Quantum State Injection

A Physical-Layer Trust Failure in Quantum Security Systems

SpecterAI Quantum Security
www.SpecterAI.ai

1. Motivation

Most analyses of quantum security focus on protocol correctness, error rates, and post-processing guarantees. These analyses implicitly assume that quantum states entering the protocol are faithful representations of the source's intent. Quantum state injection violates this assumption.

The failure does not occur at the level of cryptography, error correction, or privacy amplification. It occurs earlier, at the physical boundary where quantum states are formed, transmitted, and reconstructed. Once attacker-controlled quantum states enter a trusted reconstruction pipeline, downstream security guarantees remain mathematically correct while becoming operationally irrelevant.

2. Trust Boundaries in Quantum Systems

In classical systems, trust boundaries are explicit: input validation, execution contexts, and privilege separation. In quantum systems, trust boundaries are often implicit. Protocols assume that state preparation is honest, that emitted signals match idealized models, and that measurement outcomes reflect legitimate physical origins.

Quantum state injection exploits the fact that trust is granted before measurement rather than after provenance validation. Once a quantum state is reconstructed into a trusted internal representation, its origin is no longer observable.

3. Physical Origin of the Injection Surface

Practical quantum communication systems typically employ weak coherent sources rather than ideal single-photon emitters. A weak coherent state can be expressed as

$$|\alpha\rangle = e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle.$$

This implies a non-zero probability of multi-photon emission. Multiple physical copies of the same encoded quantum information can exist simultaneously. This is not a protocol flaw; it is a property of the underlying physics.

Reference: Code Example 1 — Photon Emission and State Selection

4. Injection Without Disturbance

For pulses containing more than one photon, an attacker can retain a single photon while forwarding the remaining photons unchanged. No measurement occurs, no basis guessing is required, and no disturbance is introduced.

From the receiver's perspective, the signal arrives on time, the encoding remains intact, and measurement statistics remain within expected bounds. The injected state is indistinguishable from a legitimate one at the protocol level.

5. Storage and Deferred Measurement

The retained quantum state is not measured immediately. Instead, it is preserved in quantum memory without interaction. Information duplication occurs before any classical reconciliation, error correction, or validation.

At this point, exclusivity over the quantum information has already been lost, but the protocol has no mechanism to detect that loss.

Reference: Code Example 2 — Delayed Measurement Logic

6. Measurement After Classical Disclosure

Quantum protocols typically involve later classical communication, including basis reconciliation and sifting. Once this information is publicly disclosed, the attacker can measure the stored quantum state in the correct basis deterministically.

No guessing is required, no uncertainty remains, and no statistical anomaly appears. The security failure is temporal rather than statistical: information is duplicated early and extracted later.

7. Why Statistical Defenses Fail

Most quantum defenses rely on monitoring error rates, disturbance, or statistical deviations. Quantum state injection produces none of these. The protocol validates consistency, not provenance.

Once a state has been accepted into the trusted pipeline, no amount of post-processing can recover information about its physical origin.

8. Minimal Numerical Demonstration

This section illustrates a fundamental point that is often lost when quantum security is discussed purely at the protocol or algorithmic level: certain security-relevant conditions arise naturally from the physics of the source itself, not from any deviation in protocol behavior.

In practical quantum communication systems, weak coherent sources do not emit single photons deterministically. Instead, emission follows a probabilistic distribution in which multi-

photon pulses occur with non-zero frequency. This behavior is intrinsic to the physical implementation and persists even when the protocol is followed correctly and honestly by all legitimate participants.

When multi-photon pulses are generated, multiple physically identical quantum carriers exist before any measurement, basis selection, or classical post-processing takes place. The presence of these additional carriers creates a latent duplication of quantum state information at the physical layer. Crucially, this duplication occurs without introducing errors, without modifying transmission timing, and without violating any protocol rules.

From the perspective of the protocol, nothing abnormal has occurred. Detection statistics remain consistent with expected behavior, error rates remain within acceptable bounds, and reconciliation proceeds normally. Nevertheless, the system has already crossed a critical threshold: trusted state reconstruction is now based on quantum information whose physical origin is no longer singular.

This demonstrates that certain trust failures emerge prior to protocol logic, arising instead from unavoidable physical properties of real-world quantum sources. Any security analysis that assumes uniqueness of transmitted quantum states without validating their physical provenance is therefore incomplete.

Reference: Code Example 3 — Trust Failure Checkpoint

9. Why Privacy Amplification Cannot Repair This Failure

Privacy amplification assumes that an adversary's knowledge is partial or probabilistic. In a state injection scenario, the attacker's knowledge of the raw key is exact for the injected states.

Hashing and compression cannot remove information that is already known. The protocol continues to produce keys that appear statistically secure while secrecy has already been lost at the physical layer.

10. Defensive Implications

Quantum state injection reframes quantum security as a systems security problem. Correctness, error-free operation, and cryptographic strength are insufficient if physical provenance is not enforced.

Effective defenses must treat state formation and transmission as explicit security boundaries rather than implementation details.

11. Core Insight

Once a quantum state is trusted, it is already too late to ask where it came from.

Security must begin before reconstruction, measurement, and statistics.

Code Example 1: Delayed Measurement via Passive Quantum State Retention

This notebook demonstrates, at a minimal numerical level, how **information duplication can arise at the physical layer** without violating protocol logic, introducing disturbance, or triggering detection mechanisms.

The model corresponds to the *exploitation phase* discussed in **Video 34 — Quantum State Injection**, where an attacker:

- Does **not** measure quantum states immediately
- Does **not** guess bases
- Does **not** introduce observable disturbance
- Passively retains valid physical states for **later measurement**

This notebook intentionally avoids hardware-specific implementation and focuses on **conceptual correctness** using classical simulation.

Physical Interpretation

Weak coherent photon sources do not emit single photons deterministically.
Instead, photon emission follows a **Poisson distribution** with a nonzero probability of emitting multiple photons in a single pulse.

When multiple photons are emitted:

- One photon can be **retained** without measurement
- Remaining photons are forwarded intact
- Protocol behavior remains statistically correct
- No errors are introduced

The attacker gains leverage **before** classical post-processing begins.

In [1]:
```
import numpy as np
```

Emission Model

We model photon emission using a Poisson process with mean photon number μ.

This does **not** represent a device exploit — it models unavoidable physical behavior of optical sources.

In [2]:
```
def emit_pulse(mean_photons=0.8):
    """
    Simulate photon emission from a weak coherent source.
    Returns the number of photons emitted in a pulse.
    """
    return np.random.poisson(mean_photons)
```

State Retention Logic

A pulse is eligible for passive retention if **more than one photon** is emitted.

- No measurement is performed
- No basis information is required
- No protocol rules are violated

In [3]:
```
def retain_state(photon_count):
    """
    Determine whether a pulse allows passive state retention.
    """
    return photon_count >= 2
```

Simulation

We now simulate a sequence of pulses and count how many times a valid quantum state could be silently retained.

In [4]:
```
num_trials = 10000
stored_states = 0

for _ in range(num_trials):
```

```
    photons = emit_pulse()
    if retain_state(photons):
        stored_states += 1

stored_states
```

Out[4]: 1821

Output Interpretation

The printed value represents the number of **valid quantum states** that could be retained **without detection** over the simulated run.

Key observations:

- Retention occurs naturally
- No protocol failure is required
- No statistical anomaly is introduced
- Security failure is **temporal**, not probabilistic

This directly motivates delayed measurement attacks once basis reconciliation information becomes available.

Connection to Quantum State Injection

This numerical demonstration supports the core claim of the video:

> Security mechanisms that assume correctness implies trust are insufficient.

The vulnerability emerges **before** cryptographic logic executes — at the boundary between physical emission and trusted reconstruction.

Reference: Code Example 1 — Delayed Measurement Logic

Code Example 2: Delayed Measurement After Basis Disclosure

This notebook demonstrates the *temporal* structure of a delayed measurement attack:

1. A weak coherent source emits pulses with a nonzero probability of multi-photon emission.
2. Multi-photon pulses enable **state retention** without disturbance or measurement.
3. The attacker **waits** until the classical channel discloses basis information.
4. The attacker then measures stored states in the correct basis, recovering bits deterministically.

This models *why the attack pays off after disclosure* rather than during transmission.

Setup

We represent each transmitted signal as an abstract BB84 symbol:

- A bit `b ∈ {0,1}`
- A basis `β ∈ {Z, X}`

Alice sends `(β, b)`.
Bob will later disclose `β` during reconciliation.

The attacker stores only those signals where the photon count is ≥ 2.

In [1]:
```
import numpy as np
```

Emission Model and Retention Rule

Photon counts are simulated with a Poisson process with mean `μ`.

A signal is eligible for passive retention if `n ≥ 2`.

In [2]:
```python
def emit_pulse(mu=0.8):
    return np.random.poisson(mu)

def retain_state(photon_count):
    return photon_count >= 2
```

Protocol Simulation (Abstract)

We now simulate a batch of signals.

For each signal:

- Alice chooses a basis and bit.
- A photon count is drawn from the emission model.
- If multi-photon, the attacker stores the state *without measurement*.

Later:

- The classical channel reveals the basis.
- The attacker measures stored states in the correct basis.
- In this abstract model, correct-basis measurement yields the encoded bit deterministically.

In [3]:
```python
def random_basis(rng):
    return rng.choice(["Z", "X"])

def random_bit(rng):
    return int(rng.integers(0, 2))

def simulate_batch(num_signals=10000, mu=0.8, seed=123):
    rng = np.random.default_rng(seed)

    # Alice's transmitted symbols
    alice_basis = []
    alice_bit = []
    photon_counts = []

    # Attacker storage (indices of stored signals)
```

```
    stored_indices = []

    for i in range(num_signals):
        b = random_bit(rng)
        beta = random_basis(rng)
        n = emit_pulse(mu)

        alice_bit.append(b)
        alice_basis.append(beta)
        photon_counts.append(n)

        if retain_state(n):
            stored_indices.append(i)

    return {
        "alice_bit": np.array(alice_bit, dtype=int),
        "alice_basis": np.array(alice_basis, dtype=object),
        "photon_counts": np.array(photon_counts, dtype=int),
        "stored_indices": np.array(stored_indices, dtype=int),
        "mu": mu,
        "num_signals": num_signals,
        "seed": seed,
    }
```

Delayed Measurement Logic

The attacker **does not** measure during transmission.

Instead, after basis disclosure:

- For each stored signal, the attacker measures in the disclosed basis.
- In this model, disclosed-basis measurement recovers the bit exactly.

We report:

- Number of stored states
- Fraction of total signals stored
- Bit recovery accuracy on stored states

In [4]:
```python
def delayed_measurement_recovery(sim):
    alice_bit = sim["alice_bit"]
    stored = sim["stored_indices"]

    # In this abstract model, once the basis is disclosed,
    # correct-basis measurement returns the original bit.
    recovered_bits = alice_bit[stored].copy()

    # Accuracy vs Alice on the stored subset
    accuracy = (recovered_bits == alice_bit[stored]).mean() if len(stored) > 0 else np.nan

    return recovered_bits, accuracy
```

Run the Demonstration

This prints a concise summary and a short sample of recovered bits.

In [5]:
```python
sim = simulate_batch(num_signals=10000, mu=0.8, seed=123)
recovered_bits, acc = delayed_measurement_recovery(sim)

stored_count = len(sim["stored_indices"])
stored_fraction = stored_count / sim["num_signals"]

print("=== Delayed Measurement Demonstration ===")
print(f"Signals simulated:         {sim['num_signals']}")
print(f"Mean photon number (μ):    {sim['mu']}")
print(f"Stored states (n≥2):       {stored_count}")
print(f"Stored fraction:           {stored_fraction:.4f}")
print(f"Recovery accuracy (stored): {acc:.4f}")
print()
print("Recovered bits sample:", recovered_bits[:20] if stored_count > 0 else "None")
```

```
=== Delayed Measurement Demonstration ===
Signals simulated:         10000
Mean photon number (μ):    0.8
Stored states (n≥2):       1929
Stored fraction:           0.1929
Recovery accuracy (stored): 1.0000

Recovered bits sample: [1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 0 0]
```

Interpretation

The key result is that **measurement happens after disclosure**, not during transmission.

- No guessing is required.
- No disturbance is introduced.
- Protocol validation checks (e.g., QBER thresholds) are not triggered by this model.
- The attacker's advantage is **temporal**: information is duplicated early and extracted later.

This aligns with the video's central claim:
security collapses when provenance is assumed rather than validated.

Reference: Code Example 2 — Delayed Measurement Logic

Code Example 3: Trust Failure Checkpoint

This notebook produces a simple checkpoint that highlights the core failure mode:

- **Protocol-side metrics look acceptable**
- **Yet attacker knowledge is non-zero and structural**
- The system validates *consistency* (e.g., QBER), not *provenance*

We simulate:

1. a weak coherent source (Poisson photon count)
2. passive state retention for multi-photon pulses (n ≥ 2)
3. delayed measurement after basis disclosure (perfect recovery on stored subset)
4. protocol checks based on QBER thresholds

The output is a single, readable summary: "ACCEPTED" while trust is violated.

In [1]:
```
import numpy as np
```

Model Parameters

- `mu` : mean photon number of the weak coherent source
- `num_signals` : number of transmitted signals
- `qber_baseline` : normal operational noise (alignment + detector + channel)
- `qber_threshold` : a typical acceptance threshold used in QKD discussions (illustrative)

Important: in this model, passive retention does **not** introduce additional errors.

In [2]:
```
mu = 0.8
num_signals = 10000

# Baseline operational noise (illustrative, not vendor-specific)
```

```
qber_baseline = 0.010  # 1.0%

# Illustrative acceptance threshold (commonly discussed order of magnitude)
qber_threshold = 0.110  # 11.0%

seed = 123
rng = np.random.default_rng(seed)
```

Helpers

We simulate:

- photon counts via Poisson emission
- basis and bit choice per signal
- passive retention when n ≥ 2
- delayed measurement after basis disclosure

We then compute:

- observed QBER (baseline only)
- attacker coverage (fraction of signals retained)
- attacker recovered bits on retained subset

In [3]:
```
def emit_pulse(mu, rng):
    return rng.poisson(mu)

def random_basis(rng):
    return rng.choice(["Z", "X"])

def random_bit(rng):
    return int(rng.integers(0, 2))

def retain_state(photon_count):
    return photon_count >= 2
```

Simulate Transmission + Passive Retention

Alice generates `(basis, bit)` per signal. Photon count is drawn from the source model. If multi-photon, the attacker can retain one valid physical state *without measurement*.

In [4]:
```
alice_basis = np.empty(num_signals, dtype=object)
alice_bit = np.empty(num_signals, dtype=int)
photon_counts = np.empty(num_signals, dtype=int)

stored_indices = []

for i in range(num_signals):
    alice_basis[i] = random_basis(rng)
    alice_bit[i] = random_bit(rng)
    photon_counts[i] = emit_pulse(mu, rng)

    if retain_state(photon_counts[i]):
        stored_indices.append(i)

stored_indices = np.array(stored_indices, dtype=int)
stored_count = len(stored_indices)
stored_fraction = stored_count / num_signals

stored_count, stored_fraction
```

Out[4]:
```
(1786, 0.1786)
```

Observed Protocol Metrics (QBER)

We model the observed QBER as baseline operational noise only.

Key point:

- Passive retention does **not** alter Bob's measurement stream in a way that raises QBER.
- Therefore, protocol-side validation can still pass.

In [5]:
```
# Simulate observed QBER as a noisy estimate around baseline.
# (Binomial sampling: number of errors in sifted bits ~ Binomial(N, qber_baseline))
# For simplicity, assume half the signals are sifted (basis match probability ~ 0.5).
sifted = num_signals // 2
```

```
observed_errors = rng.binomial(sifted, qber_baseline)
observed_qber = observed_errors / sifted

protocol_accepts = observed_qber <= qber_threshold

observed_qber, protocol_accepts
```

Out[5]: (0.011, True)

Attacker Recovery After Basis Disclosure

After basis disclosure, the attacker measures stored states in the correct basis.

In this abstract model:

- measurement on the stored subset is perfectly aligned after disclosure
- attacker recovers the bit deterministically for retained states

In [6]:
```
# On retained subset, delayed measurement after basis disclosure yields the correct bits.
recovered_bits = alice_bit[stored_indices].copy()

# Accuracy on retained subset (should be 1.0 by construction here)
recovery_accuracy = (recovered_bits == alice_bit[stored_indices]).mean() if stored_count > 0 else np.nan

recovery_accuracy
```

Out[6]: np.float64(1.0)

Trust Failure Checkpoint Report

This prints the exact contradiction we care about:

- The protocol may **ACCEPT** based on QBER thresholds
- Meanwhile, attacker knowledge is **non-zero**
- Provenance is never evaluated by the protocol-side metrics

In [7]:
```
print("=== Trust Failure Checkpoint ===")
print(f"Signals simulated:            {num_signals}")
print(f"Mean photon number (μ):       {mu}")
print()
print(f"Sifted bits (approx):         {sifted}")
print(f"Observed errors:              {observed_errors}")
print(f"Observed QBER:                {observed_qber:.4%}")
print(f"Acceptance threshold:         {qber_threshold:.4%}")
print(f"Protocol status:              {'ACCEPTED' if protocol_accepts else 'REJECTED'}")
print()
print(f"Stored states (n≥2):          {stored_count}")
print(f"Stored fraction:              {stored_fraction:.4%}")
print(f"Recovery accuracy (stored):   {recovery_accuracy:.4%}")
print()
print("Checkpoint:")
print("- Protocol validates consistency (QBER), not provenance.")
print("- Attacker knowledge can be non-zero while metrics remain normal.")
```

```
=== Trust Failure Checkpoint ===
Signals simulated:            10000
Mean photon number (μ):       0.8

Sifted bits (approx):         5000
Observed errors:              55
Observed QBER:                1.1000%
Acceptance threshold:         11.0000%
Protocol status:              ACCEPTED

Stored states (n≥2):          1786
Stored fraction:              17.8600%
Recovery accuracy (stored):   100.0000%

Checkpoint:
- Protocol validates consistency (QBER), not provenance.
- Attacker knowledge can be non-zero while metrics remain normal.
```

Interpretation

If the protocol status is **ACCEPTED** while the stored fraction is non-zero, the system has a structural trust failure:

- Correctness checks pass
- Provenance is not evaluated
- Downstream steps (error correction, privacy amplification) cannot restore lost exclusivity

Reference: Code Example 3 — Trust Failure Checkpoint