on QASMBench benchmarks, and (C) live QPU hardware campaigns. Arms A and B evaluate validation at zero QPU cost. We fixed and mechanized the policy and decision procedures before the hardware campaigns. The E4 comparison thus tests a pre-specified structural prediction. We did not fit the model to hardware data. We pre-registered Arm C and used only published vendor coupling maps for blind re-scoring. The full Lean 4 codebase, experimental harnesses, measurement logs, and reproduction scripts are available in the public artifact repository (<https://github.com/alirafiqmalik/certified-confinement-acm-pqqs-26-artifacts>). We analyze deployment conditions in §6, review prior work gaps in §7, and detail calibration snapshots, job provenance, and simulation harnesses in §5.4.

5.1 E1: Policy Compliance

Hardware legality does not imply confinement. Unmodified provider compilation emits hardware-legal circuits that violate buffered gate-support confinement, increasingly often at higher optimization levels. We evaluated the fittable QASMBench [21] benchmark suite (≤ 12 qubits, $n = 38$ circuits per optimization level) through Qiskit’s stock **transpile** targeting an IBM basis on a 12-node ring topology, passing the compiled outputs to our lexer satisfying a tested reject-safety invariant and validator. When the transpiler places qubits without spatial constraints, confinement violation rates increase monotonically with optimization effort: 18% at optimization level 0, 53% at level 1, and 71% at levels 2 and 3 (Figure 4). Every violating output remains hardware-legal. The validator rejects all observed policy breaches. Thus, hardware legality alone does not enforce the tenant boundary in this compilation setting.

Zero false rejections under buffered allocation. We measured acceptance on circuits compiled for buffered tenant allocations. We modeled the intended deployment discipline on the 12-node ring: the provider reserves a k -hop halo around the co-tenant as an unallocated buffer zone and restricts the tenant transpiler to the induced subgraph of remaining allowed qubits.

With transpilation restricted to the allowed allocation, the validator makes no false confinement rejections at the tested buffer radii and optimization levels (Table 2). At $k=0$ (tenant allocated

Table 2: Buffer operability and benchmark fitness under spatial isolation (E1). QASMBench circuits ($n = 38$) fit within the complement of the k -hop buffer around $F = \{6, \dots, 11\}$ on a 12-node ring. Across all valid allocations and optimization levels 0–3, false rejections remain zero.

Radius k	Allowed Set A	Fit ($n=38$)	Accepted	False Rejects
0	$\{0, 1, 2, 3, 4, 5\}$	31	29	0
1	$\{1, 2, 3, 4\}$	23	23	0
2	$\{2, 3\}$	6	6	0
3	$\emptyset$ (sterilized)	0	0	0

$\{0, \dots, 5\}$), 31 of 38 benchmark circuits fit the allocation, and the validator accepts 29. The lexer rejects the other 2 circuits because they use unsupported classical conditioning. Neither is a validator rejection for a confinement violation. At $k=1$ (tenant allocated $\{1, 2, 3, 4\}$), the validator accepts all 23 fitting circuits. At $k=2$ (tenant allocated $\{2, 3\}$), all 6 fitting circuits pass.

Reserving a spatial buffer reduces available chip capacity (§5.2). On a degree-2 ring, each additional hop consumes two qubits, reducing the count of benchmark circuits that fit from 31 at $k=0$ to 23 at $k=1$, 6 at $k=2$, and 0 at $k=3$ (full sterilization). Higher-degree topologies consume capacity faster. Device degree and program size therefore constrain the usable buffer radius k .

Crafted and adversarial witnesses. We evaluated the validator against concrete benchmark scenarios designed to stress its trust boundary, summarized in Table 3. Three witness circuits check baseline verdicts: a routing SWAP operation that moves secret data across the tenant boundary receives the correct rejection, an externally optimized circuit invading forbidden qubits receives a rejection, and a fully confined positive circuit on the 12-node ring receives acceptance without false alarms. Two crafted hostile-transpiler witnesses explicitly target the validator interface. Witness W1 (legal-but-adjacent) is hardware-legal and support-disjoint from F but places operations one hop away. A support-only policy accepts W1, while the $k=1$ buffered policy rejects it, addressing the adjacent-placement gap in §3.2. Witness W2 (lexer evasion) attempts to smuggle classically-conditioned gates through front-end parsing. The lexer satisfying a tested reject-safety invariant rejects W2 while the legacy parser accepted it. Lean mechanizes the circuit witness verdicts. The lexer evasion verdict is an executable front-end result (Appendix C.4, Appendix C.3).

Trust-boundary fuzzing. To evaluate front-end parsing robustness, we executed differential fuzzing over a corpus of 1,120 mutant QASM files. While the legacy parser permitted 269 unsound acceptances, the lexer satisfying a tested reject-safety invariant eliminated all false acceptances across the corpus (0 unsound accepts). In this corpus, the revised lexer blocks parser evasion before the verified core (Appendix C.3). Across these scenarios, the validator rejected every tested confinement breach and produced no false confinement rejections on compliant circuits. The lexer rejected the tested evasion attempts.

Table 3: Evaluated security witness circuits and validation verdicts. Verdicts are kernel-reduced via `decide` except where string operations or general recursion require executable evaluation (`#eval`). See §5.1 and Appendix C.4.

Witness Case	HW Legal?	Verdict	Technical Rationale
Direct contact (<code>.cz 0 3</code>)	✓	Reject	Touches forbidden set <i>F</i>
End-to-end QASM pipeline	✓	Reject	Touches forbidden set <i>F</i>
Boundary-crossing SWAP	✓	Reject	Routing path exits allowed region <i>A</i>
External optimizer breach	✓	Reject	Gate support spills into <i>F</i>
12-ring confined benchmark	✓	Accept	Fully contained in allowed set <i>A</i>
<i>Adversarial Trust-Boundary Witnesses</i>			
W1 (legal-but-adjacent)	✓	Reject	Occupies 1-hop buffer halo ($k=1$)
W2 (lexer evasion)	N/A	Reject	Refused by front-end parser

5.2 E2: Validator Practicality

Linear complexity via single-pass fold. After per-device pre-computation, validation takes $O(|C|)$ steps for circuit C under an $O(1)$ edge-membership lookup.² The mechanized step count covers one structural fold over the gates: `checkerSteps` on an n -gate line circuit equals exactly $n+1$ steps (11, 101, 1,001, 10,001, and 40,001 steps for $n = 10$ to 4×10^4), and `checkerSteps_seq` proves additivity under sequential composition (Theorem 4.2). The $O(1)$ graph lookup is an operational assumption. By construction,

²The $O(1)$ edge lookup is a modeling assumption. Edge testing on our encoded device patch executes as a list scan. Region precomputation returns concrete array data (Appendix C.2).

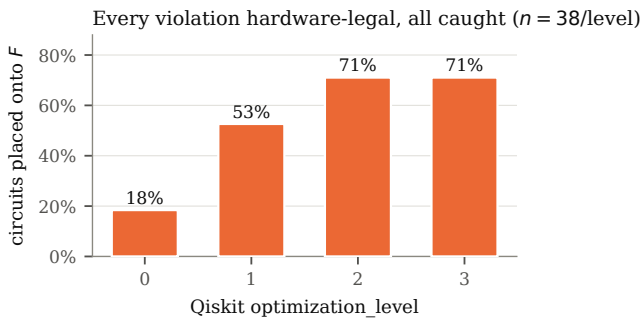


Figure 4: Confinement violation rates for unconstrained Qiskit output across optimization levels 0–3 ($n = 38$ circuits on a 12-node ring). Higher optimization levels increase violations by co-locating gates near tenant boundaries. All outputs remain hardware-coupling compliant.

`checkerSteps` counts the same gate traversals across all 11 topologies. Quantum equivalence checking has $\Theta(2^n)$ worst-case complexity [4]. The stated validation cost is linear in gate count under the lookup assumption.

Validation latency. Lean’s native binary runtime measures the latency of `certifySecurity` (Theorem 4.1). Validation returns a verdict in under 1 ms for 10^3 -gate circuits and low tens of milliseconds for 10^4 -gate circuits, tracking gate count directly without exploring state space. We trust the Lean compiler and runtime environment exclusively for wall-clock performance measurements (§4.1). These timings track the proved $\Theta(\#gates)$ traversal bound.

Memoized buffer overhead. On the 156-qubit `marrakesh` graph, region memoization brings buffered validation latency close to the unbuffered baseline. For a 10^4 -gate circuit, unbuffered validation completes in 6 ms. Naive unmemoized buffering requires 0.9 s at radius $k=1$ and 140 s at $k=2$, matching the predicted $\Theta(\#gates \cdot n^k)$ scaling. Precomputing the forbidden region via `bufferArrK`, with verdict identity proved by `certifySecurity_bufferArrK`, restores baseline latency: memoized buffered validation measures within run-to-run noise of unbuffered validation across all radii ($k \in \{1, 2, 3, 4, 8\}$ at 10^3 gates). Device and region precomputation completes once per topology and amortizes over all subsequent submissions.

Low overhead relative to transpilation. On the E1 benchmark suite, certification completes in under 1 ms against Qiskit `transpile` latencies ranging from 6 to 45 ms, contributing at most 17% and typically under 10% of total compilation time. Native compilation incurs approximately 60 s of one-time startup cost, amortized across subsequent submissions.

5.3 E3: Cross-Device Scalability

Cross-device kernel reduction scaling. We evaluated kernel-checked verification scalability across 11 real IBM QPU topologies, assessing four representative placement verdicts per device. Plain support-only confinement admits placements that are support-disjoint from the co-tenant region but sit exactly one hop away (the physical crosstalk gap). The buffered policy `bufferK` at radius $k=1$ rejects these adjacent placements (the security fix), while admitting placements at graph distances ≥ 2 to avoid over-blocking. Because the rejected placement is fully hardware-legal, the refusal reflects deliberate policy enforcement instead of topology illegality. Lean kernel reduction evaluated all 44 verdicts (`decide`) on device graphs generated from vendor snapshots and independently re-verified from clean builds. The corpus includes 44 kernel-checked verdicts and 22 kernel-checked encoding obligations across the tested graph families (Table 7).

Evaluated device corpus. The evaluated topologies span 7 to 156 qubits across three major architectural families: linear/T, heavy-hex, and degree-4 square lattices. For each device, we seeded the forbidden co-tenant region at the most interior maximum-degree qubit to avoid boundary trivialities. Verification on the degree-4 square lattice shows that the procedure also handles a graph geometry beyond heavy-hex layouts. Kernel reduction across the complete 156-qubit `marrakesh` device graph completes in 4.9 s for the full module (Appendix C.4). We instantiated topologies from Qiskit

provider snapshots, enabling zero-QPU-cost evaluation. These reductions show that the decision procedure runs in the Lean kernel on the tested production topologies.

5.4 E4: Physical Motivation

Hardware campaign summary. Lean proves the certificate sound for the structural policy (§4). Separately, across 23 physical pair-observations, the policy missed no nearest-neighbor crosstalk detected at the campaign threshold.

Experimental methodology and multi-tenant emulation. Current public cloud QPUs (e.g., IBM Quantum) operate under single-tenant execution queues where one user job occupies the QPU at a given time [12]. Following established multi-tenant quantum security methodology [18, 35], we emulate multi-tenancy within a single job execution: victim and probe circuits occupy disjoint qubit allocations and execute no shared gates, isolating the target physical coupling channel. Disjoint-region parallel execution on current IBM hardware uses this same single-job interface and leaves a default gap of two hops between partitions [5]. This protocol models the physical interaction between adjacent spatial allocations, as always-on capacitive qubit coupling operates independently of job billing boundaries.

Our primary evaluation targeted **ibm_marrakesh** (IBM Heron r2, 156 qubits) with pinned calibration parameters. We placed the victim circuit on qubit q98 and probe circuits at graph distances 1, 2, 3, and 18. Graph distance corresponds to the radius expanded by **bufferF** (§2). Each measurement executed a Y-basis Ramsey experiment with an idle duration $\tau = 40 \mu\text{s}$ and $2 \times 8,192$ total shots, consuming approximately 80 QPU-seconds. We computed the observable state perturbation $\Delta P = |P_1(1) - P_0(1)|$ alongside an unpooled two-proportion z -statistic. The pre-registered leak criterion ($z \geq 5$, corresponding to $\Delta P \approx 0.025$) exceeds the Bonferroni multiplicity threshold for 46 distance tests ($|z| > 3.1$ at $\alpha = 0.05$).

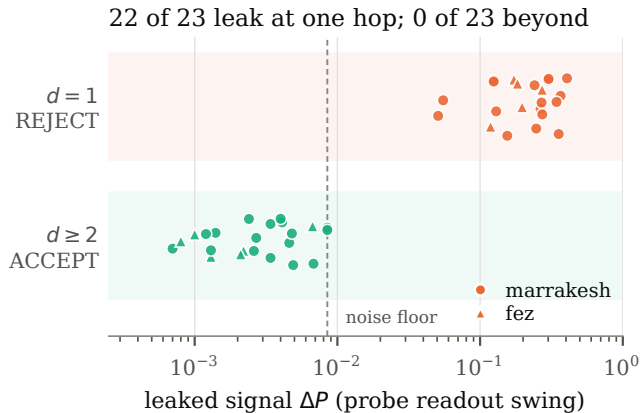


Figure 5: Measured state probability shift ΔP across physical qubit pairs grouped by graph distance d . Shaded regions indicate policy verdicts produced by the **bufferF** certificate from coupling graph topology alone. The pre-registered leak threshold ($z \geq 5$) separates detected distance-1 effects from distance $d \geq 2$ nulls in this sample. Emulated co-tenancy executes disjoint qubit allocations within single jobs (§5.4).

Table 4: Physical QPU evaluation campaign breakdown across device calibrations. Each observation measures one victim-probe pair at $d=1$ and $d \geq 2$ within a single job execution. We classify leaks by the pre-registered threshold $z \geq 5$. Detected distance-1 effects range from $z = 9.5$ to 82.3, while measured distance $d \geq 2$ nulls satisfy $|z| \leq 1.8$.

Cal.	Evaluation Scope	Obs.	Physical Leaks		Shift $\Delta P (d=1)$
			$d=1$	$d \geq 2$	
1	Baseline (8 pairs, 2 QPUs)	8	7 / 8	0 / 8	0.000–0.366
2	Recalibrated (same 8 pairs)	8	8 / 8	0 / 8	0.118–0.407
3	Stability (+18 h post-cal)	2	2 / 2	0 / 2	0.273–0.355
4	Generalization (5 fresh pairs)	5	5 / 5	0 / 5	0.055–0.302
Total	Summary (13 distinct pairs)	23	22 / 23	0 / 23	0.000–0.407

Across all measurements, observed leaks produced $z = 9.5$ –82.3 at $d=1$ (with the primary q98 run reaching $z = 86$), whereas all null observations satisfied $|z| \leq 1.8$ at $d \geq 2$.

Measured nearest-neighbor crosstalk channel. On live IBM Heron r2 hardware, adjacent probes distinguish the victim state, with probability shifts up to $\Delta P = 0.407$. At tested graph distances $d \geq 2$, signals remain below the measurement detection floor. In the primary run, the 1-hop probe exhibited a strong physical leak ($\Delta P = 0.397$, $z = 86$), while probes at distances 2, 3, and 18 showed no detectable signal ($\Delta P \leq 0.0031$). On the encoded device graph, the validator’s **bufferF** policy rejects placement at $d=1$ and accepts at $d \geq 2$ via kernel-reduced **decide**, aligning with the physical signal boundary (Figure 5).

Attacker shot cost analysis. We estimate the shot cost of distinguishing the two prepared victim states from the measured perturbation ΔP . Under the observed probabilities and a 5σ criterion, the estimate is $n = 25 (p_0(1-p_0) + p_1(1-p_1)) / \Delta P^2$. Across all evaluated snapshots, 15 of 16 distance-1 pairs exhibited state leakage, yielding estimates of 60 to 4,526 shots (median 186). At distance $d=2$, the corresponding estimates range from 1.3×10^5 to 1.9×10^7 shots. These estimates place $d=2$ three to five orders of magnitude above $d=1$ for the measured pairs and calibrations. They do not report a blind bit-recovery experiment.

Static and pure-drive controls support always-on ZZ coupling as the measured mechanism. Holding the victim in a static state reproduced the leak ($\Delta P = 0.380$, $z = 82$), whereas returning the victim to $|0\rangle$ reduced the signal below the detection threshold ($\Delta P = 0.006$, $z = 1.6$). The leakage mechanism occurs without active gate driving on the victim and depends on secret state retention.

Multi-pair and multi-calibration confirmation. A pre-registered sweep across 8 qubit pairs on two IBM Heron r2 devices (154 QPU-seconds) reproduced the $d=1$ leak on 7 of 8 pairs ($z = 9.5$ –77, $\Delta P = 0.05$ –0.37), with no detected leaks at $d \geq 2$ (0/8). We fixed pair selection deterministically from baseline calibration data (T_2 and readout error) prior to execution. Within this sweep and detection threshold, no accepted placement exhibited a detected leak (0/8 false accepts).

Extending the campaign across 4 calibration cycles and 13 distinct qubit pairs yielded 23 total pair-observations (Table 4). At the stated detection threshold, none of the $d \geq 2$ measurements registered a leak (0/23). Re-testing the 8 pre-registered pairs after full hardware recalibration (where all 24 qubits reported modified readout error parameters) found $d=1$ leakage on all 8 pairs (8/8) and no detected leakage at $d \geq 2$ (0/8).

Structural certification vs. calibration drift. Physical leakage magnitude varied across calibration cycles while graph distance boundaries remained invariant. For one evaluated pair, recalibration shifted the measured signal from a quiet baseline ($\Delta P \approx 0$) to a strong leak ($\Delta P = 0.118$, $z = 22.4$). Across all 13 pairs, detected distance-1 leakage magnitudes ranged from $\Delta P = 0.055$ to 0.407 , while one distance-1 observation fell below the detection threshold (Figure 6). All $d \geq 2$ observations remained below that threshold. The pair that changed from a null to a leak shows why a whitelist based on one calibration snapshot can fail after recalibration. Graph-based certification retains the same structural verdict when measured magnitudes drift.

Job provenance and reconciliation. The primary hardware campaign targets `ibm_marrakesh` (IBM Heron r2). The artifact pins all calibration snapshots across Heron r2 devices alongside qubit identifiers and the pre-registered pair-selection routine. The campaign records every job with its identifier, backend, submission time, shot count (1,835,008 total shots across pre-registered campaign jobs), circuit count, and raw per-circuit bitstring counts. From these raw counts, all 16 published pair-observations in the sweep and repeat snapshots recompute to within 5×10^{-4} on ΔP and 0.1 on z . The submission routine fixes per-circuit ordering deterministically (pair-major, then distance, victim bit, and repetition), rendering reconciliation independent of any separate label map.

Pre-registration and cancelled arms. To enforce pre-registration integrity and avoid reporting bias, we account for all submitted QPU jobs. We submitted and cancelled two jobs before they returned results: one was a third Heron r2

device (`ibm_kingston`), pre-registered with 5 pairs under the outcome-independent selection rule and submitted with 40 circuits at 8,192 shots (cancelled in queue). The second was a 16-circuit `ibm_marrakesh` job submitted later. Reporting these ensures full pre-registration transparency. The campaign data thus cover two Heron r2 devices with data, while we attempted a third and obtained no data.

Empirical scope of null observations. The distance $d \geq 2$ readings remain nulls at our evaluated experimental sensitivity ($\tau = 40 \mu s$, $2 \times 8,192$ shots, $z \geq 5$). The observed calibration drift shows that measurement sensitivity limits single-snapshot nulls. Reported values state direct observable shifts ΔP without quoting a fitted coupling strength ζ , as three distinct ζ values fit $\Delta P = |\sin(2\pi\zeta\tau)|$ under fixed τ . The security argument uses topological placement structure and does not require a fitted ζ . A forward master-equation simulation from stored calibration snapshots with injected ZZ coupling reproduces all 13 pair observations within $\Delta P \leq 0.011$, serving as our artifact regression harness.

5.5 E5: Capacity Trade-offs & Operability

Buffer radius and capacity trade-offs. Reserving a k -hop buffer trades usable chip area for confinement margin. We measured this capacity trade-off on an encoded 12-qubit patch of the `ibm_marrakesh` graph seeded with a single-qubit co-tenant region. Selecting $k=1$ sterilizes 4 qubits, leaving 67% of chip area usable (8 of 12 qubits). Increasing the buffer radius to $k=2$ sterilizes 7 qubits (42% usable), $k=3$ sterilizes 9 qubits (25% usable), and $k=4$ sterilizes 11 qubits (8% usable). Among the tested radii, $k=1$ removes shared coupling edges while reserving the fewest qubits. We specified $k=1$ a priori based on the physical threat model prior to hardware measurement. The lack of detected leakage at $d \geq 2$ supports that choice for the tested pairs and sensitivity (§6).

Public-metadata policy operability. To evaluate whether a tenant can verify buffered gate-support confinement policy compliance independently, we re-scored all 46 distance verdicts (23 pair-observations at two distances) using exclusively public vendor metadata. The evaluation predictor accessed only published coupling maps, with mechanical file-access guards enforcing separation from QPU measurement data. Published vendor metadata omit physical parameters (qubit frequencies, anharmonicity, coupling strengths, and coupler bias values). Public coupling maps alone cannot predict leakage magnitudes. Magnitude-dependent security policies therefore require privileged hardware access. Our structural policy uses public graph connectivity: it rejects a probe placement at graph distance 1 from a co-tenant. Evaluated blind against published coupling maps, the structural policy achieved 22 true positives, 23 true negatives, 1 conservative false positive, 0 false negatives, and no missed leaks detected at the stated threshold among the 46 verdicts. The single conservative false positive corresponds to the calibration-sensitive pair that measured quiet in snapshot 1 but leaked at $z = 22.4$ in snapshot 2. These results show that tenants can compute the tested structural policy verdicts from published device topologies without internal hardware telemetry.

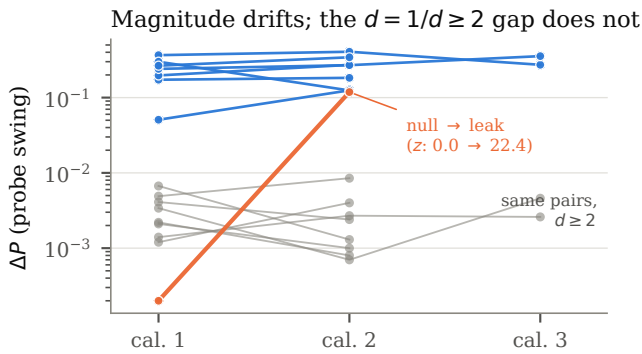


Figure 6: Pre-registered pairs re-measured across three calibrations (upper: $d=1$, lower grey: same pairs at $d \geq 2$). Detected magnitudes change by up to $2\times$ within re-measured pairs. One distance-1 pair crosses from undetectable to a strong leak. Every $d \geq 2$ observation remains below the detection threshold. The certificate checks placement. Leakage magnitude remains empirical.

6 Discussion and Limitations

Translation validation places spatial confinement enforcement in a verified client-side certificate checker. Security guarantees for certified artifacts in deployment also depend on platform conditions, policy boundaries, and hardware stability.

6.1 Platform Dependencies and Assumptions

Certificates provide security guarantees in deployment under four operational conditions (Table 5, Appendix B).

Returned-artifact execution. Validation certifies a specific compiled circuit. Detecting artifact substitution between validation and dispatch requires physical hardware attestation, an operational requirement beyond static translation checking.

Honest allocation boundaries. When the tenant checks the complement of its grant, an overstated A_{grant} shrinks the forbidden input and can permit unsafe gate placement. The certificate therefore depends on an accurate allocation grant authenticated by the provider. If the tenant knows only its grant, the provider must also reserve the buffer around co-tenants. The certificate checks confinement to the grant. It cannot verify the provider’s reservation.

Coupling topology integrity. Precomputing graph-buffered regions depends on provider coupling maps. If a map omits a physical edge, the resulting buffer may admit placements adjacent to F . Kernel reduction remains sound over the supplied graph term (§4.1), but unverified topology transcription can weaken physical confinement.

Provider-wide enforcement. Client-side validation prevents a victim’s transpiler from routing operations into adjacent allocations. Multi-tenant isolation requires providers to mandate validation for all co-tenants over mutually disjoint regions.

6.2 Policy Boundaries and Channel Scope

Translation validation formalizes physical gate-support confinement over declared coupling graphs. Operational security also depends on channels outside the certified gate-residency invariant.

Functional integrity and covert channels. Client-side checking certifies spatial gate placement without asserting circuit equivalence, which is QMA-complete [13]. Gate-support validation establishes physical placement. Information-theoretic non-interference and covert timing and coherence channels lie outside the model. Pulse-level attacks [41] can bypass gate abstractions on open-pulse hardware, so providers must restrict submissions to fixed, calibrated gate sets (§5.4).

Side-channel vector coverage. The four vulnerability vectors introduced in §2 span different physical and classical channels. Gate-support validation blocks SWAP operations into forbidden qubits (V1). A buffered allocation separates retained states from the graph-adjacent ZZ channel measured for V4, while validation prevents gates from crossing that boundary (§5.4). Other channels fall outside graph policies: readout crosstalk (V2) across shared feedlines requires hyper-edge topology extensions, while classical inverse transpilation (V3) extracts algorithmic patterns independently of physical co-location.

Buffer radii and physical state residency. The tested single-hop buffer ($k=1$) separates states from the measured nearest-neighbor channel under the observed conditions (§5.4).

Non-local interactions [2] may require larger radii k or extended coupling graphs. Gate-support validation prevents operations outside the allowed region A . Provider-enforced reservation of $A_{\text{grant}} \subseteq A$ governs idle state placement.

6.3 Formal Verification and Empirical Generalization

Mechanized kernel proofs and empirical hardware measurements support different parts of the validation claim.

Proof scope and kernel scaling. Construction-mode proofs (§4.4) cover topologies such as 12-node rings. Translation validation checks supported transpiler outputs represented in the circuit IR. Lean kernel reduction checks certificates on 11 IBM graphs (7–156 qubits) in under 5 seconds (§5.3), showing that client-side checking scales without verifying the whole compiler.

Empirical hardware bounds. The hardware campaign recorded 23 observations on 13 qubit pairs across two IBM Heron r2 QPUs over 33 hours (§5.4). Repeated measurements of the same pairs changed by up to $2\times$ across calibrations. One pair changed from no detected leak to $\Delta P = 0.118$. Across distinct pairs, distance-1 magnitudes ranged from 0.055 to 0.407 (§5.4). The checker certifies structural placement without bounding leakage magnitude. These measurements do not establish the same behavior on other device architectures.

7 Related Work

Translation validation and proof-carrying code. CompCert [20] verifies a compiler once. Translation validation [29, 31, 32, 39] checks each emitted artifact, catching producer errors without trusting the producer. Proof-Carrying Code [28] uses a consumer-defined policy and a small checker. ARMor [42] offers the closest analogue to our trust granularity: it constructs a fresh HOL proof for each rewritten ARM binary. Our certificate follows this per-artifact model. The mechanized decision procedure itself serves as the checker, and a kernel-checked theorem relates its Boolean output to the Prop-level policy (§4.1). Following the de Bruijn criterion [3] and verified UNSAT-proof checking [17], we rely on a small auditable checker.

Post-compilation confinement validation. VeriWasm [14] and RockSalt [27] also use an untrusted producer, per-instance checks, isolation policies, and rejection on failure. Our validator is the mechanized function itself, linked to policy specifications by kernel-checked soundness proofs. Prior systems verify a model of the checker. Their policies enforce memory address ranges using inserted guards. Our certificate enforces spatial separation on physical coupling graphs, where adjacency introduces crosstalk.

Secure compilation from a trusted source. OwlC [34], Jasmin [1], and CT-Wasm [40] establish security guarantees by construction. This approach requires trust in the compiler, which tenants using proprietary cloud transpilers cannot assume.

Verified quantum compilation. VOQC/SQIR [10] verifies quantum optimizations and includes a translation validator for circuit mapping. Giallar [36] and CertiQ [33] verify Qiskit passes, while QCEC [4] decides arbitrary circuit equivalence. These systems do not state or verify structural tenant policies such as buffered confinement. VOQC’s validator emits a Boolean from

extracted OCaml. Our design generates proof terms that the tenant’s Lean kernel re-checks. Because general circuit equivalence is QMA-complete [13], our tenant-side check targets structural placement policies.

Quantum multi-tenancy and crosstalk defenses. Prior multi-programming proposals establish buffering policies [6, 8, 15, 25, 26, 30, 37]. These systems trust the provider transpiler to honor mapping constraints [22, 23] or run checks on the provider side [11]. Our approach removes trust in provider transpilation by generating kernel-checked confinement certificates for supported outputs of arbitrary transpilers. A tenant can check its granted allocation by forbidding its complement, while the provider remains responsible for assigning a buffered grant (§2). Documented SWAP channels [18] and readout crosstalk [35] motivate this policy.

Foundational quantum verification. CoqQ [43] and VyZX [19] provide foundational frameworks for quantum programs. They do not return per-artifact verdicts on circuits from untrusted transpilers.

8 Conclusion

A tenant can check, in one linear pass, a kernel-checked certificate that a compiled circuit uses device wiring and keeps non-identity gate support outside co-tenant qubits and their immediate graph neighbors. The transpiler remains outside the trusted base, as does front-end parsing under a tested reject-safety invariant. Across 23 physical QPU observations, the one-hop boundary accepted no placement that leaked at our detection threshold. The certificate guarantees gate-support confinement under the stated allocation and execution-binding assumptions. Its region-parametric soundness theorem also applies to larger radii and other forbidden regions without a separate proof for each region.

Ethical Considerations

All hardware measurements ran within our allocated execution jobs. IBM Quantum schedules one job per QPU at a time [12], so our experiments did not observe, probe, or alter external tenant computations. All procedures complied with platform terms of use. Prior multi-tenant crosstalk work motivates neighbor isolation on shared QPUs [18, 26, 35]. We measured the always-on ZZ state-readout channel reported here in the paper’s controls (§5.4). Exploiting this channel would require multi-tenant co-location policies that current commercial platforms do not deploy. The artifact uses standard, documented Ramsey measurement protocols.

Open Artifact & Code Availability

The public repository at <https://github.com/alirafiqmalik/certified-confinement-acm-pqq-26-artifacts> contains the Lean 4 development, OpenQASM front end, measurement and blind-prediction scripts, recovered job provenance, and calibration snapshots. It includes committed result logs for the hardware, device-corpus, and buffered-cost numbers, along with scripts to recompute them. The build is axiom-clean and contains no **sorry**, **native_decide**, or **ofReduceBool**. Reproducing the E1 transpiler-violation rates also requires the external QASMBench suite and a Qiskit install.

References

- [1] José Bachelar Almeida, Manuel Barbosa, Gilles Barthe, Arthur Blot, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira, Hugo Pacheco, Benedikt Schmidt, and Pierre-Yves Strub. 2017. Jasmin: High-Assurance and High-Speed Cryptography. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, New York, NY, USA, 1807–1823. doi:10.1145/3133956.3134078
- [2] Evan J. D. Anderson, Kaushik Datta, and Boulal A. Bash. 2026. Toward Covert Quantum Computing. arXiv:2605.14325 [quant-ph] <https://arxiv.org/abs/2605.14325>
- [3] Henk Barendregt and Herman Geuvers. 2001. Proof-Assistants Using Dependent Type Systems. In *Handbook of Automated Reasoning*, Alan Robinson and Andrei Voronkov (Eds.). Vol. 2. Elsevier and MIT Press, Amsterdam and Cambridge, MA, Chapter 18, 1149–1238. doi:10.1016/B978-044450813-3/50020-5
- [4] Lukas Burgholzer and Robert Wille. 2021. Advanced Equivalence Checking for Quantum Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 9 (Sept. 2021), 1810–1824. doi:10.1109/tcad.2020.3032630
- [5] Avimita Chatterjee, W. Michael Brown, Siyuan Niu, Wibe Albert de Jong, and Thomas Lubinski. 2026. Parallel Circuit Execution for Scalable Quantum Computation. arXiv:2609.12172 [quant-ph] <https://arxiv.org/abs/2609.12172>
- [6] Poulami Das, Swamit S. Tannu, Prashant J. Nair, and Moinuddin Qureshi. 2019. A Case for Multi-Programming Quantum Computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. Association for Computing Machinery, New York, NY, USA, 291–303. doi:10.1145/3352460.3358287
- [7] Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction (CADE 28) (LNCS, Vol. 12699)*. Springer International Publishing, Cham, 625–635. doi:10.1007/978-3-030-79876-5_37
- [8] Emmanouil Giortamis, Francisco Romão, Nathaniel Tornow, and Pramod Bhatotia. 2025. QOS: A Quantum Operating System. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*. USENIX Association, 429–447. <https://www.usenix.org/conference/osdi25/presentation/giortamis>
- [9] Frank Haverkamp, Juergen Saalmueller, Markus Buehler, Thilo Maurer, Tristan Müller, and Jeffrey Joseph Ruedinger. 2026. Multitasking Scheme for Quantum Computers. U.S. Patent 12,541,703. <https://patents.google.com/patent/US12541703B2/en>
- [10] Kesha Hietala, Robert Rand, Liyi Li, Shih-Han Hung, Xiaodi Wu, and Michael Hicks. 2023. A Verified Optimizer for Quantum Circuits. *ACM Transactions on Programming Languages and Systems* 45, 3, Article 18 (2023), 35 pages. doi:10.1145/3604630
- [11] Barbara Hrd’a, Sanjay Deshpande, Theodoros Trochatos, and Jakub Szefer. 2025. CHEQ: Towards Enabling Circuit Integrity Checking in Quantum Controllers. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*. Association for Computing Machinery, New York, NY, USA, 265–272. doi:10.1145/3716368.3735296
- [12] IBM Quantum. 2026. Choose the Right Execution Mode. IBM Quantum Documentation. <https://quantum.cloud.ibm.com/docs/en/guides/choose-execution-mode>
- [13] Dominik Janzing, Pawel Wocjan, and Thomas Beth. 2005. “Non-Identity-Check” Is QMA-Complete. *International Journal of Quantum Information* 3, 3 (2005), 463–473. doi:10.1142/S0219749905001067
- [14] Evan Johnson, David Thien, Yousef Alhessi, Shriram Narayan, Fraser Brown, Sorin Lerner, Tyler McMullen, Stefan Savage, and Deian Stefan. 2021. Doveryai, no Proveryai: SFI Safety for Native-Compiled Wasm. In *Proceedings of the 2021 Network and Distributed System Security Symposium (NDSS)*. Internet Society. <https://www.ndss-symposium.org/ndss-paper/doveryai-no-proveryai-sfi-safety-for-native-compiled-wasm/>
- [15] Avinash Kumar, Meng Wang, Chenxu Liu, Ang Li, Prashant J. Nair, and Poulami Das. 2025. Context Switching for Secure Multi-programming of Near-Term Quantum Computers. arXiv:2504.07048 [cs.CR] <https://arxiv.org/abs/2504.07048>
- [16] Satwik Kundu and Swaroop Ghosh. 2025. Inverse-Transpilation: Reverse-Engineering Quantum Compiler Optimization Passes from Circuit Snapshots. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*. Association for Computing Machinery, New York, NY, USA, 273–277. doi:10.1145/3716368.3735298
- [17] Peter Lammich. 2020. Efficient Verified (UN)SAT Certificate Checking. *Journal of Automated Reasoning* 64, 3 (2020), 513–532. doi:10.1007/s10817-019-09525-z
- [18] Wei Jie Bryan Lee, Siyi Wang, Suman Dutta, Walid El Maouaki, and Anupam Chattopadhyay. 2025. SWAP Attack: Stealthy Side-Channel Attack on Multi-Tenant Quantum Cloud System. arXiv:2502.10115 [quant-ph] <https://arxiv.org/abs/2502.10115>
- [19] Adrian Lehmann, Ben Caldwell, Bhakti Shah, William Spencer, and Robert Rand. 2026. VyZX: Formal Verification of a Graphical Quantum Language. *ACM Transactions on Programming Languages and Systems* 48, 3, Article 11 (2026), 39 pages. doi:10.1145/3807780
- [20] Xavier Leroy. 2009. Formal Verification of a Realistic Compiler. *Commun. ACM* 52, 7 (2009), 107–115. doi:10.1145/1538788.1538814

- [21] Ang Li, Samuel Stein, Sriram Krishnamoorthy, and James Ang. 2023. QASMBench: A Low-Level Quantum Benchmark Suite for NISQ Evaluation and Simulation. *ACM Transactions on Quantum Computing* 4, 2, Article 10 (2023), 26 pages. doi:10.1145/3550488
- [22] Lei Liu and Xinglei Dou. 2024. QuCloud+: A Holistic Qubit Mapping Scheme for Single/Multi-programming on 2D/3D NISQ Quantum Computers. *ACM Transactions on Architecture and Code Optimization* 21, 1, Article 9 (2024), 27 pages. doi:10.1145/3631525
- [23] Shusen Liu, Pascal Jahan Elahi, and Ugo Varetto. 2026. DynQ: A Dynamic Topology-Agnostic Quantum Virtual Machine via Quality-Weighted Community Detection. arXiv:2601.19635 [quant-ph] <https://arxiv.org/abs/2601.19635>
- [24] Chao Lu, Esha Telang, Aydin Aysu, and Kanad Basu. 2025. Quantum Leak: Timing Side-Channel Attacks on Cloud-Based Quantum Services. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*. Association for Computing Machinery, New York, NY, USA, 252–257. doi:10.1145/3716368.3735264
- [25] Yoann Marquer and Domenico Bianculli. 2025. A Piece of QAICCC: Towards a Countermeasure Against Crosstalk Attacks in Quantum Servers. arXiv:2504.10193 [cs.SE] <https://arxiv.org/abs/2504.10193>
- [26] Devika Mehra and Amir Kalev. 2026. Towards Defending Crosstalk-Mediated Attacks in Multi-Tenant Quantum Computing. *Physica Scripta* 101, 9, Article 095102 (2026). doi:10.1088/1402-4896/ae4429
- [27] Greg Morrisett, Gang Tan, Joseph Tassarotti, Jean-Baptiste Tristan, and Edward Gan. 2012. RockSalt: Better, Faster, Stronger SFI for the x86. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. Association for Computing Machinery, New York, NY, USA, 395–404. doi:10.1145/2254064.2254111
- [28] George C. Necula. 1997. Proof-Carrying Code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. Association for Computing Machinery, New York, NY, USA, 106–119. doi:10.1145/263699.263712
- [29] George C. Necula. 2000. Translation Validation for an Optimizing Compiler. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. Association for Computing Machinery, New York, NY, USA, 83–94. doi:10.1145/349299.349314
- [30] Yasuhiro Ohkura, Takahiko Satoh, and Rodney Van Meter. 2022. Simultaneous Execution of Quantum Circuits on Current and Near-Future NISQ Systems. *IEEE Transactions on Quantum Engineering* 3, Article 2500210 (2022), 10 pages. doi:10.1109/TQE.2022.3164716
- [31] Amir Pnueli, Michael Siegel, and Eli Singerman. 1998. Translation Validation. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (LNCS, Vol. 1384)*. Springer-Verlag, Berlin, Heidelberg, 151–166. doi:10.1007/BFb0054170
- [32] Silvain Rideau and Xavier Leroy. 2010. Validating Register Allocation and Spilling. In *Compiler Construction (CC) (LNCS, Vol. 6011)*. Springer-Verlag, Berlin, Heidelberg, 224–243. doi:10.1007/978-3-642-11970-5_13
- [33] Yunong Shi, Runzhou Tao, Xupeng Li, Ali Javadi-Abhari, Andrew W. Cross, Frederic T. Chong, and Ronghui Gu. 2020. CertiQ: A Mostly-automated Verification of a Realistic Quantum Compiler. arXiv:1908.08963 [quant-ph] <https://arxiv.org/abs/1908.08963>
- [34] Pratap Singh, Joshua Gancher, and Bryan Parno. 2025. OwlC: Compiling Security Protocols to Verified, Secure, High-Performance Libraries. In *34th USENIX Security Symposium (USENIX Security 25)*. USENIX Association, Seattle, WA, 5071–5090. <https://www.usenix.org/conference/usenixsecurity25/presentation/singh>
- [35] Yizhuo Tan and Jakub Szefer. 2025. I Know What You Are Reading: Evaluating Readout Crosstalk in Cloud-based Quantum Computers. In *Proceedings of the 2025 Quantum Security and Privacy Workshop*. Association for Computing Machinery, New York, NY, USA, 10–15. doi:10.1145/3733825.3765280
- [36] Runzhou Tao, Yunong Shi, Jianan Yao, Xupeng Li, Ali Javadi-Abhari, Andrew W. Cross, Frederic T. Chong, and Ronghui Gu. 2022. Giallar: Push-Button Verification for the Qiskit Quantum Compiler. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI)*. Association for Computing Machinery, New York, NY, USA, 641–656. doi:10.1145/3519939.3523431
- [37] Runzhou Tao, Hongzheng Zhu, Jason Nieh, Jianan Yao, and Ronghui Gu. 2025. Quantum Virtual Machines. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*. USENIX Association, Boston, MA, 411–428. <https://www.usenix.org/conference/osdi25/presentation/tao>
- [38] The mathlib Community. 2020. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*. Association for Computing Machinery, New York, NY, USA, 367–381. doi:10.1145/3372885.3373824
- [39] Jean-Baptiste Tristan and Xavier Leroy. 2008. Formal Verification of Translation Validators: A Case Study on Instruction Scheduling Optimizations. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. Association for Computing Machinery, New York, NY, USA, 17–27. doi:10.1145/1328438.1328444
- [40] Conrad Watt, John Renner, Natalie Popescu, Sunjay Cauligi, and Deian Stefan. 2019. CT-Wasm: Type-Driven Secure Cryptography for the Web Ecosystem. *Proceedings of the ACM on Programming Languages* 3, POPL, Article 77 (2019), 29 pages. doi:10.1145/3290390
- [41] Chuanki Xu and Jakub Szefer. 2025. Security Attacks Abusing Pulse-level Quantum Circuits. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE Computer Society, 222–239. doi:10.1109/SP61157.2025.00083
- [42] Lu Zhao, Guodong Li, Bjorn De Sutter, and John Regehr. 2011. ARMor: Fully Verified Software Fault Isolation. In *Proceedings of the 9th ACM International Conference on Embedded Software (EMSOFT)*. Association for Computing Machinery, New York, NY, USA, 289–298. doi:10.1145/2038642.2038687
- [43] Li Zhou, Gilles Barthe, Pierre-Yves Strub, Junyi Liu, and Mingsheng Ying. 2023. CoqQ: Foundational Verification of Quantum Programs. *Proceedings of the ACM on Programming Languages* 7, POPL, Article 29 (2023), 33 pages. doi:10.1145/3571222

A Glossary

Qubit

The quantum counterpart of a bit and the basic unit manipulated by a quantum computer.

Quantum cloud / multi-tenant

Access to a provider’s quantum computer over the internet, potentially shared with other tenants at the same time (“multi-tenant”).

Transpiler (quantum compiler)

Software that rewrites a quantum program to fit a particular machine’s fixed wiring.

Routing / SWAP

Moving qubits around the chip with SWAP operations so that qubits that must interact become physically adjacent.

Coupling graph / hardware-legal

A map of the qubits physically wired to interact. A program is *hardware-legal* if it makes only wired qubits interact.

Global phase

A difference between two quantum programs that no measurement can detect. Programs “equal up to global phase” are physically identical.

Side channel / crosstalk

Information leakage through physical effects between neighboring qubits on shared hardware, outside a program’s output.

Confinement / isolation

Keeping a tenant’s secret qubits within an allowed region of the chip, away from regions controlled by other tenants.

Formal verification / proof assistant

Software that checks mathematical proofs. If its trusted core (the “kernel”) accepts a proof, the statement follows from the formal definitions and assumptions.

Certificate / translation validation

A proof attached to a tool’s output that a checker can verify without trusting the tool.

The exponential (2^n) blow-up

Representing a quantum computer’s state requires a table that doubles with each added qubit, making brute-force checking infeasible. For routing, we track qubit *positions* instead of the full state.

B Threat Model Contract

Table 5 records the trust assignments used throughout the paper. Its final row states a system-wide requirement beyond any single tenant’s certificate (§6.1).

Table 5: Threat-model contract and system-wide security boundary assignments. The final row states an operational deployment requirement beyond a single tenant’s certificate (§6.1).

System Component	Trust Status	Operational Role & Security Boundary
Provider transpiler	Untrusted	Emits candidate compiled circuit c (transpiler outside the TCB)
OpenQASM front end	Tested	Converts string input to circuit IR and passes reject-safety tests (unexpected tokens cause parse failure)
Lean 4 kernel, IR & spec	Trusted Core	Evaluates HWF , confinedb , and bufferK . The kernel checks the structural decision
Device coupling map	Assumed Ground Truth	Vendor-published graph defining physical edges and topological distance
Topology encoder	Trusted Encoder	Transcribes a published graph snapshot into a Coupling n term that the kernel reduces
Tenant grant A_{grant}	Allocation Input	Cloud allocator provides it to the tenant. Buffered deployment requires $A_{\text{grant}} \subseteq A$
Artifact binding at execution	Deployment Contract	Provider must execute the checked circuit c without post-validation mutation
Co-tenant isolation	System Requirement	Co-tenants must operate under disjoint, verified buffered allocations throughout the cloud

C Mechanization Notes

This appendix details the mechanization behind §4: theorem axioms, lemmas for the two construction-mode results, front-end mechanics, and kernel-reduction constraints. The artifact contains the circuit IR and its 2^n denotation, equivalence up to global phase, the hardware model, symbolic routing, optimization, confinement, certificates, the device corpus, and the front end.

C.1 Trust base and axiom provenance

Table 6 reports `#print` axioms for each headline theorem. The security driver `certifySecurity_sound` uses only `propext`. Step-count additivity uses no axioms. The driver establishes accepted $\Rightarrow$ legal $\wedge$ `confinedb`. The set-level conclusion (“the support of c contains no qubit of F ”) also uses only `propext`, through `confinedb_avoids`. The construction-mode path and buffer-precomputation identity use Lean’s three standard axioms (`propext`, `Classical.choice`, `Quot.sound`). The device corpus uses `propext` and `Quot.sound`. The development contains no `sorry`, `native_decide`, `ofReduceBool`, or ad-hoc axiom.

The trusted base comprises the kernel, policy specifications **HWF**, **confinedb**, and **bufferK**, circuit IR gate-support definitions, and the **Decidable** instances used by the verdict. It also includes the unproved encoder that transcribes a published topology snapshot into a **Coupling n** term (§C.4) and `parseQASMSafe`’s reject-safety, established by testing (§C.3). Construction-mode equivalence also trusts **CongPhase** and the 2^n denotation. The security path does not call the denotation. The provider transpiler and lexer translation remain untrusted. The reject-safety invariant confines lexer failures to safe rejection.

C.2 Buffer precomputation and a negative result

The naive buffered check costs $\Theta(q^k)$ per region-membership query. Precomputing the region (`bufferArr/bufferArrK`) restores the linear per-circuit bound. `certifySecurity_bufferMemo` and `certifySecurity_bufferArrK` prove that the precomputed and naive forms return identical verdicts.

Precomputation must return the table *as data*. A recursive function returning a lookup closure rebuilds it per query and runs more slowly than the unmemoized check ($k=4$ did not finish in 120 s).

C.3 Front-end mechanics and fuzz corpus

Lexer coverage. `parseQASMSafe` covers the basis $\{x, sx, id, rz, cz/cx\}$ and feeds `ofList`. It uses placeholders for irrational **rz** angles because the security verdict depends on gate *support*. From QASM strings, it accepts a confined demonstration circuit and rejects a violating one.

Unsound predecessor parser. The original `parseQASM` silently skipped unrecognized tokens. Skipping a support-bearing gate can make the validator accept a circuit it should reject. `parseQASMSafe` rejects every unrecognized support-bearing token. This failure arose in the untrusted front end. `certifySecurity_sound` remains sound for the circuit supplied to it.

Fuzz corpus structure. Four mutation operators applied to 28 base circuits produced 1120 mutants, of which 721 have distinct content. All 269 unsound accepts by the old lexer came from the classically-conditioned-gate operator. The other three produced none. The corpus generates inputs for rejection, so it cannot assess *over-rejection*. Real-transpiler runs address that question (§5.1). The fuzzer found one soundness bug in the untrusted front end, and the fix removed it from every generated mutant. These results do not establish broad coverage of lexer inputs.

C.4 Kernel reduction and device-graph encoding

decide versus #eval. The security path uses one Boolean decision procedure in three roles (§4.1). Theorem 4.1 reasons about it. `decide` reduces device-corpus and circuit-level witnesses. The wall-clock harness executes compiled code. Only the first two enter the certificate’s trusted base. §5.2 labels timings that trust the Lean compiler.

Non-reducing string literals. Lean’s kernel cannot reduce string literals. Witness W2 (§5.1) therefore has kernel-checked circuit-level endpoints but an executable (`#eval`) string-to-gate-list link. The limit arises from kernel reduction over strings, independent of input size. W2 is not kernel-checked end to end.

Well-founded recursion in the optimizer. A separate kernel-reduction limit affects optimizer witnesses. `optFix` uses well-founded recursion to iterate `optAdj` to a fixed point, which `decide` does not reduce. Thus, rejection of an *optimizer output* is an executable (`#eval`) witness. The corresponding *input* circuit is kernel-checked. This applies to the optimizer row of Table 3 and the

Table 6: Trust base and formal axiom provenance (#print axioms). “std 3” denotes Lean 4 standard foundational axioms: `propext`, `Classical.choice`, `Quot.sound`.

Theorem / Artifact Component	Formally Proven Guarantee	Minimal Axiom Profile
<code>certifySecurity_sound</code>	Accepted $\Rightarrow$ Hardware legal $\wedge$ Confined (<i>arbitrary output</i>)	propext only
<code>certifySecurity_bufferArrK</code>	Precomputed buffer array yields verdict-identical decisions	std 3
<code>route_hh_congPhase</code>	Routing preserves circuit semantics up to global phase (<i>our pipeline</i>)	std 3
<code>compile_hh_confine_correct</code>	Route + optimize $\Rightarrow$ Legal $\wedge$ Equivalent $\wedge$ Confined (<i>our pipeline</i>)	std 3
<code>compile_hh_confine_buffered</code>	Same end-to-end guarantee under buffered policy <i>k</i> (<i>our pipeline</i>)	std 3
<code>route_hh_confine</code>	Routing pass preserves support confinement (<i>our pipeline</i>)	std 3
<code>optimize_conf</code>	Gate cancellation / fusion optimizer preserves confinement (<i>our pipeline</i>)	std 3
<code>confinedb_avoids</code>	Confinement $\wedge A \cap F = \emptyset \Rightarrow$ Gate support avoids <i>F</i>	propext only
<code>checkerSteps_seq</code>	Single-pass validator step complexity is additive / strictly linear	None (0 axioms)
Device Corpus (44 verdicts)	Kernel reduction (decide) of gap/fix/no-over-block across 11 QPUs	propext, Quot.sound

compile-level buffered demonstration. `certifySecurity_sound` quantifies over any `UCom` and never evaluates the optimizer.

Device-graph encoding. We encode each graph as a canonical edge list and express side conditions as one Boolean fold. The earlier

quantified search exhausted kernel recursion depth at about 176 edges. The revised encoding checks the full 156-qubit graph in under 5 seconds (4.9 s for the module). The corpus build completes 2381 jobs with axioms `[propext, Quot.sound]`.

Table 7: Device corpus: published IBM topologies encoded as Coupling n and checked by decide at zero QPU cost. Four verdicts per device: gap, fix, no-over-block, and legality (§5.3).

Topology	Qubits	Edges	Architecture / Generation	Kernel Verdicts
lagos	7	6	Linear / T	4 / 4 ✓
guadalupe	16	16	Heavy-Hex	4 / 4 ✓
hanoi	27	28	Heavy-Hex	4 / 4 ✓
brooklyn	65	72	Heavy-Hex	4 / 4 ✓
nighthawk	120	218	Square Lattice	4 / 4 ✓
sherbrooke	127	144	Heavy-Hex (Eagle)	4 / 4 ✓
washington	127	142	Heavy-Hex (Eagle)	4 / 4 ✓
torino	133	150	Heavy-Hex (Heron r1)	4 / 4 ✓
marrakesh	156	176	Heavy-Hex (Heron r2)	4 / 4 ✓
fez	156	176	Heavy-Hex (Heron r2)	4 / 4 ✓
kingston	156	176	Heavy-Hex (Heron r2)	4 / 4 ✓
11 devices	7–156	1,304	3 Architectural Families	44 / 44 Pass