

Limiting Cross-Session Reuse of Table-Specific Attack Information: Session Diversification of S-Boxes with Per-Instance Cryptographic Guarantees

Annas Malik

University of Central Punjab

Ghulam Mustafa

University of Central Punjab

You-Rak Choi

Korea Atomic Energy Research Institute

Ki-Il Kim

`kikim@cnu.ac.kr`

Chungnam National University

Research Article

Keywords: Session diversification, Dynamic S-box, Affine equivalence, Cross-session transfer, Side-channel profiling, Quantum random number generator

Posted Date: October 19th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11245232/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Limiting Cross-Session Reuse of Table-Specific Attack Information: Session Diversification of S-Boxes with Per-Instance Cryptographic Guarantees

Abstract

A fixed substitution box (S-box) allows table-specific analysis and profiling to be reused across sessions, whereas many dynamic S-box schemes achieve diversification at the cost of uneven cryptographic quality or runtime search. This paper presents QSD-SBox, a session-diversification scheme for 8×8 S-boxes that changes the concrete table in every session while preserving per-instance cryptographic guarantees. Each session S-box is constructed as $S = A_{\text{out}} \circ I \circ A_{\text{in}}$, where I is inversion in $\text{GF}(2^8)$ and A_{in} and A_{out} are seed-derived affine permutations. Consequently, every instance has nonlinearity 112, differential uniformity 4, algebraic degree 7, and boomerang uniformity 6 without runtime search. The objective is not to strengthen a single S-box, but to limit the cross-session transfer of table-specific attack information. Across 999,000 ordered session pairs, differential characteristics, linear approximations, and two-round trails profiled in one session transferred to another only at chance level. In a simulated template attack, a profile built with a static AES S-box recovered the next session's full key with 50 attack traces, whereas cross-session profiling with QSD-SBox recovered no more key bytes than guessing under the evaluated leakage model. The AES active S-box bound also held in every evaluated session. The construction is seed-source agnostic, and a C implementation generates a new S-box in $136 \mu\text{s}$. These results show that session diversification can reduce the reuse of table-specific attack information while retaining established per-instance cryptographic guarantees.

Keywords: Session diversification, Dynamic S-box, Affine equivalence, Cross-session transfer, Side-channel profiling, Quantum random number generator

1 Introduction

Substitution boxes (S-boxes) provide the principal nonlinear component of many block ciphers. A fixed S-box, such as that used in AES, has the important advantage that its cryptographic properties can be thoroughly analyzed once and remain independent of the key. However, this stability also creates a cross-session reuse problem. Difference

047 and linear approximation tables, favorable trails, and side-channel templates tied to
048 the concrete S-box remain applicable across sessions and keys. Rekeying alone does not
049 invalidate such table-specific information because the underlying S-box is unchanged.

050 Dynamic S-box schemes address this limitation by changing the substitution table
051 according to a key, seed, or session state. However, diversification alone does not ensure
052 consistent cryptographic strength. Many existing schemes rely on chaotic maps, evolu-
053 tionary optimization, or heuristic generation, making properties such as nonlinearity
054 and differential uniformity dependent on parameters, initial conditions, or runtime
055 search. More recent algebraic and physical constructions improve particular aspects,
056 including optimal nonlinearity or physical entropy generation, but two requirements
057 are still rarely addressed together: (i) cryptographic guarantees that hold for every
058 generated S-box and (ii) an explicit evaluation of how much table-specific attack
059 information transfers between independently diversified sessions.

060 This work addresses this gap through QSD-SBox, which treats dynamic S-box gen-
061 eration specifically as a session-diversification problem. Each session S-box is drawn
062 from the affine equivalence class of multiplicative inversion in $(GF(2^8))$, with seed-
063 derived invertible affine transformations applied to its input and output. Because the
064 relevant cryptographic properties are invariant under affine equivalence, every gen-
065 erated instance retains the established nonlinearity, differential uniformity, algebraic
066 degree, and boomerang uniformity of the underlying inversion construction without
067 runtime search. At the same time, the concrete substitution table changes between
068 sessions, allowing table-specific differential, linear, trail, and side-channel information
069 learned in one session to be tested for transfer to another.

070 Accordingly, the novelty of QSD-SBox is not affine equivalence itself, nor the
071 introduction of a new nonlinear primitive. Rather, it is the systematic use of affine-
072 equivalent S-boxes as a session-diversification mechanism that combines per-instance
073 cryptographic guarantees with an explicit, quantitative evaluation of cross-session
074 transferability. The security objective is therefore deliberately narrower than claiming
075 a stronger individual S-box: QSD-SBox aims to make table-specific attack information
076 acquired in one session less reusable in subsequent sessions while preserving established
077 cryptographic properties in every instance.

078 QSD-SBox tests this hypothesis with the class of multiplicative inversion in
079 $GF(2^8)$. A session seed determines two invertible affine maps, and the session S-
080 box is $\mathcal{S} = A_{\text{out}} \circ I \circ A_{\text{in}}$. Nonlinearity, differential uniformity, algebraic degree and
081 boomerang uniformity are invariant under affine equivalence (Proposition 1), so every
082 session attains the values of the AES S-box without a search for good candidates. The
083 class holds about 2^{129} distinct tables (Section 3.7). QSD-SBox adds no new nonlinear
084 primitive; it instantiates a well understood class afresh for each session. This also lim-
085 its the security claim: a single session is as strong as a cipher built on inversion, and
086 the gain sought is less cross-session transfer of table-specific information. Knowledge
087 that depends only on affine invariants stays valid in every session (Eq. (12)), and we
088 measure the transfer of everything else.

089 The session seed must be unpredictable. A properly seeded cryptographically
090 secure pseudorandom generator (CSPRNG) suffices for the algebraic guarantees, and a
091 CSPRNG baseline confirms that they do not depend on the seed source (Section 5.2).
092

We draw seeds from the quantum random number generator (QRNG) of the OTQuS-box framework (Malik et al. 2026) for one reason: to give each seed a physical origin that does not rest on a computational assumption (Herrero-Collantes and Garcia-Escartin 2017; Ma et al. 2016). The quantum source changes where the seed comes from, not the strength of the S-box. We characterize it with the SP 800-90B entropy estimators (Turan et al. 2018) and the Dieharder battery (Brown et al. 2013) under two noise models, and with 10^6 shots on each of two IBM devices. A simulator draws its shots with a classical pseudorandom generator, so the simulated tests show how noise distorts the output distribution, and the hardware runs supply the physical evidence.

The evaluation follows the security argument. At the S-box level, the invariants are verified over 1,000 sessions seeded by the quantum source and 488 sessions seeded by hardware shots. At the cipher level, the session S-box is placed in a cipher with the structure of AES-128, and the active S-box bounds and the tightness of the bound over two rounds are checked for every session. Across sessions, differential characteristics, linear approximations and trails over two rounds profiled on one session are applied to another over all 999,000 ordered pairs of 1,000 sessions, and a simulated template attack, with a static AES S-box as the control, tests whether a side-channel profile built in one session recovers the key of the next. At the level of implementation, C code measures the cost of a session, and QSD-Audio, an authenticated post-quantum audio prototype, is a feasibility case study in which two endpoints rotate S-boxes in step.

Section 2 reviews related work. Section 3 describes the construction, the attacker model and the split between invariant and table-specific information. Section 4 characterizes the seed source. Section 5 evaluates the S-boxes, Section 6 studies security within and across sessions, Section 7 gives the cost of a session and Section 8 presents QSD-Audio. Section 9 states the limitations and Section 10 concludes.

The main contributions of this work are as follows.

- Session diversification with per-instance cryptographic guarantees. QSD-SBox generates a distinct 8×8 S-box for each session as $S = A_{\text{out}} \circ I \circ A_{\text{in}}$. Affine equivalence guarantees nonlinearity 112, differential uniformity 4, algebraic degree 7, and boomerang uniformity 6 for every instance without runtime search.
- Quantitative evaluation of cross-session transfer. We distinguish single-session cryptographic strength from the reuse of table-specific information across sessions. Differential characteristics, linear approximations, and two-round trails are evaluated over all 999,000 ordered pairs of 1,000 sessions, together with a simulated cross-session template attack using a static AES S-box as the control. Under the evaluated attack and leakage models, table-specific profiles transferred only at chance level, while affine-invariant information remains transferable.
- Practical and seed-source-independent realization. Experiments with quantum-generated, hardware-generated, and CSPRNG seeds confirm that the per-instance guarantees do not depend on the entropy source. A C implementation generates a session S-box in 136 (μ)s, while an authenticated post-quantum audio prototype demonstrates synchronized S-box rotation as a deployment feasibility case study.

2 Related work

2.1 Dynamic and algebraic S-box construction

Most dynamic S-boxes are built from chaotic maps and heuristics (Ali et al. 2022; Malik et al. 2023; Zahid et al. 2023; Yogaraja and Rani 2023; Singh et al. 2024; Dube and Yadav 2024; Waheed et al. (2023) survey the field. A more recent line uses explicit algebraic or physical structure. Garipcan et al. (2025) derive S-boxes from FPGA metastability with hash conditioning, a classical physical source without an algebraic guarantee. Aslam et al. (2025) reach $NL = 112$ with a polynomial ring construction, so optimal nonlinearity in a dynamic construction is not new. Other generators use Montgomery curves (Ullah et al. 2026), Schreier graphs (Razaq et al. 2026) and $GL(8, GF(2))$ parameters derived from fingerprints (Rasheed and Amjad 2026). Ejaz et al. (2025) generate an S-box by elliptic curve mutation in about 0.9 ms, a useful efficiency reference (Section 9). Murtaza and Hayat (2025) integrate a dynamic S-box into a complete image cipher, and Naseer et al. (2025) motivate the extended metric set we use.

2.2 Quantum random number generation on gate-based devices

Gate-based QRNGs measure qubits prepared in superposition. Kumar and Pravinkumar (2024) build one from SX and CNOT gates on IBM hardware; Saini et al. (2024) seed an S-box built by a genetic algorithm with QRNG output; Henry (2024) and Jayalatsumi et al. (2025) discuss QRNGs for encryption, including variants that use machine learning. Related quantum work includes an LED circuit for Grover attacks (Wang et al. 2024), a chaotic image cipher on NEQR (Patel et al. 2024) and a 16-bit S-box circuit (Zhang et al. 2025). Salehi et al. (2026) propose parameterized QRNG circuits for superconducting hardware that pass all 15 SP 800-22 tests in many configurations under IBM noisy simulators. Their work relies on simulators, as most of ours does, and passing SP 800-22 in simulation reflects the chosen noise model and sample size rather than a physical guarantee. The OTQuSbox framework (Malik et al. 2026) introduced the Hadamard and CNOT circuit with XOR whitening that QSD-SBox reuses; it was assessed there with SP 800-22 over 1,000 trials of 200,000 shots under a depolarizing noise model.

2.3 Trust models and randomness extraction

QRNGs differ in what they ask the user to trust (Herrero-Collantes and Garcia-Escartin 2017; Ma et al. 2016). Source-independent protocols certify randomness without trusting the source, on an IBM cloud device (Li et al. 2021) and in integrated photonics (Du et al. 2025). Device-independent protocols certify it from Bell violations (Farkas et al. 2026), and randomness amplification has been shown experimentally (Kulikov et al. 2026). Other work models laser phase noise (Yang et al. 2026) or extracts at 100 Gbit/s with Toeplitz hashing on an FPGA (Bomhals et al. 2026), which shows that an explicit extractor and a stated secure rate are expected. Toeplitz matrices form a universal hash family (Krawczyk 1994), so by the leftover hash lemma (Impagliazzo et al. 1989) they extract nearly uniform bits from any source with a

known min-entropy bound. Our source is a gate-based QRNG whose device must be trusted, evaluated in simulation and in two hardware runs, so it sits at the low end of this range of assurance.

2.4 Statistical assessment of random sources

Three families of tools are common. SP 800-22 (Bassham et al. 2010) runs 15 hypothesis tests on sequences of about 10^6 bits. Batteries such as Dieharder (Brown et al. 2013) and TestU01 (L’Ecuyer and Simard 2007) run many more tests on far larger samples and are the usual choice for detecting small biases in generators. SP 800-90B (Turan et al. 2018) takes a different view: it estimates the min-entropy of raw samples from a noise source, tests whether they are IID, checks restart behavior and defines continuous health tests. NIST states that tests of the SP 800-22 kind nearly always fail on the raw output of a noise source and are not useful for monitoring it (Turan et al. 2018). The non-IID estimators of SP 800-90B are known to underestimate entropy, sometimes by a wide margin (Zhu et al. 2017), which we observe as well. SP 800-90C (National Institute of Standards and Technology 2025) specifies how an entropy source, a conditioning function and a deterministic generator are combined; in that view our SHAKE-256 stage is a conditioning and expansion step, not an extractor, and it cannot add entropy.

2.5 Research gap

Neither QRNGs nor the class of S-boxes derived from field inversion is new, and a dynamic construction has reached optimal nonlinearity before (Aslam et al. 2025). What is missing is a session diversification scheme that gives per-instance cryptographic guarantees and also evaluates, under a stated attacker model, the cross-session transfer of table-specific information. The dynamic constructions cited above report the metrics of their tables, not this transfer. Table 1 summarizes existing approaches. Table 2 compares a static S-box, heuristic and algebraic dynamic S-boxes, OTQuS-box and QSD-SBox on common dimensions: guarantees per instance, diversity, reuse of precomputation, what the attacker is assumed to know, generation cost and implementation assumptions.

2.6 Relation to OTQuSbox

QSD-SBox and OTQuSbox (Malik et al. 2026) now share the same entropy source and differ in everything built on it. (i) *Seed use*: OTQuSbox takes the first 256 distinct whitened bytes as a seed permutation; QSD-SBox conditions the whole whitened stream with SHAKE-256 (National Institute of Standards and Technology 2015) and expands the result. (ii) *Algebra*: OTQuSbox applies inversion and then exponentiation with the fixed primitive element $g = 0x15$ to the seed permutation, so its S-box is a random byte permutation followed by fixed maps and behaves like a random permutation; QSD-SBox derives two full linear layers in $GL(8, \mathbb{F}_2)$ and two translations from the seed and keeps the inverse map intact. (iii) *Guarantees*: OTQuSbox reports $NL = 108$, BIC-NL 103.71 and $\delta = 10$ for its example and has no bound for other seeds; QSD-SBox proves $NL = 112$, $\delta = 4$, degree 7 and boomerang uniformity 6 for

Table 1 Summary of S-box construction approaches.

Reference	Approach	Contribution	Gap
Ali et al. 2022; Malik et al. 2023; Yogaraja and Rani 2023; Zahid et al. 2023; Singh et al. 2024; Dube and Yadav 2024	Chaotic, dependent on the key, heuristic	Dynamic bijective S-boxes with good metrics	Properties depend on parameters; search overhead
Kumar and Pravinkumar 2024; Henry 2024; Saini et al. 2024; Jayalatsumi et al. 2025	QRNG, with optimization or machine learning	Quantum randomness in dynamic mechanisms	Not tied to a construction with proven properties
Zhang et al. 2025; Malik et al. 2026	Algebraic, with a quantum circuit	Quantum circuits and session S-boxes; OTQuSbox uses inversion and a fixed exponent NL up to 112 (Aslam et al. 2025); one builds an S-box in about 0.9 ms (Ejaz et al. 2025)	Fixed exponent limits the seed's effect: $NL = 108$, $\delta = 10$
Garipcan et al. 2025; Aslam et al. 2025; Ullah et al. 2026; Razaq et al. 2026; Rasheed and Amjad 2026; Ejaz et al. 2025	Physical entropy or algebraic generators (2025–2026)		Guarantees for every instance and transfer across sessions not shown together
QSD-SBox	Affine class of inversion; seeds from the OTQuSbox QRNG	Invariants proven for every session; transfer across sessions measured	Knowledge based on invariants still transfers; side channels simulated only

every seed. (iv) *Source assessment*: OTQuSbox used SP 800-22 under a depolarizing model; we show that this model produces an exactly uniform distribution (Section 3.2) and assess the source with SP 800-90B and Dieharder under a harsher model as well. (v) *Synchronization*: OTQuSbox exchanges the seed with ephemeral ECDH, which is not post-quantum secure; QSD-Audio uses ML-KEM-768 and ML-DSA-65 (Section 8).

3 The QSD-SBox construction

3.1 Overview

Figure 1 shows the four stages. The OTQuSbox circuit is run for 4,096 shots; health tests check the raw shots, XOR whitening halves them to 2,048 bytes, and SHAKE-256 conditions these into a 128-byte seed K (Sections 3.2 and 3.3). A SHAKE-256 expander turns K into two invertible linear maps $L_{\text{in}}, L_{\text{out}}$ and a constant α (Section 3.4). The S-box is then

$$\mathcal{S}(x) = L_{\text{out}}\left((L_{\text{in}}(x) \oplus \alpha)^{-1}\right) \oplus \beta, \quad (1)$$

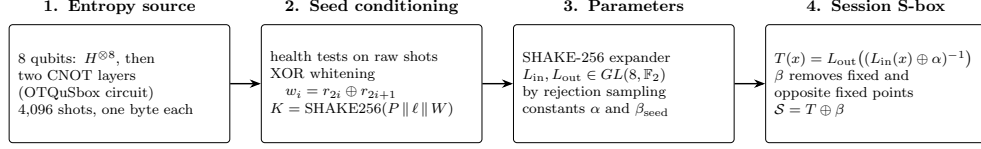
with $x \in GF(2^8)$, $0^{-1} = 0$, and β chosen by a search, driven by the seed, that removes fixed and opposite fixed points (Section 3.5). With $A_{\text{in}}(x) = L_{\text{in}}(x) \oplus \alpha$, $A_{\text{out}}(y) = L_{\text{out}}(y) \oplus \beta$ and $I(x) = x^{-1}$,

$$\mathcal{S} = A_{\text{out}} \circ I \circ A_{\text{in}}. \quad (2)$$

Table 2 Session diversification compared under a common attacker model. Values marked [‡] were measured or recomputed by us; the others are reported in the cited papers or follow from the construction. “Not evaluated” means the cited papers do not measure the property.

Scheme	Per-instance guarantees	Diversity across sessions	Reuse of table-specific precomputation	Attacker is assumed to know	Generation cost per session	Implementation assumptions
Static AES S-box (Daemen and Rijmen 2020)	$NL = 112$, $\delta = 4$, degree 7, boomerang uniformity 6 [‡]	None; one public table	Complete, under every key and session	The table	None	Constant-time bitsliced code and AES instructions exist
Chaotic and heuristic S-boxes (Ali et al. 2022; Malik et al. 2023; Zahid et al. 2023; Singh et al. 2024; Ejaz et al. 2025)	None; reported NL 103.5 to 111.75 for sample tables	Per key or seed	Not evaluated	The generator, not the key	Search per table; 0.9 ms reported for (Ejaz et al. 2025)	Both ends must reproduce the generator’s arithmetic exactly
Algebraic dynamic S-boxes (Aslam et al. 2025; Rasheed et al. 2025; Rasheed and Amjad 2026)	$NL = 112$ in (Aslam et al. 2025); not established in (Rasheed and Amjad 2026)	Per parameter or fingerprint	Not evaluated	The construction	Not reported	Not stated
OTQuSbox (Malik et al. 2026)	None per seed; sample table $NL = 108$, $\delta = 10$	Per session	Not evaluated	Circuit and fixed exponent, not the seed	Not reported	Seed exchanged by ECDH, which is not post-quantum
QSD-SBox	$NL = 112$, $\delta = 4$, degree 7, boomerang uniformity 6 for every seed (proof; checked on 2,488 sessions [‡])	Per session, from a class of about 2^{129} tables [‡]	Invariant knowledge reusable; table-specific profiles at chance level [‡] (Section 6.3)	The construction and the class, not the seed	136 μs in $\mathbf{C}^{\dagger}$	Table lookups, not constant time; seeds by ML-KEM in QSD-Audio

277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322



Every S is affine equivalent to inversion in $GF(2^8)$: $NL = 112$, $\delta = 4$, algebraic degree 7, boomerang uniformity 6.

Fig. 1 QSD-SBox pipeline: entropy source, seed conditioning, parameter derivation and S-box construction.

3.2 Entropy source: the OTQuSbox QRNG

The source is the QRNG of OTQuSbox (Malik et al. 2026), used without change (Fig. 2). Eight qubits start in $|0\rangle^{\otimes 8}$. A Hadamard gate on each qubit gives the uniform superposition, a first CNOT layer acts on the pairs (0, 1), (2, 3), (4, 5), (6, 7) and a second on (1, 2), (3, 4), (5, 6), the first index being the control:

$$U = \prod_{i \in \{1, 3, 5\}} CX_{i \rightarrow i+1} \prod_{i \in \{0, 2, 4, 6\}} CX_{i \rightarrow i+1} H^{\otimes 8}. \quad (3)$$

After the gates of Eq. (3), all qubits are measured in the computational basis. Shot j gives the byte $r_j = \sum_{i=0}^7 b_i^{(j)} 2^i$, where $b_i^{(j)}$ is the bit read from qubit i . Adjacent shots are then combined as in OTQuSbox,

$$w_i = r_{2i} \oplus r_{2i+1}, \quad (4)$$

which halves the output rate. Every CNOT acts on neighboring qubits, so on a device with linear or heavy-hex connectivity the circuit maps to a chain of physical qubits without extra swap gates. A circuit with rotation and Toffoli gates on qubits that are not adjacent, which we tested first, showed on IBM hardware about three times the bit bias of a circuit with Hadamard gates only (Online Resource 1, Section S2), so we use the OTQuSbox circuit.

Ideal and published noise model. The state $|+\rangle^{\otimes 8}$ is unchanged by any CNOT between its qubits, so without noise the output is uniform on $\{0, \dots, 255\}$ and the CNOT layers do not alter the distribution. The noise model published with OTQuSbox adds a depolarizing error with $p = 0.01$ after each Hadamard and $p = 0.03$ after each CNOT. This does not change the output either: a CNOT permutes computational basis states, and a depolarizing channel mixes the state with the maximally mixed state, so both keep a uniform distribution in the computational basis. The exact distribution computed from the simulator’s density matrix confirms this (min-entropy 8.000000 bits per byte). A statistical test of simulated output under this model therefore tests only the simulator’s classical sampler; we include it as a control.

Hardware stress model. To see how the source behaves when noise does bias it, we also use a harsher model: depolarizing errors (10^{-3} after gates on one qubit, 10^{-2} after CNOT), thermal relaxation with $T_1 = 100 \mu s$ and $T_2 = 80 \mu s$ over gate times of 50 ns and 300 ns, and asymmetric readout error $\Pr(1 | 0) = 0.015$, $\Pr(0 | 1) = 0.025$.

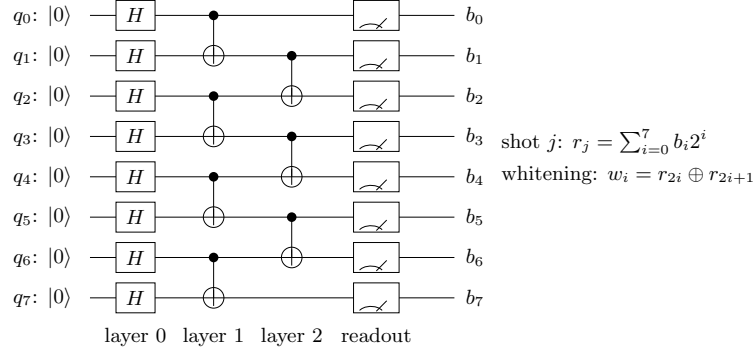


Fig. 2 The OTQuSBox circuit used as the entropy source: a Hadamard layer, two layers of CNOT gates between neighboring qubits, readout in the computational basis, and XOR whitening of adjacent shots.

Relaxation and the readout asymmetry both push outcomes toward 0. The exact distribution of one shot has min-entropy 7.8244 bits per byte, and the largest bit bias, $|\Pr(b_i = 1) - 1/2|$, is 0.0078. The CNOT layers add relaxation time, so the raw bias is larger than for a circuit with Hadamard gates only under the same model (0.0052).

Effect of whitening. For any mask $u \neq 0$, let $\varepsilon_u = \Pr[u \cdot r = 0] - \frac{1}{2}$ be the bias of the parity $u \cdot r$ of one shot. If two shots are independent, the piling-up lemma (Matsui 1994) gives

$$\Pr[u \cdot w = 0] - \frac{1}{2} = 2\varepsilon_u^2. \quad (5)$$

Every bias, and every linear correlation between bits, is therefore squared and doubled. Under the stress model the largest bit bias falls from 0.0078 to 1.2×10^{-4} and the exact min-entropy rises from 7.8244 to 7.9977 bits per byte. Equation (5) depends on independence between shots, which Section 4 tests.

Preparation that compensates readout error (CAQ). The readout bias can also be removed at the source by preparing each qubit with $R_y(\theta_i^*)$ instead of a Hadamard, with θ_i^* chosen from the qubit’s readout error rates $\varepsilon_{01}^{(i)} = \Pr(1 | 0)$ and $\varepsilon_{10}^{(i)} = \Pr(0 | 1)$ so that the observed probability of reading 1 is 1/2:

$$p_i^* = \frac{0.5 - \varepsilon_{01}^{(i)}}{1 - \varepsilon_{01}^{(i)} - \varepsilon_{10}^{(i)}}, \quad \theta_i^* = 2 \arcsin \sqrt{p_i^*}. \quad (6)$$

For ideal readout $\theta^* = \pi/2$ and the circuit reduces to the generator with Hadamard gates only. We evaluate this variant, alone and with whitening, as an alternative to whitening in Section 4. It needs current calibration data for each qubit, whereas whitening needs none, so QSD-SBox uses whitening by default.

3.3 Seed conditioning

Before conditioning, each session’s 4,096 raw shots pass the two continuous health tests of SP 800-90B (Turan et al. 2018): the repetition count test and the adaptive proportion test (window 512). Their cutoffs, 5 repeated values and 20 occurrences per

415 window, follow from a claimed raw min-entropy of 6.6 bits per shot, just below the
 416 smallest SP 800-90B assessment of the simulated raw stream (Table 5; one hardware
 417 run assessed slightly lower, Section 4.4), and a false alarm probability of 2^{-20} . A failing
 418 batch is discarded and redrawn. Over 10,000 simulated sessions under the stress model
 419 the tests raised no alarm, all 488 batches of the two hardware runs of Section 4.4
 420 passed, and over the 2.0×10^9 seeds of the Dieharder seed stream (Section 4.3) they
 421 redrew about one batch in a million. They caught every session whose shots were all
 422 identical and every session with four qubits stuck at 0 (min-entropy 3.9 bits per byte),
 423 and 995 of 1,000 sessions with readout collapsed to $\Pr(1) = 0.2$ (2.6 bits per byte). A
 424 single stuck qubit (6.85 bits per byte, above the claimed 6.6) went undetected, as the
 425 tests are designed to catch gross failures rather than small losses. The 2,048 whitened
 426 bytes W are then conditioned as

$$427 \quad K = \text{SHAKE256}(P \parallel \text{Len}(W) \parallel W), \quad (7)$$

429 where P is the personalization string "QASBOX-LINEARIZED-POLYNOMIAL-V1",
 430 $\text{Len}(W)$ is the 8-byte big-endian length and K is 128 bytes long. A determinis-
 431 tic conditioning function cannot output more entropy than it receives (Turan et al.
 432 2018, Sec. 3.1.5). Here the input carries at least $2,048 \times 6.638 \approx 13,600$ bits of min-
 433 entropy under the most conservative SP 800-90B assessment of the whitened stream
 434 (Section 4.2), far above the 256-bit security level of SHAKE-256, so the strength of K
 435 is limited by SHAKE-256 and not by the source.

437 3.4 Algebraic parameter derivation

438 The seed is expanded with SHAKE-256 under domain separation,

$$439 \quad \text{Expand}(K, c, \ell) = \text{SHAKE256}(P' \parallel K \parallel c \parallel \ell), \quad (8)$$

441 with $P' = \text{"QASBOX-PARAMETER-EXPANDER-V1"}$, an 8-byte counter c that increases
 442 with every call, and fixed labels ℓ for the input map, the output map, α and β (the
 443 attempt number is appended to the labels of the two maps; the exact strings are in
 444 the released code). Field arithmetic uses the modulus $x^8 + x^6 + x^5 + x^4 + x^2 + x + 1$
 445 (0x177). A linearized polynomial

$$446 \quad L(x) = \sum_{i=0}^7 c_i x^{2^i}, \quad c_i \in GF(2^8), \quad (9)$$

447 is $GF(2)$ -linear, $L(a \oplus b) = L(a) \oplus L(b)$, and every $GF(2)$ -linear map of $GF(2^8)$ has
 448 this form. For L_{in} (and in the same way L_{out}), the expander of Eq. (8) draws coefficient
 449 vectors $(c_0, \dots, c_7) \leftarrow \text{Expand}(K, c, \ell \| t)$ for Eq. (9), $t = 1, 2, \dots$, and the first vector
 450 whose map is bijective on all 256 elements is accepted. This rejection sampling selects
 451 a uniformly random element of $GL(8, \mathbb{F}_2)$ and places no other restriction on the map.
 452 A random 8×8 binary matrix is invertible with probability $\prod_{i=1}^8 (1 - 2^{-i}) \approx 0.2899$,
 453 so about 3.4 draws are needed on average. The constant α is one expanded byte, and
 454 a 32-byte value $\beta_{\text{seed}} \leftarrow \text{Expand}(K, c, \ell_{\beta})$ is kept for the choice of β .

Algorithm 1 Session S-box construction**Require:** Conditioned seed K (Eq. (7))**Ensure:** Bijective S-box $\mathcal{S}$

```

1: Derive  $L_{\text{in}}, L_{\text{out}}$  by rejection sampling until each is bijective;  $\alpha \leftarrow$ 
   Expand( $K, c, \ell_\alpha$ );  $\beta_{\text{seed}} \leftarrow \text{Expand}(K, c, \ell_\beta)$ 
2: for  $x \leftarrow 0$  to 255 do
3:    $T(x) \leftarrow L_{\text{out}}((L_{\text{in}}(x) \oplus \alpha)^{-1})$ , with  $0^{-1} = 0$ 
4: end for
5: Order  $\beta_1, \dots, \beta_{256}$  by Eq. (10);  $\beta^* \leftarrow \beta_1$ ; best  $\leftarrow \infty$ 
6: for  $k \leftarrow 1$  to 256 do
7:   compute FP( $\beta_k$ ) and OFP( $\beta_k$ ) for  $T \oplus \beta_k$ 
8:   if FP + OFP < best then
9:      $\beta^* \leftarrow \beta_k$ ; best  $\leftarrow$  FP + OFP
10:  end if
11:  if best = 0 then break
12:  end if
13: end for
14:  $\mathcal{S}(x) \leftarrow T(x) \oplus \beta^*$ ; check linearity of  $L_{\text{in}}, L_{\text{out}}$  and bijectivity of  $\mathcal{S}$ 
15: return  $\mathcal{S}$ 

```

3.5 Session S-box construction

The base table $T(x) = L_{\text{out}}((L_{\text{in}}(x) \oplus \alpha)^{-1})$, Eq. (1) without β , is bijective because $L_{\text{in}}, L_{\text{out}}$ and inversion are. The constant β is then chosen among all 256 values. Candidates are ordered by

$$d(\beta) = \text{SHA256}(\beta_{\text{seed}} \parallel \beta \parallel \ell_{\beta\text{-order}}), \quad \beta \in \{0, \dots, 255\}, \quad (10)$$

read as 256-bit integers, and for each candidate $\mathcal{S}_\beta = T \oplus \beta$ the search counts

$$\text{FP}(\beta) = |\{x : \mathcal{S}_\beta(x) = x\}|, \quad \text{OFP}(\beta) = |\{x : \mathcal{S}_\beta(x) = x \oplus 0\text{xFF}\}|. \quad (11)$$

It returns the first candidate with the smallest FP + OFP in Eq. (11) and stops as soon as the sum is zero (Algorithm 1). XOR with a constant only relabels outputs, so this step leaves nonlinearity, differential uniformity, algebraic degree and boomerang uniformity unchanged. Before use, the implementation checks again that L_{in} and L_{out} are linear and that $\mathcal{S}$ is bijective, and it rebuilds from a new seed if a check fails.

Illustrative session. One session was generated end to end from the source under the hardware stress model: 4,096 shots passed both health tests, and whitening gave 2,048 bytes (ones fraction 0.4935 before whitening and 0.4975 after). The seed gave $\alpha = 0\text{x0F}$, the two linear maps were accepted at the first and second attempts, and the β search stopped at $\beta = 0\text{x49}$ with no fixed or opposite fixed points. Table 3 lists the resulting S-box; the base table T and the raw shots, whitened bytes and seed are in Online Resource 1 and the released data.

Table 3 S-box of the illustrative session (hexadecimal); entry (i, j) is $S(16(i - 1) + j - 1)$.

i/j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	F4	CC	03	71	79	66	AD	31	98	37	4A	8C	A8	45	D9	E2
2	A7	F7	5B	5E	BE	27	A6	EF	8A	34	82	39	62	11	FB	26
3	46	56	53	EC	3D	F1	16	20	D1	E8	67	ED	DB	70	4D	44
4	A1	C9	C5	3A	2E	76	E1	C1	2F	63	D4	41	AB	48	D8	21
5	97	69	15	5F	96	7D	C2	CD	EA	9F	2A	E4	90	F2	DA	1D
6	6E	DF	3F	40	10	A9	29	83	42	9B	17	F9	9A	D7	B7	3B
7	64	BF	E0	B6	07	01	5D	06	FD	DE	FE	6C	CA	BC	04	94
8	6A	C3	D2	43	A5	7B	B5	9E	84	77	30	9C	A2	D5	86	52
9	B3	5A	00	F0	C4	1C	C0	81	1A	91	93	58	23	6D	DC	BA
10	72	2B	80	51	F5	09	2C	02	8B	CE	7A	2D	F6	92	73	FC
11	4C	F8	13	74	88	EB	28	C7	EE	BB	CB	A0	B0	3E	25	0D
12	50	1B	8F	3C	FF	49	4E	C8	75	0C	0B	54	38	33	7F	4F
13	59	1F	6B	E9	AE	4B	95	C6	18	68	7E	FA	A4	8E	E7	AF
14	A3	DD	05	E5	AC	08	B2	65	D3	22	8D	B1	D6	0A	D0	E3
15	47	78	85	12	B8	1E	24	89	55	CF	14	99	0E	F3	E6	87
16	AA	9D	57	32	60	7C	0F	B9	35	5C	B4	61	19	36	6F	BD

3.6 Attacker model

Knowledge and assumptions. Following Kerckhoffs’s principle, security rests on the secrecy of K and of the parameters derived from it. The session S-box $\mathcal{S}$ is a function of K and is secret session material. The attacker knows the circuit, the SHAKE-256 and SHA-256 derivation, the β procedure and the field modulus, and therefore knows that $\mathcal{S}$ lies in the affine class of I and knows the public tables of I . The attacker does not know K , the derived parameters or $\mathcal{S}$. SHAKE-256 and SHA-256 are modeled as random oracles. K , or the secret it is derived from, reaches both parties over a channel that authenticates it and keeps it secret; in QSD-Audio this is an ML-KEM exchange signed with ML-DSA (Section 8), and compromise of that channel is the most likely practical attack. The attacker queries the cipher with chosen plaintexts and, in the side-channel experiment of Section 6.3, observes simulated leakage of one S-box lookup.

What is precomputed and what must be repeated. Against a fixed table, an attacker computes once the difference and linear approximation tables and their best entries, a catalogue of trails that reach the bound, and side-channel templates of the table’s outputs. After that, each new session costs only data collection. Session diversification aims to make the table-specific part of this work invalid at every rotation, so that it must be redone for each session, while the part that depends only on invariants stays valid by design (Section 3.7). We consider two variants. In the main one the table is secret. In a weaker one the attacker learns $\mathcal{S}$ after a session, for example through a memory disclosure; the table then only changes, and the question is whether what was learned about one session helps against the next.

Who sees the quantum output. A gate-based QRNG is usually a cloud service. Its operator, here IBM, receives every shot, and so does anyone who can read the job results in the user’s account. From the shots they can recompute W and, by Eq. (7), the seed K and the session S-box. Used alone with a cloud device, QSD-SBox therefore trusts the operator as much as the two endpoints. There are two remedies. The first is

a device under the user’s own control. The second mixes the quantum output with a secret that never leaves the endpoint, $K = \text{SHAKE256}(P \parallel \text{Len}(W) \parallel W \parallel R)$ with a 32-byte R from a local generator; in the random oracle model K then stays secret unless the adversary knows both W and R , so a curious operator and a flawed local generator must coincide to break it. The released code offers this option; the experiments in this paper use Eq. (7) without R . QSD-Audio avoids the question: its S-box key K_{SBox} is derived from the ML-KEM shared secret, and the quantum output enters only as public diversification, so a provider that sees every shot learns nothing about the session keys (Section 8.1).

Invariants. The following statement makes the guarantee for each session precise.

Proposition 1 *Let $I(x) = x^{254}$ on $GF(2^8)$ and let $A_{\text{in}}(x) = L_{\text{in}}(x) \oplus \alpha$ and $A_{\text{out}}(y) = L_{\text{out}}(y) \oplus \beta$ with $L_{\text{in}}, L_{\text{out}} \in GL(8, \mathbb{F}_2)$. Then $\mathcal{S} = A_{\text{out}} \circ I \circ A_{\text{in}}$ of Eq. (2) has nonlinearity 112, differential uniformity 4, algebraic degree 7 in every nonzero component, and boomerang uniformity 6, for every choice of $L_{\text{in}}, L_{\text{out}}, \alpha, \beta$.*

Proof For masks a, b , substituting $x = A_{\text{in}}^{-1}(z)$ in the Walsh sum of $\mathcal{S}$ gives $|W_{\mathcal{S}}(a, b)| = |W_I(a', b')|$ with $a' = (L_{\text{in}}^{-1})^{\top} a$ and $b' = L_{\text{out}}^{\top} b$, a bijection of nonzero mask pairs. The extended Walsh spectrum is therefore that of I , whose largest absolute value is 32 (Nyberg 1994), so $NL(\mathcal{S}) = 128 - 32/2 = 112$. Since $L_{\text{in}}, L_{\text{out}}$ are linear, α and β cancel in differences and

$$\text{DDT}_{\mathcal{S}}(\Delta x, \Delta y) = \text{DDT}_I(L_{\text{in}}(\Delta x), L_{\text{out}}^{-1}(\Delta y)), \quad (12)$$

so the difference table of $\mathcal{S}$ is a relabeling of that of I , whose largest entry for $\Delta x \neq 0$ is 4 (Nyberg 1994). Each component $b \cdot \mathcal{S}$ equals $b' \cdot I \circ A_{\text{in}}$ plus a constant, and composing a Boolean function with an invertible affine map does not change its degree; every nonzero component of I has degree 7. Finally, boomerang uniformity is invariant under affine equivalence, and for inversion in $GF(2^n)$ with $n \equiv 0 \pmod{4}$ it equals 6 (Boura and Canteaut 2018). $\square$

Proposition 1 also covers the linear bias (the largest Walsh value fixes it) and BIC-NL, since the XOR of two output bits is a nonzero component. SAC and BIC-SAC are not affine invariants; they vary between sessions and are reported empirically in Section 5.

3.7 Invariant and table-specific information

Equation (12) and its counterpart for the Walsh spectrum in the proof of Proposition 1 split what an attacker can know about a session S-box into two parts (Table 4). The *invariant* part is the same in every session and public, because it is that of I : the values of nonlinearity, differential uniformity, degree and boomerang uniformity, the multiset of entries of each table, and the bounds on characteristics that follow from them. The *table-specific* part is the position of each entry: which difference pair reaches 4, which pair of masks gives the largest bias, which trail reaches the bound, and which output belongs to which input, which is what a side-channel template of the S-box output models. The secret maps of each session move these positions. Session diversification is aimed at the table-specific part only; an attack that needs nothing beyond the

invariants works equally well against every session, and so does an attack on leakage of the S-box input $x = p \oplus k$, which does not involve the table at all.

Table 4 What an attacker may know about a session S-box, and whether it is the same in every session.

Information	Same in every session	Evidence
NL , δ , degree, boomerang uniformity; LP and DP	Yes	Prop. 1; Sec. 5.2
Multiset of entries of the difference, Walsh and boomerang tables	Yes	Eq. (12)
Minimum number of active S-boxes; bound $(4/256)^k$	Yes	Table 9
Leakage of the S-box input $x = p \oplus k$	Yes	Table 10
Positions of the best characteristics	No	Table 10
Masks of the best linear approximations	No	Table 10
Concrete trails that reach the bound	No	Table 10
Output $S(x)$ of each input; templates of its leakage	No	Table 10
Fixed points, SAC, BIC-SAC	No	Sec. 5.2

Size of the class. The class $\{A_{\text{out}} \circ I \circ A_{\text{in}}\}$ has $|AGL(8, \mathbb{F}_2)|^2 / |\text{Aut}(I)|$ members, where $\text{Aut}(I)$ is the group of affine pairs (A, B) with $I \circ A = B \circ I$. An exhaustive search that propagates linearity, in the manner of the linear equivalence algorithm of Biryukov et al. (2003), finds exactly 2,040 such pairs, the maps $x \mapsto cx^{2^i}$ with $c \neq 0$ and $0 \leq i < 8$, and none with a nonzero translation. The class therefore holds $|AGL(8, \mathbb{F}_2)|^2 / 2040 \approx 9.2 \times 10^{38} \approx 2^{129.4}$ distinct S-boxes. The derivation draws L_{in} , L_{out} and α uniformly, so two sessions share a table only with negligible probability; within every experiment of this study, all session tables were distinct.

Recovering the affine maps. If the attacker can evaluate $\mathcal{S}$ on its own, the affine equivalence algorithms of Biryukov et al. (2003) and Dinur (2018) recover A_{in} and A_{out} from I with modest effort for 8-bit S-boxes, and when the whole implementation is exposed, as in white-box AES, secret affine encodings around the AES S-box are removed by the attack of Billet et al. (2004). QSD-SBox relies on the table being reachable only through the keyed cipher, where every S-box input is masked by a round key and every output passes through ShiftRows, MixColumns and further rounds; neither algorithm applies directly in that setting. Rounds remain necessary: Tiessen et al. (2015) recover both the key and a secret S-box of AES reduced to four, five and six rounds with integral attacks of time and data complexity $2^{17}/2^{16}$, $2^{38}/2^{40}$ and $2^{90}/2^{64}$, so QSD-BC keeps all ten rounds of AES-128. In the weaker model, where the table of one session leaks, the attacker holds that table only, and the next session draws a new member of the class. In QSD-Audio the seed is K_{SBox} , derived from the ML-KEM shared secret and never sent, while the value Q_{ext} derived from the quantum source is public and only diversifies the derivation.

4 Characterization of the seed source

The algebraic guarantees hold for any seed, so this section asks a narrower question: whether the OTQuSbox source, with whitening and conditioning, delivers seeds of the quality it is credited with, in simulation and on hardware.

4.1 Method

Why SP 800-90B and Dieharder. Applied to the raw stream of the earlier circuit with rotation and Toffoli gates, SP 800-22 failed 10 of 14 eligible tests at a readout bias of less than one percentage point, which showed that the stream was biased but not how much entropy it held. SP 800-90B answers the second question directly and Dieharder detects small departures at data volumes that SP 800-22 does not reach, so we replace SP 800-22 with these two tools.

What simulated data can show. Qiskit Aer ([Javadi-Abhari et al. 2024](#)) computes the output distribution of the noisy circuit and then draws shots from it with a classical pseudorandom generator. Tests of simulated shots therefore measure two things: how far the noise model moves the distribution away from uniform, and the quality of the sampler. They cannot show quantum unpredictability. We use the simulations only for the first purpose, as a controlled way to check whether whitening removes the distortion a given noise model introduces. The physical evidence is the two hardware runs of Section 4.4.

Conditions. Table 5 lists the configurations: the OTQuSbox circuit under the published model and under the stress model, raw and whitened; the circuit with Hadamard gates only; and the CAQ variant of Eq. (6), raw and whitened (the seed stream is described at the end of this subsection). For each circuit we computed the exact distribution from the density matrix and the readout confusion matrix. A χ^2 test of 10^7 Aer shots against this distribution gave p -values between 0.15 and 0.68 for all five circuit and model pairs, so the exact distribution and the simulator agree.

SP 800-90B. We ran NIST’s reference implementation ([National Institute of Standards and Technology 2024c](#)) of the non-IID estimators, the IID tests and the restart test on samples of eight bits taken directly from Aer: five independent datasets of 10^6 samples per configuration, and for the restart test 1,000 independent Aer executions of 1,000 samples each.

Dieharder. A full Dieharder 3.31.1 run (`-a -k 2 -Y 1`: all tests, exact Kolmogorov–Smirnov p -values, and WEAK results retested with more samples until they resolve) reads about 250 GB, and more when WEAK results are retested (up to 521 GB in our runs). Reading a finite file of that size would be impractical, and a smaller file would be rewound and reused, which makes the results unreliable. We therefore streamed the data through standard input. Aer cannot produce this volume, so a C program draws bytes independently from the exact distribution of each configuration, using the alias method with 2^{-56} resolution and OpenSSL’s CTR_DRBG ([Barker and Kelsey 2015](#)) as its randomness, and XORs two (one whitening pass) or four (two passes) draws for the whitened configurations. This is the same operation Aer performs internally. A χ^2 test on 2×10^8 output bytes per configuration against

691 the exact distribution or its XOR convolution gave p -values between 0.05 and 0.96. A
692 control run fed the DRBG output directly.

693 *The seeds themselves.* QSD-SBox uses only the 128-byte seeds K , never the shots,
694 so we also tested a stream of seeds. Each seed is made as in Section 3.3 from 4,096
695 shots drawn from the exact distribution of the stress model: health tests (a failing
696 batch is redrawn), XOR whitening and SHAKE-256. A multithreaded C version of
697 this path reproduces the seed of the illustrative session and agrees with the Python
698 code, in seeds and in the verdicts of the health tests, on 2,000 further batches, 300 of
699 them with planted faults. Passing these tests does not show entropy, because SHAKE-
700 256 output passes them for any input that does not repeat. It shows that the seed
701 path works as described and that the seeds carry no visible structure; the evidence for
702 entropy is the assessment of the source.

703 *Tool checks.* Both tools were built from source with MinGW-w64 on Windows. Two
704 counters in NIST’s IID code overflow where `long` has 32 bits, and Dieharder needed
705 binary input and a 64-bit file offset; Online Resource 1, Section S3 lists every change.
706 With these changes the non-IID tool reproduces NIST’s reference output for all 11
707 of its self-test files, the IID tool accepts NIST’s true random sample file and rejects
708 its ring oscillator file, and Dieharder gives identical p -values for an MT19937 stream
709 read internally, from a file or through standard input. NIST’s code built without our
710 patches on Linux gives the same 17 non-IID estimates, to the last printed digit, on
711 the raw and whitened shots of both hardware runs of Section 4.4.

712

713 4.2 Min-entropy (SP 800-90B)

714

715 Table 5 reports, for each configuration, the exact min-entropy and bit bias, the verdict
716 of the IID tests, the estimate of the IID track, and the non-IID assessment over five
717 datasets.

718 All seven configurations, the seeds included, passed the IID tests (chi-square,
719 longest repeated substring and permutation tests with 10,000 rounds), as expected for
720 samples drawn independently. On the IID track the assessment is the most common
721 value estimate with a 99% upper confidence bound, and it orders the configurations
722 as the exact values do: the raw stream under the stress model scores 7.775 bits per
723 byte, against 7.888 after whitening and 7.885 for the exactly uniform published model.
724 About 0.11 bits of the gap to 8 is the confidence bound itself, which is as high as this
725 estimator can go with 10^6 samples. The seeds scored 7.840 because one byte value
726 occurred 4,197 times in the first dataset, against 3,906 expected; the other four seed
727 datasets gave 7.857 to 7.891, and 40 further seed datasets averaged 7.878, the same
728 as 200 datasets of uniform bytes (7.877).

729 The non-IID assessments are lower. They lie between 6.64 and 7.37 for every simu-
730 lated configuration, including the exactly uniform one and the seeds, and NIST’s own
731 sample of hardware random bytes scores 7.23. Two estimators decide the value. The
732 compression estimate on the bit string gives about 7.1 to 7.3, and the prediction esti-
733 mators on 8-bit symbols drop to exactly 6.638 whenever a predictor happens to guess
734 a short run of consecutive symbols (a local probability of 0.0100), which occurred in
735 8 of the 35 datasets. Both are known sources of underestimation in the non-IID track
736 (Zhu et al. 2017). We therefore read the non-IID values as conservative lower bounds

that do not separate the configurations, and we use the smallest of them, 6.638 bits per sample, wherever a bound is needed: the health test cutoffs, the entropy accounting of Section 3.3 and the extractor length. The restart test of SP 800-90B, on 1,000 independent Aer executions of 1,000 samples each, passed for both the raw and the whitened stream at $H_I = 6.638$.

Table 5 SP 800-90B assessment (bits per sample of eight bits). Exact values come from the simulated distribution and are not defined for the seeds, which are SHAKE-256 output; “IID track” is the most common value estimate, valid when all IID tests pass; non-IID values are the median and range over five datasets of 10^6 samples. For the two IBM devices the bias in parentheses is measured, and one dataset per device (10^6 raw shots, 5×10^5 whitened bytes) gives one non-IID value. The last row is NIST’s own reference file of hardware random bytes.

Configuration	Output	Exact $H_{\min}$ (max bit bias)	IID tests	IID track	Non-IID, median [range]
OTQuSbox, published model	raw	8.0000 (0)	pass	7.885	7.147 [6.638, 7.260]
OTQuSbox, stress model	raw	7.8244 (0.0078)	pass	7.775	7.110 [6.638, 7.175]
OTQuSbox, stress model	whitened	7.9977 (1.2×10^{-4})	pass	7.888	7.137 [6.638, 7.314]
Seeds K , stress model	SHAKE- 256	—	pass	7.840	7.226 [6.638, 7.368]
Hadamard only, stress model	raw	7.8797 (0.0052)	pass	7.818	7.153 [6.638, 7.318]
CAQ, stress model	raw	7.9943 (2.5×10^{-4})	pass	7.865	7.174 [7.111, 7.296]
CAQ, stress model	whitened	8.0000 (1.2×10^{-7})	pass	7.880	7.239 [6.638, 7.261]
OTQuSbox, <code>ibm_fez</code>	raw	— (0.0168)	pass	7.715	7.232
OTQuSbox, <code>ibm_fez</code>	whitened	— (0.0012)	pass	7.824	7.017
OTQuSbox, <code>ibm_marrakesh</code>	raw	— (0.0144)	pass	7.590	6.558
OTQuSbox, <code>ibm_marrakesh</code>	whitened	— (0.0015)	pass	7.823	6.961
NIST <code>truerand.8bit</code> (National Institute of Standards and Technology 2024c)	hardware	—	pass	7.865	7.234

4.3 Dieharder

Table 6 lists the eight runs. A full run gives 114 results from 31 test types; we leave out `diehard_sums`, which Dieharder itself rates “Do Not Use” and which in our runs failed even on the exactly uniform stream once enough samples accumulated. The control and the published model, whose output is exactly uniform, passed every test, which shows that the battery, the build and the sampler produce no false alarms at this volume. The raw stream under the stress model failed 80 of 114 results in 20 test types, so the modeled bias of 0.0078 is plainly visible. The raw CAQ stream, whose largest bit bias is only 2.5×10^{-4} , still failed 38, all in tests that count ones or sum values.

One whitening pass cut the failures of the OTQuSbox stream from 80 to 16. All 16 are tests that count ones or sum values (`sts_monobit`, 14 lagged sum tests and one `dab_monobit2` result): at the scale of a full run, 10^{11} bytes, and after the retests of `-Y`

1 pushed this run to 521 GB, the residual bias of about 10^{-4} predicted by Eq. (5) is still detectable. A second pass (four shots per output byte, predicted bias 3.0×10^{-8}) and the combination of CAQ with one pass (1.2×10^{-7}) passed every test, like the control.

For the session seed this residual bias does not matter. K is derived from W through SHAKE-256, and its strength rests on the min-entropy of W , which the bias lowers only from 8 to 7.998 bits per byte (Table 5). Uses that need nearly uniform bits directly, without a hash, should apply a second pass or the Toeplitz extractor of Section 4.4, as QSD-Audio does.

The seeds themselves passed all 114 results. The run read 257 GB, about 2.0×10^9 seeds, each conditioned from its own 4,096 shots under the stress model, and the health tests redrew 1,823 raw batches along the way, a false alarm rate of about 10^{-6} per batch. As explained in Section 4.1, this confirms the seed path and the absence of visible structure in the output that QSD-SBox uses; it adds no evidence of entropy beyond the tests of the source.

798

Table 6 Full Dieharder battery (-a -k 2 -Y 1) on streams read through standard input without reuse. One whitening pass is Eq. (4); two passes XOR four shots. Counts are over the 114 final results of a run, after WEAK results were retested with more samples. Dieharder stops retesting a test once one of its results has failed, so the two WEAK results of raw CAQ (sts_serial, failed at another block length) stay unresolved; diehard_sums (rated “Do Not Use” by Dieharder) is excluded. Numbers in parentheses count results of the same test at different parameters.

Stream	GB read	Passed	Weak	Failed	Failed tests
Control: CTR_DRBG output	247	114	0	0	none
OTQuSbox, published model, raw	246	114	0	0	none
OTQuSbox, stress model, raw	250	34	0	80	20 of the 31 test types
OTQuSbox, stress model, one pass	521	98	0	16	sts_monobit, rgb_lagged_sum (14), dab_monobit2
OTQuSbox, stress model, two passes	247	114	0	0	none
Seeds K , stress model	257	114	0	0	none
CAQ, stress model, raw	270	74	2	38	sts_monobit, sts_serial, rgb_lagged_sum (33), dab_bytedistrib, dab_filltree2, dab_monobit2
CAQ, stress model, one pass	247	114	0	0	none

816

817

4.4 Hardware runs on `ibm_fez` and `ibm_marrakesh`

We ran the OTQuSbox circuit for 10^6 shots on each of two IBM Quantum devices with 156 qubits, `ibm_fez` and `ibm_marrakesh`, each as one job of ten parts of 10^5 shots (261 s and 266 s of QPU time). On both devices the transpiler placed the circuit on physical qubits 0–7, a linear chain, with one CZ gate per CNOT and no swap gates (depth 12). Calibration at the time of the runs gave $T_1 = 53\text{--}206\ \mu\text{s}$ and $161\text{--}292\ \mu\text{s}$, $T_2 = 36\text{--}210\ \mu\text{s}$ and $43\text{--}246\ \mu\text{s}$, and readout errors of 0.0035–0.026 and 0.0046–0.040.

Both devices are harsher than our stress model. The largest bias of a single qubit in the raw shots is 0.0168 on `ibm_fez` and 0.0144 on `ibm_marrakesh`, against 0.0078 in the model, and on both devices most qubits read 1 less often than 0, as relaxation

and asymmetric readout predict. The raw shots of both runs passed all three IID tests of SP 800-90B on 10^6 samples. All 488 batches of 4,096 shots passed the health tests, and taken through the seed path they gave 488 distinct seeds and 488 distinct S-boxes, all with nonlinearity 112, differential uniformity 4, degree 7, boomerang uniformity 6, BIC-NL 112 and no fixed points.

On `ibm_fez` the correlation between consecutive shots is within one standard deviation (10^{-3}) of zero on every qubit (jointly $p = 0.99$), and after whitening (5×10^5 bytes) the bias of every bit agrees with the prediction $2\epsilon^2$ of Eq. (5) within 1.3 standard deviations of sampling noise; the largest is 0.0012, against a noise level of 7×10^{-4} . SP 800-90B assesses the raw data at 7.232 bits per byte and the whitened data at 7.017 (Table 5); the whitened set is half as large, which lowers its non-IID value.

`ibm_marrakesh` shows two effects that the stress model does not contain. First, qubits 3 and 4, the pair joined by a CNOT of the second layer, are correlated within a shot (correlation 0.12, against at most 0.0075 for any pair on `ibm_fez`), so their parity $b_3 \oplus b_4$ has a bias of 0.0604. Eq. (5) predicts that one whitening pass leaves $2 \times 0.0604^2 = 0.00729$; we measured 0.00724. Second, consecutive shots are slightly correlated on two qubits (2.5×10^{-3} and 2.1×10^{-3} at lag 1, jointly $p = 0.03$ over the eight qubits). The correlation appears only at lag 1 and does not come from drift, since $\Pr(1)$ is the same in all ten parts of the job; the IID tests of SP 800-90B do not detect it. SP 800-90B assesses the raw data at 6.558 bits per byte, set by the collision estimate on the bit string, which is sensitive to the correlated pair, and the whitened data at 6.961. The raw value lies slightly below the 6.6 bits used for the health test cutoffs (Section 3.3); recomputed at 6.55 bits the adaptive proportion cutoff is 21 instead of 20, so the deployed tests are slightly stricter than needed on this device, which can only add false alarms. The seed accounting of Section 3.3 uses the whitened stream, whose smallest assessment is still 6.638 bits per byte.

These runs, 10^6 shots on each of two devices on one day, meet the sample size of the SP 800-90B estimators (Section 9 lists what they do not cover). They also show why QSD-SBox conditions the whitened stream with SHAKE-256 instead of using it directly: one pass leaves a parity bias of 7×10^{-3} on `ibm_marrakesh`, which a second pass would reduce to about 10^{-4} .

Toeplitz extraction. Applications that need nearly uniform bits with a stated bound, such as QSD-Audio, apply a Toeplitz extractor (Krawczyk 1994). For an n -bit input with min-entropy rate h and a public uniform seed of $n + m - 1$ bits, the output of length $m = \lfloor nh - 2\log_2(1/\epsilon) \rfloor$ is ϵ -close to uniform by the leftover hash lemma (Impagliazzo et al. 1989); we use $\epsilon = 2^{-64}$. On the raw shots of the two runs, with h taken from their SP 800-90B assessments, the extractor keeps 90.4% and 82.0% of the input bits, and in six extractions per device with independent extractor seeds the ones fraction ranged from 0.4998 to 0.5005, within 2.4 standard deviations of $1/2$.

5 S-box evaluation

5.1 Illustrative session

All criteria were computed exhaustively for the S-box of Table 3 with one code base that is also used for the AES row of Table 8.

Bijectivity and nonlinearity. The 256 outputs are distinct, as the composition of three bijections guarantees (Adams and Tavares 1990). The nonlinearity $NL(f) = \min_g d_H(f, g)$ over affine functions g is 112 for each of the eight coordinate functions and for every nonzero component, the value of the AES S-box (Nyberg 1994).

SAC and BIC. The strict avalanche criterion (SAC) (Webster and Tavares 1986) asks that flipping one input bit flips each output bit with probability 1/2. The SAC matrix (Table 7) has mean 0.5083, mean absolute deviation 0.0254 from 0.5 and largest deviation 0.0625. The bit independence criterion (BIC) (Adams and Tavares 1990) applies the same tests to the XOR of every pair of output bits: BIC-NL is 112 for all pairs and the mean BIC-SAC is 0.4994 (matrix in Online Resource 1). The interval $0.45 < \text{SAC} < 0.55$ often quoted in the literature is a design guideline, not a statistical test.

Table 7 SAC matrix of the illustrative S-box (row: flipped input bit; column: output bit).

In/Out	0	1	2	3	4	5	6	7
0	0.5469	0.5312	0.5156	0.5469	0.5156	0.5312	0.5469	0.5156
1	0.5469	0.5156	0.5000	0.4688	0.5156	0.5312	0.4531	0.5156
2	0.4844	0.5156	0.5156	0.5312	0.5312	0.5469	0.5312	0.5156
3	0.5312	0.5156	0.4688	0.5156	0.5312	0.4844	0.5156	0.5156
4	0.4844	0.5312	0.4531	0.4531	0.4375	0.5000	0.5156	0.5156
5	0.5000	0.5000	0.5312	0.4844	0.5312	0.4688	0.5469	0.4531
6	0.5156	0.4844	0.5000	0.5312	0.4844	0.4688	0.4844	0.4688
7	0.5312	0.5469	0.5156	0.5469	0.5000	0.4844	0.4844	0.5312

Linear and differential properties. As is usual in the S-box literature, we report the linear bias as LP and the differential probability as DP (Selçuk 2008):

$$\begin{aligned}
 LP &= \max_{a,b \neq 0} \left| \frac{\#\{z : a \cdot z = b \cdot \mathcal{S}(z)\}}{2^n} - \frac{1}{2} \right|, \\
 DP &= \max_{\Delta z \neq 0, \Delta y} \frac{\#\{z : \mathcal{S}(z) \oplus \mathcal{S}(z \oplus \Delta z) = \Delta y\}}{2^n}.
 \end{aligned} \tag{13}$$

In Eq. (13), the largest absolute Walsh coefficient is 32, so $LP = 32/512 = 0.0625$, and the differential uniformity is 4, so $DP = 4/256 = 0.015625$; both equal the AES values. Every coordinate function has algebraic degree 7, the boomerang uniformity (Cid et al. 2018) is 6, and there are no fixed or opposite fixed points. The output histogram is flat because $\mathcal{S}$ is a permutation, so its entropy says nothing about the seed (Dimitrov 2020). Unlike the invariants, the absence of fixed points is enforced by the β search; we know of no attack that exploits fixed points in this setting and do not assign the property a security value.

5.2 Statistics over many sessions

We generated 1,000 sessions end to end from the OTQuSbox source under the stress model, each with its own 4,096 shots, health tests, whitening and conditioning. Every

S-box had nonlinearity 112, differential uniformity 4, algebraic degree 7 in all coordinates, BIC-NL 112 for every pair of output bits, and no fixed or opposite fixed points; the first 100 had boomerang uniformity 6. All 1,000 S-boxes were distinct. The mean SAC was 0.5019 (standard deviation 0.0038, range 0.4912 to 0.5132) and the mean BIC-SAC 0.5020 (standard deviation 0.0020, range 0.4955 to 0.5083). The two linear layers took 3.53 and 3.34 draws on average (medians 3 and 2), in line with the expected $1/0.2899 \approx 3.45$. A second run of 1,000 sessions seeded from the operating system’s CSPRNG gave the same results: all invariants held, all S-boxes were distinct, the mean SAC was 0.5019 (standard deviation 0.0039) and the mean BIC-SAC 0.5020 (standard deviation 0.0019). The CSPRNG baseline isolates the algebraic pipeline: the invariants do not depend on where the seed comes from, as Proposition 1 requires, so the claim made for the quantum source concerns only the physical origin of the seed. The 488 sessions seeded by the two hardware runs of Section 4.4 agreed as well (all invariants held, all S-boxes were distinct, mean SAC 0.5019 and BIC-SAC 0.5019).

5.3 Comparison with other S-boxes

Table 8 compares the illustrative S-box with AES, OTQuSbox and recent designs. The AES row was recomputed with our code. Averaging the SAC of single output bits instead of their pairwise XOR gives a BIC-SAC of 0.5049 for AES; the correct value, 0.5046, agrees with the literature. The equality with AES in NL, BIC-NL, LP and DP is structural (Proposition 1); only SAC and BIC-SAC vary with the session. The chaotic and heuristic designs report lower nonlinearity and higher LP and DP. Those values are reported by their authors and were not recomputed, and LP, DP and BIC are not always defined identically across papers, so the comparison is indicative.

Table 8 Comparison of S-box designs. [†]Recomputed with our code; the other rows are the values reported by their authors.

S-box	NL	SAC	BIC-NL	BIC-SAC	LP	DP
QSD-SBox (Table 3) [†]	112	0.5083	112	0.4994	0.0625	0.015625
AES (Daemen and Rijmen 2020) [†]	112	0.5049	112	0.5046	0.0625	0.015625
OTQuSbox (Malik et al. 2026)	108	0.4998	103.71	0.5021	0.0625	0.0391
Lu et al. (2019)	106.3	0.5050	103.8	0.4994	0.1250	0.0390
Ali et al. (2022)	103.5	0.4962	104.64	0.4985	0.1028	0.03125
Malik et al. (2023)	106.75	0.5027	103.6	0.5020	0.1328	0.0391
Zahid et al. (2023)	111.75	0.5000	103.9	0.5000	0.1250	0.0390
Singh et al. (2024)	104.0	0.5103	104.0	0.5115	0.0833	0.04688
Saini et al. (2024)	108.75	0.4971	108.7	0.4970	0.1015	0.03125

6 Security within and across sessions

This section places the session S-box in a full cipher and asks two questions: whether every session keeps the differential bounds of AES, and whether table-specific information profiled on one session transfers to another. Unless stated otherwise, sessions

are seeded from a CSPRNG; by Proposition 1 the results do not depend on the seed source.

6.1 The cipher QSD-BC

QSD-BC has the AES-128 parameters and structure: 128-bit block and key, ten rounds, no MixColumns in the last round. The session S-box replaces the AES S-box in SubBytes of every round and in SubWord of the key schedule; RotWord and the round constants are unchanged, and decryption uses $\mathcal{S}^{-1}$ (listings in Online Resource 1). With the standard S-box, the implementation reproduces the known-answer test of FIPS-197 Appendix B (National Institute of Standards and Technology 2001) exactly, so the S-box is the only variable under test.

6.2 Bounds within a session

Trails over two rounds. Because δ is invariant and ShiftRows and MixColumns do not depend on the session, the best possible probability of a trail with k active S-boxes, $(\delta/256)^k$, is the same for all sessions; whether a session reaches it depends on its difference table. Using each session’s own table, an exhaustive branch-and-bound search over all 255 nonzero input differences in one byte through one MixColumns column (two rounds) found, in 5 sessions, a best trail with exactly 5 active S-boxes, the branch number of AES (Daemen and Rijmen 2020), and probability $(4/256)^5 = 9.3132 \times 10^{-10}$, exactly the bound. Repeating the search on 500 further sessions, $\delta = 4$, five active S-boxes and the exact bound held in all 500, so no weak instance appeared among 505 sessions.

Active S-box bound. For a bijective S-box, a byte has a nonzero input difference exactly when it has a nonzero output difference, and round keys cancel in differences. The minimum number of active S-boxes over r rounds therefore depends only on the linear layer and is the same for AES and every session. We computed it with a mixed-integer linear program that encodes the branch number 5 of each MixColumns column (Mouha et al. 2012). It matches the published AES values for all ten round counts (Table 9). With $\delta = 4$ for every session, $(4/256)^k$ bounds the probability of any single characteristic in the single-key setting.

Table 9 Minimum number of active S-boxes k over r rounds (MILP; identical to the published AES values) and the characteristic bound $(4/256)^k$, which holds for every session.

r	k	$(4/256)^k$	r	k	$(4/256)^k$
1	1	1.6×10^{-2}	6	30	6.5×10^{-55}
2	5	9.3×10^{-10}	7	34	3.9×10^{-62}
3	9	5.6×10^{-17}	8	50	4.9×10^{-91}
4	25	7.0×10^{-46}	9	51	7.7×10^{-93}
5	26	1.1×10^{-47}	10	55	4.6×10^{-100}

Tightness and boomerang uniformity. The search over 505 sessions shows that the bound is reached at two rounds in every session tested. We implemented the

boomerang connectivity table (Cid et al. 2018), checked it against the published boomerang uniformity of AES (6), and computed it for 1,000 independent sessions: all had boomerang uniformity 6, as Proposition 1 predicts.

6.3 Cross-session transfer

Method. We generated 1,000 sessions and took every ordered pair (A, B) with $A \neq B$, 999,000 pairs in all. For each pair, the attacker profiles session A with full knowledge of its table and applies the profile unchanged to session B . Three profiles are tested: the 255 best characteristics of A (in each nonzero row of its difference table, the pair $(\Delta x, \Delta y)$ with count 4), its 1,275 best linear approximations (Walsh value ± 32 , correlation $\pm 1/8$), and 255 trails over two rounds through one MixColumns column that reach the bound $(4/256)^5$ in A , one for each input difference. The control is a static S-box, where a profile built in one session meets the same table in the next and every measure takes its largest value. The chance level is that of a random cell of the tables of I , which Eq. (12) predicts for a fixed position in an independent session. Two of the four S-boxes of the second round receive the same difference, because MixColumns maps $(d, 0, 0, 0)$ to $(2d, d, d, 3d)$, so a trail depends on four distinct cells; the chance levels for trails account for this.

Differential, linear and trail profiles. Table 10 shows every transferred profile at the chance level. The best characteristic of A has probability 0.00392 in B on average, against $1/256 = 0.00391$ for an arbitrary pair and $4/256$ in A itself; it keeps its count of 4 in 0.39% of the pairs, which is the share of such cells in any row. The linear approximations behave in the same way, and the difference and Walsh tables of two sessions are uncorrelated. Of the 2.5×10^8 transferred trails, none reached the bound in the target session and 6.2% had a nonzero probability at all; on average a transferred trail holds with $1/490$ of the probability it had in its own session.

Side-channel profiles, with a static S-box as the control. This experiment asks whether a side-channel profile built in one session recovers the key of the next session. The leakage is simulated. An 8-bit device computes SubBytes of the first round for the 16 key bytes one after another, and each byte leaks two samples: a sum of the bits of the S-box output with unequal weights, and the Hamming distance of the register change from the input $x = p \oplus k$ to the output $\mathcal{S}(x)$, each with Gaussian noise of standard deviation $\sigma \in \{1, 2, 4\}$. In session 1, with a known key, the attacker builds Gaussian templates (Chari et al. 2003) for every value of x from 25,600 traces of a clone device. In session 2, under a new secret key, it ranks the 256 guesses of every key byte by likelihood. Over 100 pairs of sessions and the three noise levels, the profile of session 1 recovered the full 16-byte key of session 2 in 90% of trials with 20, 50 and 200 attack traces when both sessions used the static AES S-box. When they used independent QSD-SBox tables, the same profile ranked on average 0.02 to 0.07 of the 16 key bytes first with 2,000 traces, against 0.06 for guessing, and never recovered the full key. A profile built on a clone that runs the target session’s own table recovered the full key with 20, 50 and 200 traces, as fast as for AES, so the loss comes from the change of table and not from a weaker attack. These results hold for the simulated leakage model; they show the loss of cross-session transfer, not resistance to side-channel attacks in general. A simpler study of one key byte, with a Hamming

Table 10 Cross-session transfer of table-specific information from session A to session B . Differential, linear and trail values are means over all 999,000 ordered pairs of 1,000 sessions. Template attack values are over 100 pairs of sessions with simulated leakage of all 16 key bytes; the row on input leakage is from the study of one key byte (Online Resource 1, Table S16). In the control the S-box is static, so a profile built in one session meets the same table in the next.

Profile built on session A , measured on the target	Static S-box (control)	Session B	Chance level
<i>Differential: the 255 best characteristics of A</i>			
Probability of the best characteristic	0.01563	0.00392	0.00391
Best characteristics that are still best	100%	0.39%	0.39%
Correlation of the difference tables	1	0.0000	0
<i>Linear: the 1,275 best approximations of A</i>			
Mean absolute correlation	0.125	0.053	0.053
Best approximations that are still best	100%	1.96%	1.96%
Correlation of the absolute Walsh tables	1	0.0000	0
<i>Trails over two rounds: 255 per session, 2.5×10^8 transferred</i>			
Trails with nonzero probability	100%	6.22%	6.06%
Trails that reach the bound $(4/256)^5$	100%	0 of 2.5×10^8	$2.3 \times 10^{-8}\%$
Mean probability relative to the bound	1	1/490	1/504
<i>Simulated template attack on all 16 key bytes, $\sigma = 2$, 25,600 profiling traces, 100 session pairs</i>			
Key bytes ranked first (of 16), 100 attack traces	16.00	0.07	0.06
Key bytes ranked first (of 16), 2,000 attack traces	16.00	0.06	0.06
Full key recovered, 2,000 attack traces	100%	0%	2^{-128}
Attack traces for 90% success on the full key	50	not reached with 2,000	–
Input leakage, one key byte: attack traces for 90% success	100	100	–

weight model and a correlation attack (Brier et al. 2004) as well as templates, gives the same pattern and shows the limit of the result: when the device leaks the S-box input x , which does not involve the table, a profile from another session transfers fully (Online Resource 1, Tables S16 and S17).

Setup cost per session. For a static table, a profile is built once and every later session costs only the attack traces. At $\sigma = 2$, 25,600 profiling traces and 50 attack traces amount to 306 traces per session over 100 sessions. With session S-boxes, a profile from another session is of no use, so each session needs its own profile, and building one requires a device that runs that session’s secret table under a known key. The differential profile shows the same pattern. Even when the attacker may query the S-box directly, rebuilding its difference table and confirming its best characteristic took 0.006 to 0.007 s in total for a static S-box over $N = 1$ to 50 sessions, and grew linearly to 0.343 s for session S-boxes, a factor of 57.7 at $N = 50$ (0.93 at $N = 1$). These figures measure the work that can no longer be spread over many sessions; the strength of each single session is bounded in Section 6.2 and is the same for all sessions.

7 Cost of a session

This section measures what session diversification costs: the time to build a session S-box, the quantum hardware needed for its seed, and the cipher’s throughput.

Generation cost. Table 11 gives the time of each stage on one platform (Intel Core i5-1145G7, 4 cores, 7.7 GB RAM, Windows 11, Python 3.14, Qiskit 2.5 and Aer 0.17). In Python the algebraic stages take 27.4 ms on average (p95 56.5 ms), mostly the rejection sampling of the two linear layers. The quantum stage, 4,096 shots simulated from the density matrix, followed by the health tests, whitening and conditioning, takes 206 ms. Drawing 4,096 shots instead of 100,000 cuts this stage from 3.45 s, while the input to SHAKE-256 still carries far more min-entropy than its security level (Section 3.3).

Cost on quantum hardware. On a real device the quantum stage is set by the device and its queue, not by the simulator. In the two runs of Section 4.4, 10^6 shots used 261 s and 266 s of QPU time, about 0.26 ms per shot, so the 4,096 shots of one session take about 1.1 s of QPU time. In both runs the elapsed time of the job exceeded its QPU time by about 37 s although the device queues were nearly empty, and queues on shared devices can take far longer. Two consequences follow for deployment. First, shots should be drawn in batches, not per session: that one job gave 244 sessions, and IBM’s open plan, with 10 minutes of QPU time per month, covers about 560 sessions at this rate, while many small jobs would each add their own overhead and wait. Second, a session should never wait for the device. Whitened batches can be kept in a local pool, each used once and then deleted, and when the pool is empty a seed from the local CSPRNG keeps every invariant (Proposition 1) at the cost of the physical origin for that session. In standalone use a pool of shots determines future seeds and must be protected like key material; with the local secret R of Section 3.6 it need not be.

Table 11 S-box generation time by stage (500 sessions for the algebraic stages and 200 for the quantum stage; Python unless marked C).

Stage	Mean	Median	p95
Parameter derivation ($L_{\text{in}}, L_{\text{out}}, \alpha$)	23.8 ms	20.3 ms	53.0 ms
Base table T	3.1 ms	3.0 ms	3.6 ms
β search	0.5 ms	0.4 ms	0.8 ms
Algebraic stages	27.4 ms	24.0 ms	56.5 ms
Aer simulation, 4,096 shots	199.2 ms	195.7 ms	233.0 ms
Health tests ^a	6.6 ms	2.2 ms	2.6 ms
Whitening and SHAKE-256	0.05 ms	0.05 ms	0.07 ms
Quantum stage	205.8 ms	198.4 ms	235.5 ms
Algebraic stages, C (10,000 sessions)	0.136 ms	0.128 ms	0.184 ms

^aThe mean includes importing the statistics library, which happens once, on the first call.

Throughput. In the same pure Python implementation, throughput on a 512 KiB payload is the same for the standard S-box (0.041 MB/s encryption, 0.022 MB/s decryption) and a session S-box (0.042 and 0.023 MB/s), as expected of a table lookup.

1151 The setup, done once per session, differs: 0.06 ms for key expansion with the standard
1152 S-box against 25.0 ms including the construction of the session S-box.

1153 *Seed synchronization.* ML-KEM-768 ([National Institute of Standards and Technol-](#)
1154 [ogy 2024b](#)) establishes the shared secret without sending the S-box, and the secret is
1155 fed through the SHAKE-256 expander. Unlike Diffie–Hellman, which Shor’s algorithm
1156 breaks, ML-KEM is designed to resist quantum attack. It sends 2,272 bytes (1,184-byte
1157 key and 1,088-byte ciphertext) and takes 14.5 ms (median) in pure Python. Ephemeral
1158 X25519, as used in OTQuSbox ([Malik et al. 2026](#)), is kept only as a classical reference
1159 (64 bytes, 0.14 ms).

1160 *Compiled implementation.* To separate the cost of the design from the cost
1161 of Python, we also implemented the construction, QSD-BC and the audio frame
1162 operation in C (`fast/qsdfast.c` in the released code). It uses OpenSSL 3.6 for
1163 SHAKE-256, SHA-256 and KMAC256, and a 32-bit table implementation of QSD-
1164 BC whose four lookup tables are rebuilt from each session S-box. It agrees exactly
1165 with the Python reference: identical S-boxes, α , β and attempt counts for 300 random
1166 seeds, identical ciphertexts for 300 random blocks, and identical ciphertexts and tags
1167 for 100 random audio frames; with the standard S-box it passes the FIPS-197 known-
1168 answer test. On the platform above, a complete S-box construction takes 136 μ s on
1169 average (median 128 μ s, p95 184 μ s): 24 μ s for the parameters and 111 μ s for the
1170 β search, which is dominated by 256 SHA-256 calls. Building the four lookup tables
1171 takes 0.4 μ s. In counter mode QSD-BC runs at 294 MB/s with a session S-box and
1172 at 294 MB/s with the standard AES S-box, so the choice of table costs nothing in
1173 throughput; OpenSSL AES-128-CTR with the AES instructions reaches 4,981 MB/s.
1174 Deriving an epoch S-box, including four KMAC256 calls, takes 144 μ s, and protecting
1175 one 20 ms audio frame (counter mode and KMAC256 tag) takes 12.3 μ s, a real-time
1176 factor of 6.1×10^{-4} (OpenSSL AES-128-GCM: 0.74 μ s and 3.7×10^{-5}).

1177

1178 8 QSD-Audio: a feasibility case study

1179

1180 QSD-Audio tests whether session S-boxes can be deployed end to end: two endpoints
1181 must derive the same S-boxes, rotate them in step, and protect audio frames without
1182 noticeable delay. It combines established components and is a feasibility study of
1183 session diversification, not a new secure audio protocol. Its handshake and frames are
1184 nevertheless verified formally (Section 8.3).

1185

1186 8.1 Architecture and handshake

1187

1188 QSD-Audio has four layers (Fig. 3). The quantum layer runs the OTQuSbox QRNG,
1189 the health tests, whitening and a Toeplitz extractor. The protocol layer runs an authen-
1190 ticated post-quantum handshake. The algebraic layer is QSD-SBox without change.
1191 The application layer rotates the S-box per epoch and protects each audio frame. The
1192 quantum device is needed only when a session is established or rekeyed, not per frame.
1193 In deployment the quantum layer would draw from a pool filled in batches (Section 7);
1194 because Q_{ext} is public diversification, the pool needs no secrecy, and a local CSPRNG
1195 can stand in when it is empty.

1196

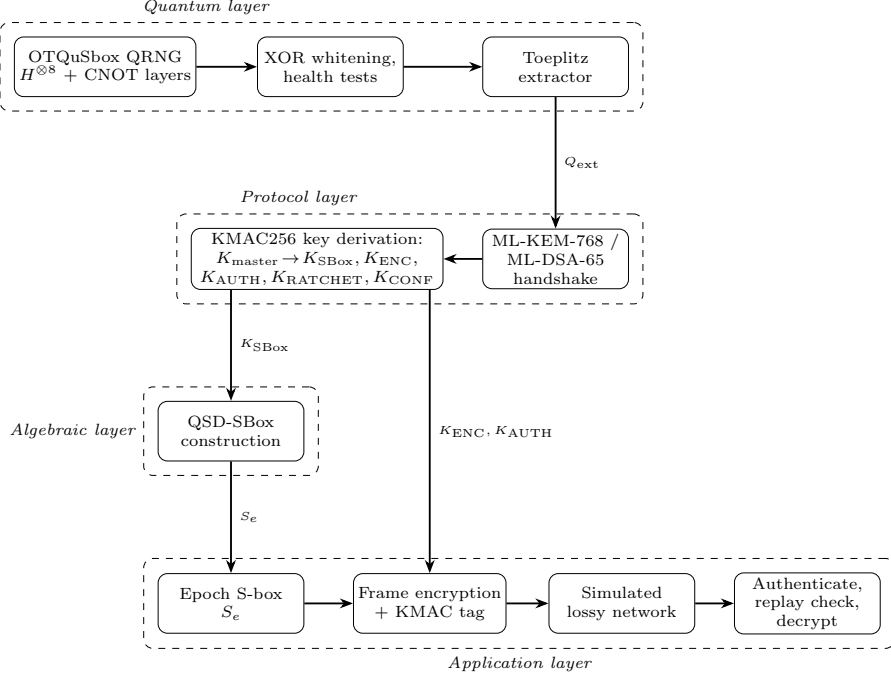


Fig. 3 QSD-Audio layers and the values passed between them: the extracted quantum output Q_{ext} , the subkeys derived with KMAC256, and the epoch S-box S_e . The quantum device is used only at session establishment or rekeying.

The handshake uses ML-KEM-768 ([National Institute of Standards and Technology 2024b](#)) and ML-DSA-65 ([National Institute of Standards and Technology 2024a](#)), both at NIST security category 3. Bob sends the signed message

$$M_1 = v \| \text{ID}_B \| \text{ek}_B \| N_B \| \text{SID}, \quad (14)$$

where v is the protocol version and ek_B the encapsulation key of an ephemeral ML-KEM key pair, signed with his long-term ML-DSA key. Alice verifies M_1 of Eq. (14), runs the quantum layer to obtain Q_{ext} , and encapsulates to obtain (ct, SS) . She sends a fixed 32-byte commitment Q_{salt} to Q_{ext} in the signed message

$$M_2 = \text{ID}_A \| \text{ct} \| Q_{\text{salt}} \| N_A \| \text{AudioParameters}, \quad (15)$$

where `AudioParameters` fixes every negotiated value (codec, sample rate, channels, bits per sample, frame and epoch duration), so a downgrade or a replay with other parameters fails verification. Bob verifies M_2 of Eq. (15) and decapsulates. KMAC256

1243 with 32-byte outputs then derives the keys under domain separation:

1244

1245 $\text{Transcript} = H(M_1 \| M_2 \| \sigma_A \| \sigma_B),$

1246 $K_{\text{master}} = \text{KMAC256}(\text{SS}, Q_{\text{salt}} \| \text{Transcript}, \text{"QSD-AUDIO-V1"}),$ (16)

1247

1248 $K_X = \text{KMAC256}(K_{\text{master}}, \cdot, \text{label}_X),$

1249

$X \in \{\text{SBox}, \text{ENC}, \text{AUTH}, \text{RATCHET}, \text{CONF}\}.$

1250

1251 Bob confirms the key with a KMAC256 tag over the transcript, which Alice checks
 1252 before she releases application data; the three messages take 9,049 bytes. Online
 1253 Resource 1, Section S5 gives the message layout and a sequence diagram. Confidentiality
 1254 rests on SS, so Q_{ext} and Q_{salt} need not be secret, and a cloud quantum service
 1255 that sees the QRNG output learns nothing about the session keys. The prototype uses
 1256 `kyber-py` 1.2, `dilithium-py` 1.4 and `PyCryptodome` 3.23. In the tests both parties
 1257 derived the same K_{master} and five distinct subkeys in every trial, a changed byte of
 1258 M_1 broke signature verification, a corrupted confirmation tag was rejected, and three
 1259 rekey rounds gave three different values of K_{master} .

1260

1261 8.2 Epoch S-boxes and authenticated frames

1262 The quantum layer runs once per session or rekey, and each epoch's S-box is derived
 1263 locally:

1264 $K_e = \text{KMAC256}(K_{\text{RATCHET}}, \text{SID} \| e, \text{"EPOCH"}),$ (17)

1265

1266 $K_{\text{SBox}}^{(e)} = \text{KMAC256}(K_e, \cdot, \text{"SBOX"}),$

1267 and $K_{\text{SBox}}^{(e)}$ of Eq. (17), with K_{RATCHET} from Eq. (16), is the seed of the unchanged
 1268 construction (Section 3.4), so every epoch S-box satisfies Proposition 1. Over 200
 1269 epochs from one K_{RATCHET} and SID, nonlinearity 112, differential uniformity 4, algebraic
 1270 degree 7 and the absence of fixed points held in every epoch, all S-boxes were
 1271 distinct, both endpoints derived identical S-boxes, and a different SID gave a different
 1272 S-box for epoch 0. Deriving an epoch S-box takes 29.7 ms in Python and 144 μs in C;
 1273 with 20 ms frames and 5 s epochs (250 frames) the Python cost amortizes to 119 μs
 1274 per frame.

1275 Audio is protected as independent authenticated frames, so a lost packet
 1276 does not affect later frames. Frame i of epoch e uses the 96-bit nonce
 1277 $N_i = \text{Encode}_{64}(e) \| \text{Encode}_{32}(i)$ and associated data A_i (SID, epoch, frame index,
 1278 timestamp, codec, rate). Block j of its keystream encrypts the counter block
 1279 $N_i \| \text{Encode}_{32}(j)$, so the counter blocks of different frames never coincide and no counter
 1280 block is used twice under one key:

1281

1282 $C_i = P_i \oplus \text{CTR}_{\text{QSD}}(K_{\text{ENC}}^{(e)}, S_e, N_i), \quad \text{Tag}_i = \text{KMAC256}(K_{\text{AUTH}}^{(e)}, A_i \| N_i \| C_i),$ (18)

1283

1284 where CTR_{QSD} is the counter mode keystream of QSD-BC (Section 6.1) with S-box
 1285 S_e . The receiver checks the tag before it releases plaintext, reads the frame index from
 1286 A_i , checks that A_i names its session, epoch and parameters and that N_i belongs to
 1287 that index, and keeps a sliding replay window per epoch on the index. Seven scenarios
 1288

Table 12 Properties of QSD-Audio proved with ProVerif 2.05 for an unbounded number of sessions (models and outputs in the released code, `experiments/formal`).

Property	Result
Server authenticates client: injective agreement on M_1 , M_2 and K_{master}	Proved
Client authenticates server through M_3 : injective agreement on M_1 , M_2 and K_{master}	Proved
Every accepted frame was sent in the same session and epoch, with the same index and audio	Proved
Secrecy of K_{master} at both ends, with forward secrecy	Proved
Secrecy of every epoch S-box seed $K_{\text{SBox}}^{(e)}$	Proved
Secrecy of the audio	Proved
Client's signing key compromised: client still authenticates server; keys and audio stay secret	Proved
Server's signing key compromised: server still authenticates client and frames; keys and audio stay secret	Proved
Control: server skips the check of σ_A	Attack found
Control: client accepts without M_3	Attack found
Control: K_{master} derived without SS	Attack found
Receiver of the prototype up to version 5 (index from the transport)	Attack found

behaved as specified: a round trip in order (20 of 20 frames), replay of an accepted frame (rejected), replay of an accepted frame under a new index in the transport header (rejected), ciphertext tampering (rejected), tampering with the associated data, such as a spoofed frame index (rejected), reordered delivery inside the window, [0, 1, 3, 2, 4] (accepted), and a stale frame after the window moved (rejected).

8.3 Formal verification

Model. We verified the handshake, the key schedule, the epoch keys and the frames with ProVerif 2.05 (Blanchet 2016), which proves properties for an unbounded number of sessions against a Dolev–Yao attacker (Dolev and Yao 1983). The attacker controls the network, starts sessions with any party, plays either role under keys of its own, chooses the audio parameters and the epoch and frame indices, and sees Q_{ext} . The model follows the code: ML-DSA-65 is an ideal signature over the SHA3-512 hash of the signed messages, ML-KEM-768 an ideal key encapsulation whose shared secret requires the decapsulation key, SHA3-512, SHAKE256 and KMAC256 are one-way functions, and the frame cipher is ideal symmetric encryption under $K_{\text{ENC}}^{(e)}$, $K_{\text{SBox}}^{(e)}$ and N_i . Two honest identities each run both roles any number of times, with each other, with themselves or with the attacker. After all sessions, both long-term signing keys are given to the attacker, so the secrecy results include forward secrecy.

Results. Table 12 lists the properties; ProVerif proves all of them in about 10 s. Agreement on M_1 and M_2 covers the version and the audio parameters, so neither party can be led to accept parameters the other did not send. The authentication results also hold when the signing key of the verifying party itself is compromised, which is resistance to key compromise impersonation. Three control models, each with one check removed, show that the checks matter: without the server's check of σ_A , the server accepts a client that never ran; without M_3 , the client accepts a server that never answered; and without SS in K_{master} , the attacker learns the key. ProVerif finds each of these attacks.

1335 *A flaw in the prototype.* The analysis found one flaw. The receiver of the prototype
1336 took the frame index from its caller, that is, from the transport, and not from the
1337 authenticated data. An attacker who controls the transport header can then replay an
1338 accepted frame under a new index, and the replay window accepts it. ProVerif returned
1339 this trace, and the receiver of version 5 of the released code accepted such a replay
1340 in a direct test. The receiver now reads the index from A_i and rejects a header index
1341 that disagrees with it, and the frame property of Table 12 is proved for this receiver.
1342 *Scope.* The results hold in the symbolic model, where primitives are ideal. They
1343 correspond to the usual computational assumptions, IND-CCA2 security of ML-KEM-
1344 768, EUF-CMA security of ML-DSA-65, pseudorandomness of KMAC256 and collision
1345 resistance of SHA3-512, but they are not a computational proof. ProVerif does not
1346 model the state of the replay window: it proves that every accepted frame was sent in
1347 the same session and epoch with the same index and content, and the window, which
1348 the tests check, accepts each index once.

1349

1350 8.4 Results of the whole pipeline

1351

1352 The full pipeline (quantum layer, handshake, the S-box of epoch 0, frame encryption,
1353 a simulated lossy network and authenticated decryption) ran on 5 s of synthesized
1354 48 kHz 16-bit mono PCM carrying a multitone signal (250 frames of 20 ms). It also
1355 ran on eight clips of speech synthesized offline with the `pytttsx3`/SAPI5 engine (41.3 s,
1356 2,063 frames), which have formant structure, prosody and silences. The quantum layer
1357 produced a Q_{ext} of 1,684 bytes from 4,096 shots in 0.34 s, with the extractor length
1358 set by the SP 800-90B bound of Section 4.2, and the handshake took 0.31 s. With 5%
1359 loss and local reordering (window 8), 233 of 250 frames were delivered and all 233 were
1360 recovered byte for byte. Over loss rates of 0 to 20% (150 independently seeded trials)
1361 every delivered frame was recovered in every trial, and all 2,063 speech frames were
1362 recovered byte for byte. Encryption ran at a real-time factor of about 2 in Python and
1363 6.1×10^{-4} in C.

1364 Changing one bit of the plaintext changed exactly $1/1920 \approx 0.05\%$ of the ciphertext
1365 bytes. This is correct for the stream cipher of Eq. (18), $C_i = P_i \oplus \text{keystream}$, whose
1366 confidentiality rests on a keystream that is never reused under the same key and
1367 nonce. Diffusion measures such as NPCR and UACI, designed for image ciphers based
1368 on permutation, do not apply to it; the checks that do apply are the statistics of the
1369 ciphertext over a whole signal.

1370

1371 8.5 Histogram analysis

1372 Fig. 4 compares the distribution of 16-bit sample values before and after encryption,
1373 for the multitone signal (250 frames) and for the speech clips (2,063 frames), each
1374 encrypted as in deployment with a new epoch S-box every 250 frames. Before encryp-
1375 tion both histograms are far from flat: the tone fills a band around zero, and the speech
1376 has a sharp peak at zero from its silences. After encryption both are flat. Against the
1377 uniform distribution, the chi-square statistics over 256 bins (255 degrees of freedom)
1378 are 269.2 ($p = 0.26$) and 250.6 ($p = 0.57$) for the 16-bit sample values, and 301.2
1379

1380

($p = 0.025$) and 287.2 ($p = 0.081$) for the bytes, so none of the four tests rejects uniformity at the 1% level; the plaintext statistics are 1.1×10^5 to 5.9×10^7 . The byte entropy of the ciphertext is 7.9995 and 7.9999 bits per byte, against 7.83 and 6.32 for the plaintext. The correlation between adjacent samples falls from 0.96 and 0.99 to 0.0001 and -0.0020 , and the correlation between plaintext and ciphertext samples is 0.0025 and -0.0003 . All 247,560 counter blocks used for the speech clips are distinct, and the keystream of a silent frame agrees with that of the next frame in 0.47% of its bytes, the rate expected by chance ($1/256$).

These tests are sensitive to keystream reuse, which the entropy is not. If the counter blocks are formed as the 128-bit sum $N_i + j$ of a 128-bit frame nonce and the block index, as in an earlier version of the prototype, block j of frame i equals block $j - 1$ of frame $i + 1$, and consecutive frames share all but one keystream block. The byte entropy of the encrypted speech then stays at 7.9994 bits per byte, but the chi-square statistics rise to 3,151 for the bytes and 4,066 for the sample values ($p < 10^{-100}$), because the silences expose the shared keystream. This is how we found the error, which the counter layout of Section 8.2 removes.

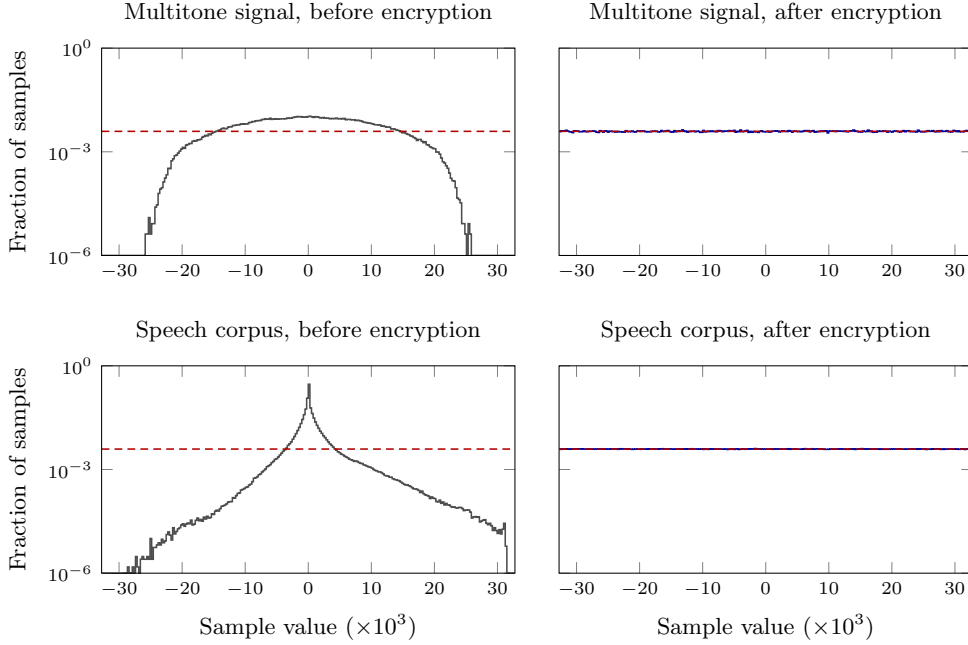


Fig. 4 Histograms of 16-bit sample values (256 bins, logarithmic scale) before and after QSD-Audio encryption, for the multitone signal and the speech clips. The dashed line is the uniform level, $1/256$ per bin.

8.6 Deployment baselines

Three baselines (Table 13) locate the cost of QSD-Audio on the same 250 frames of 1,920 B (5 s of audio). Baseline 1 uses QSD-Audio’s frame format, KMAC authentication and nonce schedule with the standard AES S-box, which isolates the cost of S-box rotation. Baselines 2 and 3 are OpenSSL’s AES-128-GCM and ChaCha20-Poly1305 with hardware acceleration (the latter for processors without AES instructions).

Table 13 Deployment baselines (250 frames of 1,920 B; mean of 3 repeats). RTF is encryption time divided by audio duration.

Scheme	Encryption RTF	Overhead (B/frame)	Correct
QSD-Audio, Python (this work)	2.14	101	Yes
Baseline 1: same protocol, standard S-box, Python	2.04	101	Yes
QSD-Audio frame operation, C (this work)	6.1×10^{-4}	101	Yes
Baseline 2: AES-128-GCM, OpenSSL	1.5×10^{-4}	44	Yes
Baseline 3: ChaCha20-Poly1305, OpenSSL	1.2×10^{-4}	44	Yes

Against Baseline 1, which differs only in the S-box, QSD-Audio is 1.05 times slower in Python, so rotating the S-box costs about 5% there, and in C the session and standard S-boxes give the same throughput. Against the production baselines the Python prototype is 15,000 to 19,500 times slower, a gap caused by the implementation, not by QSD-SBox. The C frame operation is 4 to 5 times slower than these OpenSSL calls (and 17 times slower than AES-128-GCM called from C), because it uses table lookups instead of the AES instructions and computes a KMAC256 tag, but it still needs only 0.06% of the audio duration. The overhead of 101 B per frame, against 44 B, comes from QSD-Audio’s larger explicit associated data (version, SID, epoch, frame index, timestamp and audio parameters), not from the cipher.

9 Limitations

Scope of the security evaluation. The security evaluation follows the objective of QSD-SBox: limiting the cross-session transfer of table-specific information while every session keeps its per-instance cryptographic guarantees. We therefore evaluate the invariant properties, active S-box bounds, exact differential behavior over two rounds, boomerang uniformity, and the cross-session transfer of differential, linear and trail profiles and of a simulated side-channel profile. These experiments are not a complete cryptanalysis of QSD-BC. Three complementary evaluations are left for future work: a search for weighted trails over all rounds, where checking the tightness of the bound would need a MILP that encodes the probabilities of the difference table; cryptanalytic attacks that recover the key over all rounds with differential, linear, integral or related-key techniques; and side-channel experiments on physical devices. The key recovery in Section 6.3 is a side-channel attack on simulated leakage; its result concerns cross-session transfer under that leakage model and does not imply resistance to side-channel attacks in general. The same comparison with a static S-box on a physical

device, with measured traces, would test it under real leakage. Because all QSD-SBox instances are affine equivalent to inversion, the construction is not intended to defeat attacks based on properties invariant under affine equivalence, or side-channel attacks on the S-box input, which does not involve the table; its objective is to diversify the information tied to a concrete table. SHAKE-256 and SHA-256 are modeled as random oracles, and of the rows of Table 8 only AES was recomputed.

Validation of the entropy source. The algebraic guarantees of QSD-SBox depend on the construction, not on quantum randomness, so a properly seeded CSPRNG provides the same per-instance cryptographic guarantees; the quantum source gives the seeds a physical origin. Most large statistical experiments use simulated noise models and therefore characterize the distortion of the distribution and the effect of whitening, not physical quantum unpredictability. The hardware validation is 10^6 shots from each of two IBM devices, collected on one day; it does not capture drift over time, restart behavior or changes after recalibration. Both devices were about twice as biased as the stress model, and one showed crosstalk between two qubits, which whitening and SHAKE-256 absorb; a user who needs uniform bits without a hash must check each device. The non-IID estimates of SP 800-90B are conservative lower bounds (Zhu et al. 2017), not exact entropy rates. The QRNG assumes a trusted device; it is not a source-independent or device-independent generator.

Implementation and application scope. The Python reference implementation serves reproducibility, not performance. The C implementation shows that S-box generation and rotation are cheap in practice, but it remains slower than AES with the AES instructions, and its table lookups are not constant time, so, like any AES built on table lookups, it leaks through cache timing; bitsliced, hardware or embedded implementations remain future work. QSD-Audio is a deployment case study. Its handshake and frames are verified in the symbolic model with ProVerif, which assumes ideal primitives, gives no computational bound and leaves the uniqueness of accepted frame indices to the replay window. Its evaluation covers local protocol logic, authenticated session establishment, S-box synchronization, replay handling, frame recovery and processing cost, but not network transport, resistance to denial of service, a large corpus of natural speech, a production codec such as Opus, or frames in both directions, which would need separate keys for each direction. Standalone QSD-SBox assumes an authenticated and confidential channel for the seed and, with a cloud device, trust in its operator unless a local secret is mixed in (Section 3.6).

10 Conclusion

This work presented QSD-SBox, a session diversification scheme for 8×8 S-boxes that changes the concrete substitution table across sessions without giving up established cryptographic properties. Each session instance is generated as $\mathcal{S} = A_{\text{out}} \circ I \circ A_{\text{in}}$ and is affine equivalent to multiplicative inversion in $GF(2^8)$, so nonlinearity 112, differential uniformity 4, algebraic degree 7 and boomerang uniformity 6 hold for every instance by construction, not by testing after generation or by heuristic search. The class it draws from holds about 2^{129} tables.

1519 The construction separates the strength of a single session from session diversifica-
1520 tion. QSD-SBox does not make a single session stronger than a cipher built on a well
1521 analyzed S-box derived from inversion, and affine diversification does not stop attacks
1522 based on affine invariants. It changes the concrete table between sessions, so that anal-
1523 ysis or profiling tied to one table is not automatically a reusable asset in the next
1524 session. All 1,000 simulated sessions and 488 sessions seeded by hardware were distinct
1525 and kept the invariants. Over 999,000 ordered pairs of sessions, differential character-
1526 istics, linear approximations and trails over two rounds profiled on one session held
1527 in another only at chance level, and none of 2.5×10^8 transferred trails reached the
1528 bound. In the simulated template attack, a profile built in one session recovered the
1529 next session's full key with 50 attack traces when both used a static AES S-box, and no
1530 more key bytes than guessing when they used QSD-SBox tables, under the evaluated
1531 leakage model. The AES active S-box bound remained valid in every session.

1532 The OTQuSbox quantum random number generator provides one physical source
1533 of session seeds, and the CSPRNG baseline shows that the per-instance cryptographic
1534 guarantees do not depend on the source. SP 800-90B, Dieharder, experiments under
1535 noise models and measurements on two IBM devices characterize the source and the
1536 effect of whitening and conditioning; they support its use for seeds and also show the
1537 limits of the present hardware evidence.

1538 Practical experiments show that session diversification costs little at run time.
1539 The C implementation generates a new S-box in 136 μs , and the QSD-Audio case
1540 study rotates S-boxes per epoch in step at both endpoints of an authenticated post-
1541 quantum session, with processing far faster than the audio plays. ProVerif proves
1542 the authentication and secrecy properties of its handshake and frames, and the same
1543 analysis found a replay flaw in the prototype, now fixed.

1544 QSD-SBox is therefore not a new nonlinear primitive but a mechanism for session
1545 diversification with per-instance cryptographic guarantees. It preserves established
1546 cryptographic bounds for every generated S-box while, under the attack and leakage
1547 models evaluated here, reducing the cross-session transfer of table-specific attack infor-
1548 mation to the level of chance. Future work should extend the analysis to cryptanalysis
1549 over all rounds, cross-session side-channel profiling on physical devices, repeated hard-
1550 ware measurements over time and constant-time implementations. Designs that derive
1551 the S-box from the key should keep the quantum circuit independent of secret mate-
1552 rial, since gates parameterized by key bits would reveal those bits to the operator of
1553 a cloud quantum service; quantum output should be combined with the session secret
1554 classically, in a key derivation function, as QSD-Audio does.

1555 **Supplementary information.** Online Resource 1 (PDF) holds the base table and
1556 the BIC and difference tables of the illustrative session, the noise model parameters,
1557 the individual SP 800-90B estimates, the complete Dieharder results, the per-bit and
1558 qubit-pair results of both hardware runs, the changes made to the test tools, the QSD-
1559 BC listings, the QSD-Audio handshake with its ProVerif verification, and the success
1560 rates of the simulated side-channel attacks. Online Resource 2 (ZIP) holds the code,
1561 the raw shots of both hardware runs and all test outputs, anonymized for review.

1563
1564

References

- Adams C, Tavares S (1990) The structured design of cryptographically good S-boxes. *Journal of Cryptology* 3(1):27–41. <https://doi.org/10.1007/BF00203967>
- Ali A, Khan M, Ayyasamy R, et al (2022) A novel systematic byte substitution method to design strong bijective substitution box (S-box) using piece-wise-linear chaotic map. *PeerJ Computer Science* <https://doi.org/10.7717/peerj-cs.940>
- Aslam CMA, Ullah I, Ishaq M (2025) An optimal and efficient framework for the construction of nonlinear components based on a polynomial ring. *Mathematics and Computers in Simulation* 237:263–280. <https://doi.org/10.1016/j.matcom.2025.04.022>
- Barker E, Kelsey J (2015) Recommendation for random number generation using deterministic random bit generators. NIST Special Publication 800-90A Rev. 1, National Institute of Standards and Technology, <https://doi.org/10.6028/NIST.SP.800-90Ar1>
- Bassham LE, Rukhin AL, Soto J, et al (2010) A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22 Rev. 1a, National Institute of Standards and Technology, <https://doi.org/10.6028/NIST.SP.800-22r1a>
- Billet O, Gilbert H, Ech-Chatbi C (2004) Cryptanalysis of a white box AES implementation. In: *Selected Areas in Cryptography (SAC 2004)*, Lecture Notes in Computer Science, vol 3357. Springer, pp 227–240, https://doi.org/10.1007/978-3-540-30564-4_16
- Biryukov A, De Cannière C, Braeken A, et al (2003) A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In: *Advances in Cryptology – EUROCRYPT 2003*, Lecture Notes in Computer Science, vol 2656. Springer, pp 33–50, https://doi.org/10.1007/3-540-39200-9_3
- Blanchet B (2016) Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Foundations and Trends in Privacy and Security* 1(1–2):1–135. <https://doi.org/10.1561/33000000004>
- Bomhals A, Bruynsteen C, Meysmans C, et al (2026) 100 Gbit/s secure and real-time vacuum fluctuation based quantum random number generator. *Journal of Lightwave Technology* 44(16):7182–7190. <https://doi.org/10.1109/jlt.2026.3665249>
- Boura C, Canteaut A (2018) On the boomerang uniformity of cryptographic Sboxes. *IACR Transactions on Symmetric Cryptology* 2018(3):290–310. <https://doi.org/10.13154/tosc.v2018.i3.290-310>

1611 Brier E, Clavier C, Olivier F (2004) Correlation power analysis with a leakage
1612 model. In: Cryptographic Hardware and Embedded Systems (CHES 2004), Lecture
1613 Notes in Computer Science, vol 3156. Springer, pp 16–29, [https://doi.org/10.1007/](https://doi.org/10.1007/978-3-540-28632-5_2)
1614 [978-3-540-28632-5_2](https://doi.org/10.1007/978-3-540-28632-5_2)

1615

1616 Brown RG, Eddelbuettel D, Bauer D (2013) Dieharder: A random num-
1617 ber test suite, version 3.31.1. Duke University Physics Department, URL
1618 <https://webhome.phy.duke.edu/~rgb/General/dieharder.php>, source used here:
1619 <https://github.com/eddelbuettel/dieharder>

1620

1621 Chari S, Rao JR, Rohatgi P (2003) Template attacks. In: Cryptographic Hardware
1622 and Embedded Systems (CHES 2002), Lecture Notes in Computer Science, vol 2523.
1623 Springer, pp 13–28, https://doi.org/10.1007/3-540-36400-5_3

1624

1625 Cid C, Huang T, Peyrin T, et al (2018) Boomerang connectivity table: A new crypt-
1626 analysis tool. In: Advances in Cryptology – EUROCRYPT 2018, Lecture Notes
1627 in Computer Science, vol 10821. Springer, pp 683–714, [https://doi.org/10.1007/](https://doi.org/10.1007/978-3-319-78375-8_22)
1628 [978-3-319-78375-8_22](https://doi.org/10.1007/978-3-319-78375-8_22)

1629

1630 Daemen J, Rijmen V (2020) The Design of Rijndael: The Advanced Encryption Stan-
1631 dard (AES), 2nd edn. Information Security and Cryptography, Springer, <https://doi.org/10.1007/978-3-662-60769-5>

1632

1633 Dimitrov MM (2020) On the design of chaos-based S-boxes. IEEE Access 8:117173–
1634 117181. <https://doi.org/10.1109/ACCESS.2020.3004526>

1635

1636 Dinur I (2018) An improved affine equivalence algorithm for random permutations. In:
1637 Advances in Cryptology – EUROCRYPT 2018, Lecture Notes in Computer Science,
1638 vol 10820. Springer, pp 413–442, https://doi.org/10.1007/978-3-319-78381-9_16

1639

1640 Dolev D, Yao A (1983) On the security of public key protocols. IEEE Transactions on
1641 Information Theory 29(2):198–208. <https://doi.org/10.1109/TIT.1983.1056650>

1642

1643 Du Y, Hua X, Zhao Z, et al (2025) Source-independent quantum random number
1644 generators with integrated silicon photonics. Communications Physics 8:9. <https://doi.org/10.1038/s42005-024-01917-x>

1645

1646 Dube A, Yadav R (2024) Enhanced dynamic S-box design based on adaptive heuris-
1647 tic evolution and multi-stage nonlinearity for securing IoT-based remote health
1648 monitoring systems. In: International Conference on Intelligent Computing and
1649 Emerging Communication Technologies (ICEC). IEEE, pp 1–8, [https://doi.org/10.](https://doi.org/10.1109/ICEC59683.2024.10837025)
1650 [1109/ICEC59683.2024.10837025](https://doi.org/10.1109/ICEC59683.2024.10837025)

1651

1652 Ejaz A, Murtaza G, Haider T, et al (2025) An efficient S-box generation with mutation
1653 optimization for secure image encryption. Security and Privacy 8(4):e70068. <https://doi.org/10.1002/spy2.70068>

1654

1655

1656

Farkas M, Volčić J, Storgaard SAL, et al (2026) Maximal device-independent randomness in every dimension. <i>Nature Physics</i> 22(2):319–324. https://doi.org/10.1038/s41567-025-03141-y	1657 1658 1659 1660
Garipcan AM, Aydin Y, Ozkaynak F (2025) A novel S-box generation method based on metastable inducing over FPGA for block ciphers. <i>Knowledge-Based Systems</i> 310:112968. https://doi.org/10.1016/j.knosys.2025.112968	1661 1662 1663 1664
Henry E (2024) The role of quantum random number generation in enhancing encryption security. Available at SSRN 4966139, https://doi.org/10.2139/ssrn.4966139	1665 1666 1667
Herrero-Collantes M, Garcia-Escartin JC (2017) Quantum random number generators. <i>Reviews of Modern Physics</i> 89(1):015004. https://doi.org/10.1103/RevModPhys.89.015004	1668 1669 1670
Impagliazzo R, Levin LA, Luby M (1989) Pseudo-random generation from one-way functions. In: <i>Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '89)</i> . ACM, pp 12–24, https://doi.org/10.1145/73007.73009	1671 1672 1673 1674
Javadi-Abhari A, Treinish M, Krsulich K, et al (2024) Quantum computing with Qiskit. arXiv preprint arXiv:2405.08810, https://doi.org/10.48550/arXiv.2405.08810	1675 1676 1677 1678
Jayalathsumi U, Godhavari T, Keerthana B, et al (2025) Enhancing quantum random number generators, cryptographic security, and machine learning-based cryptanalysis. In: <i>Humans and Generative AI Tools for Collaborative Intelligence</i> . IGI Global Scientific Publishing, p 327–342, https://doi.org/10.4018/979-8-3693-8332-2.ch014	1679 1680 1681 1682 1683
Krawczyk H (1994) LFSR-based hashing and authentication. In: <i>Advances in Cryptology – CRYPTO '94, Lecture Notes in Computer Science</i> , vol 839. Springer, pp 129–139, https://doi.org/10.1007/3-540-48658-5_15	1684 1685 1686 1687
Kulikov A, Storz S, Schär JD, et al (2026) Experimental randomness amplification. <i>Nature</i> 653(8116):1033–1038. https://doi.org/10.1038/s41586-026-10521-8	1688 1689 1690
Kumar V, Pravinkumar P (2024) Quantum random number generator on IBM QX. <i>Journal of Cryptographic Engineering</i> 14:253–259. https://doi.org/10.1007/s13389-023-00341-1	1691 1692 1693
L'Ecuyer P, Simard R (2007) TestU01: A C library for empirical testing of random number generators. <i>ACM Transactions on Mathematical Software</i> 33(4):22:1–22:40. https://doi.org/10.1145/1268776.1268777	1694 1695 1696 1697
Li Y, Fei Y, Wang W, et al (2021) Quantum random number generator using a cloud superconducting quantum computer based on source-independent protocol. <i>Scientific Reports</i> 11:23873. https://doi.org/10.1038/s41598-021-03286-9	1698 1699 1700 1701 1702

1703 Lu Q, Zhu C, Wang G (2019) A novel S-box design algorithm based on a new
1704 compound chaotic system. Entropy 21(10):1004. <https://doi.org/10.3390/e21101004>
1705
1706 Ma X, Yuan X, Cao Z, et al (2016) Quantum random number generation. npj Quantum
1707 Information 2:16021. <https://doi.org/10.1038/npjqi.2016.21>
1708
1709 Malik A, Zahid A, Bhatti D, et al (2023) Designing S-box using Tent-Sine chaotic
1710 system while combining the traits of Tent and Sine map. IEEE Access 11:79265–
1711 79274. <https://doi.org/10.1109/ACCESS.2023.3298111>
1712
1713 Malik AW, Mustafa G, Suh B, et al (2026) OTQuSbox framework: A quantum-assisted
1714 session-based S-box construction with secure seed synchronization. IEEE Access
1715 14:73555–73567. <https://doi.org/10.1109/ACCESS.2026.3692919>
1716
1717 Matsui M (1994) Linear cryptanalysis method for DES cipher. In: Advances in Cryptology – EUROCRYPT ’93, Lecture Notes in Computer Science, vol 765. Springer,
1718 pp 386–397, https://doi.org/10.1007/3-540-48285-7_33
1719
1720 Mouha N, Wang Q, Gu D, et al (2012) Differential and linear cryptanalysis
1721 using mixed-integer linear programming. In: Information Security and Cryptology
1722 (Inscrypt 2011), Lecture Notes in Computer Science, vol 7537. Springer, pp 57–76,
1723 https://doi.org/10.1007/978-3-642-34704-7_5
1724
1725 Murtaza G, Hayat U (2025) Efficient image encryption algorithm based on ECC and
1726 dynamic S-box. Journal of Information Security and Applications 90:104004. <https://doi.org/10.1016/j.jisa.2025.104004>
1727
1728
1729 Naseer M, Tariq S, Riaz N, et al (2025) A quantitative security analysis of S-boxes
1730 in the NIST lightweight cryptography finalists. Discover Computing 28:209. <https://doi.org/10.1007/s10791-025-09721-z>
1731
1732 National Institute of Standards and Technology (2001) Advanced encryption standard
1733 (AES). FIPS 197, National Institute of Standards and Technology, <https://doi.org/10.6028/NIST.FIPS.197>
1734
1735
1736 National Institute of Standards and Technology (2015) SHA-3 standard: Permutation-
1737 based hash and extendable-output functions. FIPS 202, National Institute of
1738 Standards and Technology, <https://doi.org/10.6028/NIST.FIPS.202>
1739
1740 National Institute of Standards and Technology (2024a) Module-lattice-based digital
1741 signature standard. FIPS 204, National Institute of Standards and Technology,
1742 <https://doi.org/10.6028/NIST.FIPS.204>
1743
1744 National Institute of Standards and Technology (2024b) Module-lattice-based key-
1745 encapsulation mechanism standard. FIPS 203, National Institute of Standards and
1746 Technology, <https://doi.org/10.6028/NIST.FIPS.203>
1747
1748

National Institute of Standards and Technology (2024c) SP800-90B_EntropyAssessment: C++ implementation of the SP 800-90B min-entropy estimators. GitHub repository usnistgov/SP800-90B_EntropyAssessment, URL https://github.com/usnistgov/SP800-90B_EntropyAssessment	1749 1750 1751 1752 1753
National Institute of Standards and Technology (2025) Recommendation for random bit generator (RBG) constructions. NIST Special Publication 800-90C, National Institute of Standards and Technology, final, September 2025	1754 1755 1756 1757
Nyberg K (1994) Differentially uniform mappings for cryptography. In: Advances in Cryptology – EUROCRYPT '93, Lecture Notes in Computer Science, vol 765. Springer, pp 55–64, https://doi.org/10.1007/3-540-48285-7_6	1758 1759 1760 1761
Patel S, Thanikaiselvan V, Rearajan A (2024) Secured quantum image communication using new two dimensional chaotic map based encryption methods. International Journal of Theoretical Physics 63(2):49. https://doi.org/10.1007/s10773-024-05548-4	1762 1763 1764 1765
Rasheed MW, Amjad M (2026) A biometric-algebraic block cipher framework using fingerprint minutiae, ridge orientation fields, and $GL(8, GF(2))$ transformations for dynamic S-box generation. Security and Privacy 9(5):e70247. https://doi.org/10.1002/spy2.70247	1766 1767 1768 1769 1770
Razaq A, Ramzan M, Xin Q, et al (2026) From Schreier graphs to secure S-boxes using a group-theoretic design framework. Scientific Reports 16:26411. https://doi.org/10.1038/s41598-026-56441-5	1771 1772 1773 1774
Saini A, Tsokanos A, Kirner R (2024) QuantumGS-box—A key-dependent GA and QRNG-based S-box for high-speed cloud-based storage encryption. Sci 6(4):86. https://doi.org/10.3390/sci6040086	1775 1776 1777 1778
Salehi R, Malik AW, Novotny MA, et al (2026) Parameterized quantum random number generators for superconducting quantum hardware: novel architectures and NIST SP 800-22 evaluation under IBM noisy simulators. Scientific Reports 16:22780. https://doi.org/10.1038/s41598-026-51033-9	1779 1780 1781 1782 1783
Selçuk AA (2008) On probability of success in linear and differential cryptanalysis. Journal of Cryptology 21(1):131–147. https://doi.org/10.1007/s00145-007-9013-7	1784 1785 1786
Singh LD, Lahoty A, Devi C, et al (2024) Image encryption using dynamic S-boxes generated using elliptic curve points and chaotic system. Journal of Information Security and Applications 83:103793. https://doi.org/10.1016/j.jisa.2024.103793	1787 1788 1789 1790
Tiessen T, Knudsen LR, Kölbl S, et al (2015) Security of the AES with a secret S-box. In: Fast Software Encryption (FSE 2015), Lecture Notes in Computer Science, vol 9054. Springer, pp 175–189, https://doi.org/10.1007/978-3-662-48116-5_9	1791 1792 1793 1794

- 1795 Turan MS, Barker E, Kelsey J, et al (2018) Recommendation for the entropy
1796 sources used for random bit generation. NIST Special Publication 800-90B, National
1797 Institute of Standards and Technology, <https://doi.org/10.6028/NIST.SP.800-90B>
1798
- 1799 Ullah I, Arif S, Abbas F, et al (2026) Efficient and secure Montgomery curve based sub-
1800 stitution box generator with optimal nonlinearity. Security and Privacy 9(1):e70168.
1801 <https://doi.org/10.1002/spy2.70168>
1802
- 1803 Waheed A, Subhan F, Suud M, et al (2023) An analytical review of current S-
1804 box design methodologies, performance evaluation criteria, and major challenges.
1805 Multimedia Tools and Applications 82(19):29689–29712. <https://doi.org/10.1007/s11042-023-14910-3>
1806
- 1807 Wang S, Jang K, Baksi A, et al (2024) New results in quantum analysis of LED:
1808 Featuring one and two oracle attacks. Cryptology ePrint Archive, Paper 2024/1982,
1809 <https://eprint.iacr.org/2024/1982>
1810
- 1811 Webster AF, Tavares SE (1986) On the design of S-boxes. In: Advances in Cryptology
1812 – CRYPTO ’85, Lecture Notes in Computer Science, vol 218. Springer, pp 523–534,
1813 https://doi.org/10.1007/3-540-39799-X_41
1814
- 1815 Yang J, Wu M, Zhang Y, et al (2026) An ultra-fast quantum random number gen-
1816 eration scheme based on laser phase noise. Communications Physics 9:5. <https://doi.org/10.1038/s42005-025-02436-z>
1817
- 1818
- 1819 Yogaraja, Rani SS (2023) Key-based dynamic S-box approach for PRESENT
1820 lightweight block cipher. KSII Transactions on Internet and Information Systems
1821 17(12):3398–3415. <https://doi.org/10.3837/tiis.2023.12.010>
1822
- 1823 Zahid AH, Elahi HAM, Ahmad M, et al (2023) Secure key-based substitution-boxes
1824 design using systematic search for high nonlinearity. IEEE Access 11:139745–139758.
1825 <https://doi.org/10.1109/ACCESS.2023.3339389>
1826
- 1827 Zhang L, Li G, Dai J, et al (2025) Efficient quantum circuit research and imple-
1828 mentation for 16-bit S-boxes based on composite field. Journal of King Saud
1829 University Computer and Information Sciences 37(4):55. <https://doi.org/10.1007/s44443-025-00065-8>
1830
- 1831 Zhu S, Ma Y, Chen T, et al (2017) Analysis and improvement of entropy estimators in
1832 NIST SP 800-90B for non-IID entropy sources. IACR Transactions on Symmetric
1833 Cryptology 2017(3):151–168. <https://doi.org/10.13154/tosc.v2017.i3.151-168>
1834

1836 Statements and Declarations

1838 **Funding.** Funding information is given on the separate title page to preserve
1839 anonymity during review.
1840

Competing interests. The authors have no relevant financial or non-financial interests to disclose.	1841
	1842
Author contributions. Given on the separate title page.	1843
	1844
Ethics approval and consent to participate. Not applicable.	1845
	1846
Consent for publication. Not applicable.	1847
Data availability. The code, the raw shots of both IBM hardware runs and all test outputs are provided for review as Online Resource 2, anonymized. They are also public in a repository with a persistent identifier, together with a public notebook that reruns the main checks; both links identify the authors, so they are withheld for double-anonymous review and listed on the title page.	1848
	1849
	1850
	1851
	1852
	1853
Code availability. See Data availability. The code is released under the MIT License, and the data and results under CC BY 4.0.	1854
	1855
Use of generative AI. During the preparation of this manuscript, the authors did not use generative artificial intelligence or AI-assisted technologies in the writing, editing, or preparation of the manuscript.	1856
	1857
	1858
	1859
	1860
	1861
	1862
	1863
	1864
	1865
	1866
	1867
	1868
	1869
	1870
	1871
	1872
	1873
	1874
	1875
	1876
	1877
	1878
	1879
	1880
	1881
	1882
	1883
	1884
	1885
	1886