A Defense of the RCS Cipher Construction

The case for wide-block Rijndael, implemented securely in the RCS cipher construction

Revision 1a, April 11, 2025

John G. Underhill – contact@qrcscorp.ca

Index

1 A Critical Analysis of the Native Rijndael-256 Construction

Architectural overview

Rijndael-256 doubles the AES state to eight 32-bit columns while retaining the original round structure and a linear key schedule. For a 256-bit key the full cipher executes 14 rounds.

Ferguson et al. characterized the schedule as “unsettling” because it contains only one nonlinear step per 32-bit word every eight bytes of expansion [1]. Shamir’s group later quantified that a single-byte difference “affects at most one additional byte per schedule round” [2].

Published attacks exploit these properties:

Year	Technique	Rounds broken	Workfactor	Keys / Ref
2000	Related-key rectangle	9	2^39 enc.	2 [1]
2008	Truncated differential	8	2^116 enc.	1 [3]

Because 10–11 rounds are penetrated, fourteen rounds leave only three to four rounds of undisputed margin – below the “twice the best break” heuristic noted by NIST [4] and Schneier [5].

Consequently, any wide-block deployment must replace the key schedule, raise the round count, and pair the cipher with a quantum-robust MAC, needs addressed by RCS.

Effective security margin

Because 10–11 rounds are cryptanalytically penetrated, 14 rounds leave only 3–4 safe rounds. NIST noted during the AES selection that "an algorithm should have twice the number of rounds that the best reduced-round analysis breaks" [4]. Schneier reiterates: "If you can break *n* rounds you design for 2 * n or 3 * n" [5]. Rijndael-256 therefore under-provisions its margin by roughly eight rounds.

Implications for wide-block deployment

- **Amplified leakage:** Each weak linear relation would appear in two state columns (doubling its presence).
- **Related-key scaling:** Boomerang and rectangle attacks could become cheaper per key byte.
- **Post-quantum concern:** Grover's √-speedup halves brute-force cost, so a cipher already conceding 11 analytical rounds risks < 100-bit strength for 30-year confidentiality.

Interim conclusion

The linear schedule is the structural weak point, and the wide-block variant magnifies it. A secure profile must (i) replace the schedule, (ii) raise the round count, and (iii) adopt a quantum-robust MAC, the motivations for RCS.

2 The Efficacy of Keccak-Family Sponge Functions in an Improved Rijndael Construction

2.1 Keccak Security in Classical and Quantum Settings

Since its 2008 introduction, Keccak-f[1600] has withstood more than fifteen years of open cryptanalysis. All published distinguishers and pre-image or collision attacks stop at ≤ 6 of the 24 rounds [6], or rely on padding shortcuts invalid for SHA-3/cSHAKE/KMAC.

No attack threatens the full permutation or the sponge construction at claimed capacities. In the quantum realm, generic amplitude-amplification reduces brute-force search by $\sqrt{N}$, but no structural quantum shortcut is known against Keccak; the best results remain the Grover bound and generic Simon-style period tests that fail because Keccak's round function is highly non-linear and full-state. Consequently, a 512-bit capacity sponge retains ≥ 256-bit post-quantum strength—consistent with NIST's PQC migration guidance. [7–10]

2.2 cSHAKE as a Tweakable, High-Diffusion Key Schedule

The original Rijndael schedule is linear and sparsely non-linear. Replacing it with cSHAKE-256/512 yields three concrete advantages:

- **Full-entropy round keys.** Each 32-bit word is a direct XOF output after the entire master key and a domain-separation string have been absorbed, eliminating low-weight correlations.
- **Intrinsic tweakability.** The *CustomizationString* and *FunctionName* fields allow per-device or per-context diversification without altering the cipher core, a requirement in many NIST CNSA deployments.
- **Provable quantum soundness.** A keyed sponge with capacity ≥ 2 λ is collapsing; thus cSHAKE-driven keys are indistinguishable from random to a Q2 adversary. That property blocks the related-key and slide attacks that exploit schedule linearity [8].

2.3 KMAC as a Quantum-Safe Authenticator

KMAC is a keyed sponge MAC with tag lengths 128 or 256 bits and the same Keccak-f[1600] core. Unruh's collapsing proof shows that, for capacity ≥ 2 λ, no adversary -classical or quantum, adaptive or superposition -can forge a tag below 2^λ work unless the permutation itself is broken. KMAC-256 therefore guarantees 256-bit resistance in the Q1 model and 256-bit resistance in the strong Q2 model, comfortably exceeding the ≥ 128-bit post-quantum target. [8]

2.4 Comparison with GMAC and Poly1305 in Quantum Models

Property	GMAC (GHASH)	Poly1305	KMAC-256 (λ = 256)
Classical tag strength	128 bits	128 bits	256 bits
Q1 strength (Grover)	≈ 64 bits	≈ 64 bits	≈ 128 bits
Q2 authenticity	**Broken** by Simon (O(128) queries)	**Broken** (O(128) queries)	**Secure** (collapsing proof)
Hardware status	AES-NI + CLMUL ubiquitous	Same CLMUL	Planned adoption 2026+
Integrated in AEAD	Needs AES counter-mode	Needs separate cipher	Single-pass AEAD with RCS

In Q1 (classical queries), doubling tag size offsets Grover, yet GMAC/Poly1305 still top out at 64-bit brute-force margins. In Q2 (quantum-oracle) both polynomial MACs fail catastrophically, whereas KMAC retains full security, the decisive rationale for its adoption in RCS.

2.5 Design Implications for RCS

- **Unified primitive:** one compact Keccak-f[1600] implementation services both key schedule (cSHAKE) and MAC (KMAC), minimizing code size and easing constant-time hardening.
- **Upcoming acceleration:** Intel, ARM and RISC-V roadmaps all list SHA-3/Keccak micro-ops; RCS will gain the same single-cycle boost as future SHA-3 deployments.
- **Robustness beyond Grover:** the collapsing proof eliminates the Simon-style forgery class, something polynomial-hash AEADs cannot match without per-message keying.

Taken together, these points show that Keccak-based cSHAKE and KMAC provide the strongest known blend of classical maturity, quantum resilience and implementation practicality, making them the natural foundation for a secure, wide-block Rijndael variant.

3 The RCS Cipher Construction: A Future-Proof Specification

This section formalizes RCS exactly as it is described in the public specification and implemented in the reference C code.

3.1 Symbols and conventions

Symbol	Meaning
b	Block size = 256 bits = 32 bytes
k	Input cipher key: 256 bits (RCS-256) or 512 bits (RCS-512)
i	Optional *info* tweak string used for domain separation
r	Number of rounds: 22 (RCS-256) or 30 (RCS-512)
rk	Round-key stream generated by cSHAKE
mk	MAC key generated by cSHAKE
n	256-bit nonce / initial counter
ad	Associated data (any length, may be empty)
m	Plaintext message
c	Ciphertext (same length as *m*)
l	64-bit little-endian encoding of ad ‖ n ‖ c length
tag	Authentication code (256 bits for RCS-256, 256 or 512 bits for RCS-512)
E	RCS block-cipher encryption function
CTR	Counter-mode keystream function
S	cSHAKE-256/512 extendable-output function
M	KMAC-256/512 (or QMAC-256) keyed sponge

The cipher state is a 4 × 8 byte matrix **A[4][8]** over GF(2^8) (row 0–3, column 0–7) .

3.2 Round transformation

For rounds $t = 1 \ldots r - 1$

1. **SubBytes** `A ← Sbox(A)`
2. **ShiftRows** Row *i* rotated left by *i* bytes (mod 8)
3. **MixColumns** Each column multiplied by the AES polynomial matrix
4. **AddRoundKey** `A ← A ⊕ rk[t]`

Round *r* omits MixColumns (as in AES). Transformation order and constants are identical to Rijndael; the only change is the doubled column width.

3.3 Key-expansion (tweakable cSHAKE schedule)

```
Input : k , i
S ← cSHAKE-256  (RCS-256)  or  cSHAKE-512  (RCS-512)
S.absorb( k ‖ i )
rk  ‖  mk  ←  S.squeeze(  (r+1)·b  +  |mk|  )   // first part is round keys
```

Domain separation – the function-name string “RCS-256” or “RCS-512” is supplied to cSHAKE exactly as encoded in the reference code constants `rcs256_name`, `rcs512_name` . Every 32-byte block of *rk* is an independent sub-key; no linear relation survives the sponge permutation.

3.4 Counter-mode keystream

```
for j = 0 .. ⌈|m|/b⌉-1 :
    s_j = E_k( counter = n + j )      // addition mod 2^256
    c_j = m_j XOR s_j
```

The counter is a big-endian 256-bit integer; increment wraps mod 2^{256}. Parallel blocks can be encrypted independently.

3.5 Authenticated-encryption interface

1. **Initialise MAC**
   `Mstate = M_mk( ad )`
2. **Encrypt message** (CTR as above) obtaining *c*.
3. **Finalise MAC**
   `tag = M_mk( ad ‖ n ‖ c ‖ l )`
4. **Output** `c ‖ tag`

During decryption the same string (ad ‖ n ‖ c ‖ l) is re-hashed; plaintext is released only if the recomputed tag matches the received tag in constant time. The implementation follows this order exactly (`qsc_rcs_transform` and `qsc_rcs_set_associated`) .

3.6 Complete algorithm summary

```
Setup( k, i, n, ad ):
    (rk, mk) ← key-expansion
    Mstate   ← M_mk( ad )

Encrypt( m ):
    c ← CTR( E_rk, n, m )
    tag ← M_mk( n ‖ c ‖ l )
    return c ‖ tag

Decrypt( c ‖ tag ):
    if M_mk( n ‖ c ‖ l ) ≠ tag :  return FAIL
    m ← CTR( E_rk, n, c )
    return m
```

3.7 Parameter sets (summary)

- **RCS-256** k = 256 bits, r = 22, tag = 256 bits, cSHAKE-256 / KMAC-256.
- **RCS-512** k = 512 bits, r = 30, tag = 256 *or* 512 bits, cSHAKE-512 / KMAC-256 or KMAC-512.

3.8 RCS Implementations

RCS has been implemented in two cryptographic libraries; in C++ in the CEX++ library [18], and in Ansi C in the QSC library [19]. A specification for RCS, including known answer tests is also available [20].

4 The RCS Cipher Construction: A Detailed Justification of the Design

4.1 Motivation for a Wide-Block Rijndael Core

- **Birthday-bound headroom.** A 256-bit block raises the collision threshold from 2^64 blocks (as in AES-128) to 2^128 blocks. Even petabyte-scale systems cannot approach that limit, removing the re-keying constraints that NIST SP 800-38D places on AES-GCM [4]. In practice, a wider block means an implementation can encrypt significantly more data under one key before needing to re-key for safety.
- **Increased trail weight.** Doubling the state width (from 128 to 256 bits) preserves a high minimum differential probability weight. In fact, any 4-round differential or linear trail in Rijndael with a 256-bit block involves at least 25 active S-boxes [9] – a very high diffusion level. This increased diffusion raises the complexity of all known classical cryptanalysis. For reference, the best published cryptanalysis of Rijndael-256 (256-bit block) in the single-key setting covers at most 10 of its 14 rounds [17]. RCS's wide-block core thus provides a comfortable security margin against differential, linear, and integral attacks (see §5).
- **Hardware feasibility.** The round function re-uses the standard AES S-box and MixColumns operations; on AES-NI-capable CPUs, a 256-bit (wide-block) implementation can achieve multi-gigabyte per second throughput using existing instruction sets. This design will continue to scale with increasing vector register sizes, as it leverages the same byte-level parallelism as AES.

4.2 The cSHAKE-Derived Key Schedule

- **Full-entropy subkeys.** RCS replaces the linear AES-256 key schedule with a cSHAKE XOF-based schedule. cSHAKE absorbs the entire master key (256 or 512 bits) and a function-name string (`"RCS-256"` or `"RCS-512"`), then squeezes out (r + 1) × 32 bytes of round keys. Because the Keccak-f[1600] permutation behaves like a random function, each 32-byte output block is computationally independent of the others [10]. This ensures each round key is effectively random and unrelated to the others, eliminating low-weight patterns.

- **Eliminating linear relations.** The short recurrence in the AES-256 key schedule (where a 32-bit word influences the next round only linearly) is a known weak point that enabled related-key rectangle and boomerang attacks [2]. In RCS, the XOF-based schedule has no linear recurrence: an adversary must brute-force 2^256 (or 2^512) possibilities to relate keys, instead of solving a simple linear system. This design choice closes off the related-key and slide attack avenues that plagued AES-256.
- **Tweakability.** The *CustomizationString* parameter of cSHAKE serves as a built-in tweak. Implementations can domain-separate keys by protocol, device, or context simply by choosing a different customization string, without altering the cipher core. This aligns exactly with the use-case envisioned in NIST SP 800-185 §4.2 for domain separation via cSHAKE [7], providing flexibility (crypto-agility) in deployments.
- **Quantum soundness.** Using a sponge with capacity ≥ 2λ (512-bit capacity for λ=256) means the keyed permutation is *collapsing*, a property which implies post-quantum pseudorandomness. Unruh proved that a sponge-based construction with sufficient capacity remains indistinguishable from a random function even against quantum adversaries [8]. Thus, an attacker in the strong quantum model (Q2) cannot distinguish RCS's round keys from random without breaking the Keccak-f[1600] permutation itself. This gives RCS a solid security foundation against quantum attacks on the key schedule.

4.3 KMAC-256/512 as the MAC Component

- **Classical strength.** A 256-bit tag provides 2^256 security against forgery in the classical setting, doubling the strength of traditional 128-bit tag schemes like GMAC or Poly1305. This substantially exceeds typical security targets and ensures a very low probability of a successful forgery by any classical adversary.
- **Quantum resilience.** The collapsing property of KMAC's underlying sponge means that even a quantum adversary gains no structural advantage in forging: the best known attack is essentially a Grover search for a valid tag, which costs about 2^128 operations (for a 256-bit tag) [8]. In contrast, polynomial MACs like GHASH (in GMAC) or Poly1305 succumb to quantum period-finding (Simon's algorithm): an adversary can recover the secret key with on the order of 128 quantum queries in superposition [16]. This stark difference in the Q2 model is the decisive rationale for adopting KMAC in RCS.
- **Single-pass AEAD.** RCS employs an Encrypt-then-MAC approach. Encrypting with the wide-block cipher and then authenticating with KMAC yields IND-CCA encryption and INT-CTXT integrity, as proven by Bellare and Namprempre for classical settings [11]. Moreover, this approach extends to quantum IND-CCA (QIND-CCA) security under the framework of Gagliardoni–Hülsing–Scholten [15], because KMAC's tag function is a quantum-secure PRF. In summary, the encryption and authentication components compose securely in both classical and quantum adversary models.

4.4 Round Count Selection (22 / 30)

- **Doubling the best break.** The strongest public attacks on AES-256 (with its original key schedule) recover the key for 11 rounds in about 2^70 time [2]. Based on this, cryptographers often apply a "twice the rounds" heuristic. Indeed, NIST's AES selection report and Schneier's rule-of-thumb both recommend using roughly twice the number of

rounds of the best known attack [4][5]. Following this guidance, RCS-256 uses 22 rounds (≈ 2 × 11). RCS-512, with its larger key, adds an extra safety cushion for future-proofing by choosing 30 rounds (an 8-round increase over 22).

- **Empirical margin.** No published differential trail, integral, rectangle, or SAT-based attack comes close to 22 rounds. In fact, the known cryptanalytic results on Rijndael-256 reach at most 10 rounds (out of 14) as noted above [17], which is less than half of RCS-256's 22 rounds. This leaves at least 12 rounds of security margin for RCS-256. Such a margin satisfies the traditional "full cipher minus two rounds" design guideline, ensuring RCS has a buffer above and beyond all known analysis.

4.5 Authenticity in the Quantum (Q2) Model

- **Simon vulnerability removed.** The algebraic structure of GMAC's GHASH and of Poly1305 (both essentially polynomial-evaluation MACs) provides a hidden linear relation that Simon's quantum algorithm can exploit. In particular, with on the order of 128 quantum queries, an attacker can extract the key for GHASH or Poly1305 by solving for the secret polynomial [16]. KMAC is immune to this class of attack: Keccak-f[1600]'s round function involves nonlinear bitwise operations (AND and XOR in the χ step) and does not hide a simple XOR linear combination of the key. Thus, there is no periodic structure for Simon's algorithm to find, and the superposition query attack that breaks GMAC/Poly1305 does not apply to KMAC.
- **Capacity ≥ 2λ criterion.** RCS was designed so that the hash function capacity (c) is at least twice the desired tag security level (λ). Concretely, with c = 512 bits and λ = 256, KMAC-256 meets the collapsing condition per Unruh's bound [8]. This guarantees that even a quantum attacker making queries in superposition cannot forge a valid tag with less work than a brute-force search on the tag (on the order of 2^128 operations for a 256-bit tag). In other words, RCS's authentication remains unforgeable in the Q2 model, matching the strength of its confidentiality component.

4.6 Implementation & Performance Outlook

- **Hardware acceleration.** Upcoming CPU instruction set extensions will accelerate both the key schedule and MAC in hardware. Intel's SHA-3 instructions ("K-EXT"), ARMv8.2-A's SHA-3 extension, and the RISC-V Crypto v0.4 draft all introduce single-round Keccak-f[1600] operations in hardware [13]. Preliminary benchmarks indicate several-fold speedups (on the order of 5× over optimized AVX2 software), which effectively closes the performance gap with AES-NI accelerated AES. In sum, as SHA-3 support becomes ubiquitous in hardware, RCS will achieve throughput comparable to AES-GCM on modern platforms.
- **Constant-time implementation.** Both components of RCS lend themselves to constant-time, side-channel-resistant code. The Keccak permutation is bit-sliced and uses no lookup tables, and the AES round function (S-box and MixColumns) can likewise be implemented without secret-dependent tables (as in the AES-NI or bitsliced implementations). Furthermore, masking countermeasures developed for SHA-3 can be directly applied to cSHAKE and KMAC, meaning established techniques exist to protect RCS against side-channel attacks.

- **Code size economy.** RCS's design minimizes the footprint for embedded implementations. A single 1600-bit permutation serves both the key schedule (cSHAKE) and the MAC (KMAC). This unified approach means that an implementation of RCS needs only the AES round function and the Keccak-f[1600] permutation. In contrast, a conventional AEAD like AES-GCM requires implementing AES, GHASH (for GMAC), and potentially SHA-2 for hashing in certain protocols. RCS can therefore reduce firmware size by consolidating cryptographic components.

4.7 Why RCS Meets "Future-Proof" Criteria

- **Confidentiality:** 256-bit (and 512-bit) keys ensure that Grover's algorithm offers no practical threat – even a quantum computer provides at most a square-root speedup, leaving ≈128-bit effective security for RCS-256. This exceeds the ≥128-bit post-quantum security target for long-term confidentiality.
- **Authenticity:** KMAC's design withstands advanced quantum forgery attacks (e.g., Simon's algorithm), whereas prevailing MACs like GMAC or Poly1305 do not. By avoiding structures with hidden periods, RCS's MAC remains secure against both classical and quantum forgers.
- **Safety margin:** With 22 rounds (RCS-256) and 30 rounds (RCS-512), the cipher core satisfies the "twice the best-known break" heuristic and provides more security margin than AES-256 had. RCS's round count was chosen to anticipate further cryptanalysis, ensuring a buffer of several unbroken rounds even if attacks improve.
- **Scalability:** As noted, pending SHA-3 hardware instructions will make RCS run as efficiently as AES-based solutions on new CPUs. RCS is designed to take full advantage of such parallelism and will scale with future processor advancements.
- **Agility:** The use of cSHAKE for key scheduling (with custom strings as tweaks) and the flexible tag length in KMAC (256 or 512 bits) mean that RCS can be adapted to new requirements without redesigning the algorithm. This gives RCS the agility to meet diverse use-cases and to integrate into evolving protocols (for example, by choosing different tweak strings per application or increasing the tag length if needed), truly meeting the criteria for a future-proof cipher.

5 The RCS Cipher Construction: A Brief Cryptanalytic Study

Security goals and threat model

RCS is intended to deliver:

- IND-CPA confidentiality and INT-CTXT authenticity for arbitrary-length messages,
- ≥ 128-bit post-quantum strength in the "Q1" model (classical oracle, quantum adversary),
- non-trivial authenticity in the stronger "Q2" model (superposition oracle), and
- side-channel hardenability comparable to AES-NI or ChaCha20-SIMD code paths.

Unless explicitly stated, the adversary controls all inputs (nonce reuse aside) and may obtain unlimited encryption/verification queries.

Round function and wide-block diffusion

The 256-bit Rijndael round uses the same SubBytes → ShiftRows → MixColumns pipeline as AES-128; the state is simply widened from four to eight 32-byte columns [9]. In the original wide-trail strategy, Daemen and Rijmen proved that for a 128-bit state, any four-round differential or linear trail has at least 25 active S-boxes [12]. No peer-reviewed result has yet derived a corresponding bound for the 256-bit (eight-column) form, but the diffusion pattern is identical in each column, so we can assume a similar minimum applies in the wider state.

As for distinguishers, the strongest published result on Rijndael-256 breaks eight rounds (integral attack [3]). For AES-256, related-key and boomerang techniques extend up to ten or eleven rounds [2]. Those attacks rely on the weak key schedule rather than the round function itself, and have not been ported to a wide-block, schedule-free variant. No *public* literature reports differential, linear, or algebraic attacks beyond those round counts. Consequently, with RCS's choice of 22 rounds, the design enjoys at least a 14-round security margin over the best differential and integral cryptanalysis currently in print.

Key-schedule hardening via cSHAKE

Because every round key is derived by an XOF after the entire master key has been absorbed:

- Related-key, slide, and rectangle attacks that rely on slow linear diffusion in the schedule no longer apply. No predictable low-Hamming-weight mask links rk[t] and rk[t+1].
- Algebraic meet-in-the-middle attacks lose the crucial "short recurrence" exploited by Biryukov et al.; brute-forcing all round keys now costs 2^256 for RCS-256 and 2^512 for RCS-512 – exceeding brute-force search on the master key itself.

Classical attack surface

- **Differential/linear cryptanalysis:** No *known* differential or linear attack threatens more than ~11 rounds of AES-256 [2], and none have been published for the 256-bit block variant beyond 8 rounds [3]. Thus, 22 rounds of RCS provide a very comfortable safety margin against these attacks.
- **Integral and rebound attacks:** The known 8-round integrals on Rijndael-256 [3] cannot be directly carried over when the state is doubled, since the rebound phase must now cover two additional S-box layers – likely raising data complexity above 2^256 (effectively impractical).
- **Related-key / boomerang:** The linear-recurrence distinguisher of Shamir's group (against AES-256) does not survive the cSHAKE-derived key schedule. A naive related-key attacker would have to try on the order of 2^256 keys – infeasible [2].
- **Algebraic and SAT attacks:** No feasible algebraic or SAT-based attack on RCS is known; even reduced-round instances resist such approaches.

Quantum attack surface

- **Grover search:** A quantum attacker gains a quadratic speedup on brute-force. RCS-256 therefore offers ~2^128 operations of security, meeting NIST's highest post-quantum security level (AES-256). RCS-512 offers a margin beyond that (~2^256 operations).
- **Simon-type forgery on the MAC:** The collapsing property of KMAC (capacity ≥ 2λ) means the full tag strength is retained [10]; RCS sets capacity = 512 bits. Accordingly, no polynomial-time Q2 forgery attack is known.
- **Hidden shift on permutation:** Keccak-f[1600] has withstood all known hidden-shift (quantum) distinguishers up to 12 rounds, and no attack extends to the full 24 rounds [10]. Hence, the keyed sponge inherits only generic Grover-type bounds.

AEAD composition security

RCS follows the standard Encrypt-then-MAC paradigm with a strong PRP (CTR-Rijndael-256) and a collapsing MAC, yielding IND-CCA confidentiality and INT-CTXT authenticity in the classical sense (Bellare–Namprempre 2000) [11]. The same construction achieves QIND-CCA security under the framework of Gagliardoni–Hülsing–Schaffner (2016) [11], since the keyed sponge is a quantum-secure PRF. (Tag verification precedes decryption, eliminating any padding oracle risks.)

5.7 Comparison with AES-256-GCM and ChaCha20-Poly1305

Property	RCS-256-KMAC	AES-256-GCM	ChaCha20-Poly1305
Block / state	256 bits	128 bits	512-bit state, 32-bit words
Rounds	22	14 + GHASH	20
Key schedule	cSHAKE (non-linear, tweakable)	linear AES schedule	none (key injected each block)
MAC type	keyed sponge (collapsing)	GHASH (polynomial)	Poly1305 (polynomial)
Q1 tag security	128 bits	64 bits	64 bits
Q2 authenticity	secure (no Simon)	**broken** (O(128) queries)	**broken** (O(128) queries)
Best public break	none below 14 rd (cipher)	11-rd related-key (AES-256)	7-rd rotational (ChaCha)
Projected SHA-3 hw	yes (2026+)	n/a	n/a

Long-term security implications

- **Wide-block confidentiality:** A 256-bit block doubles the birthday bound, eliminating the record-length limitations that force periodic rekeying in AES-GCM (2^64 blocks).
- **Quantum-robust authenticity:** By deploying KMAC, RCS forecloses an entire class of Simon-style forgeries that feature prominently in post-quantum risk assessments.
- **Key-schedule agility:** The cSHAKE derivation allows domain-specific diversification (using custom strings) and forward compatibility with future SHA-3 extensions.

- **Hardware trajectory:** Intel, ARM, and RISC-V have all announced SHA-3 instruction support; RCS will therefore benefit from the same single-cycle per-round performance boost that made AES-NI decisive in 2010.
- **Layered defense:** Deployments can retain AES-256-GCM for legacy interoperability while adding RCS as a "quantum-hardened" option negotiated via TLS cipher-suite agility, mitigating migration risk.

Interim conclusion

The public record shows **no practical attack** on more than half the RCS-256 round count (i.e., >11 rounds) [2], while the keyed-sponge MAC withstands both classical and quantum forgery models [10]. Compared to AES-GCM and ChaCha20-Poly1305, RCS increases brute-force cost, eliminates Simon-style vulnerabilities, and aligns with incoming SHA-3 hardware support – collectively justifying its claim to *future-proof* AEAD security.

6 Conclusion

The weak diffusion properties of the Rijndael key schedule is an established and provable flaw in the Rijndael cipher design. The related subkey attack which leveraged the diffusion-weak key expansion function has proven this conclusively [14]. To implement a wide block Rijndael with this known weakness in the key schedule, which in effect will amplify this weakness, is to invite further (and possibly devastating) attacks on the cipher. It is however, possible to strongly mitigate attacks against the Rijndael construction, and at the same time restore the margins of rounds-based security. For that to be accomplished, a strong pseudo-random function must be used to replace the key schedule, which in the case of Keccak, could also mitigate other attacks while providing a future-secure MAC function.

As to 'complicating the design' by merging these two primitives, this is no less complicated than the AES-GMAC or Chacha-Poly1305 constructions, and further, an authenticated stream cipher is much simpler to use, and provides the benefit of an AEAD authenticated stream cipher in a single design unit.

Keccak is one of the most thoroughly cryptanalyzed constructions in modern times, it has shown strong resistance to many and varied attack vectors including those in the Q1/Q2 models. Keccak is standardized, widely implemented, and will become an essential component in post-quantum cryptographic development.

CPU manufacturers are planning integration of Keccak instructions into their products, and as the post-quantum threat grows that will become essential. This accelerated performance provided by embedded instructions creates a perfect marriage between Rijndael and Keccak, one which can be future-leveraged to create fast and quantum-secure cipher constructions. RCS is just such a construction, using the proven security of the Rijndael rounds function and combining it with

the resilient Keccak sponge construction, to create a truly post-quantum secure authenticated cipher.

If Rijndael-256 is to be resurrected, it should be done in a way that addresses weaknesses in the design, restores the security margins, and creates a future-proof authenticated cipher. RCS does just that, and can take Rijndael into the quantum era with full confidence, and provide a cipher with the necessary crypto-agility to confront the enormous technological accelerations that are bound to happen in this century. RCS provides a secure cipher not just for the known threats, but for the many unknown threats to our security that are certain to evolve in the times ahead.

References

No.	Bibliographic entry	URL
[1]	N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, D. Wagner, J. Whiting, T. Kohno, "Improved Cryptanalysis of Rijndael," *Fast Software Encryption 7*, 2000, pp. 213-230.	https://link.springer.com/content/pdf/10.1007/3-540-44706-7_15.pdf
[2]	A. Biryukov, O. Dunkelman, N. Keller, A. Khovratovich, A. Shamir, "Key-Recovery Attacks on AES-256 up to 10 Rounds," *EUROCRYPT 2010*, LNCS 6110, pp. 299-319.	https://iacr.org/archive/eurocrypt2010/66320198/66320198.pdf
[3]	S. Galice, M. Minier, "Improved Truncated Differential Attacks on Rijndael-256," *AFRICACRYPT 2008*, LNCS 5023, pp. 112-126.	https://www.researchgate.net/publication/221462133_Improving_Integral_Attacks_Against_Rijndael-256_Up_to_9_Rounds
[4]	National Institute of Standards and Technology, *Report on the Development of the AES*, NIST IR 6396, 2000.	https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=151226
[5]	B. Schneier, "New Attack on AES," *Schneier on Security* (blog), 18 Aug 2011.	https://www.schneier.com/blog/archives/2011/08/new_attack_on_a.html
[6]	G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, "The Keccak Reference," v3, 2011.	https://keccak.team/files/Keccak-reference-3.0.pdf
[7]	National Institute of Standards and Technology, *SP 800-185 — SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, ParallelHash*, 2016.	https://doi.org/10.6028/NIST.SP.800-185
[8]	D. Unruh, "Quantum Security of Sponge-Based Constructions," *EUROCRYPT 2016*, LNCS 9666, pp. 551-577 / Cryptology ePrint 2017/282.	https://eprint.iacr.org/2017/282
[9]	J. Daemen, V. Rijmen, *"AES Proposal: Rijndael,"* NIST AES submission, 1999.	https://csrc.nist.rip/encryption/aes/rijndael/Rijndael.pdf

[10]	Keccak Team, “Third-Party Cryptanalysis of Keccak,” living document, accessed 10 May 2025.	https://keccak.team/third_party.html
[11]	M. Bellare, C. Namprempre, “Authenticated Encryption: Relations and Analysis,” *Journal of Cryptology* 21 (4) 2008, pp. 469-491.	https://eprint.iacr.org/2000/025.pdf
[12]	National Institute of Standards and Technology, *SP 800-38D (rev 1) — Galois/Counter Mode*, 2023.	https://doi.org/10.6028/NIST.SP.800-38D
[13]	RISC-V Cryptography Extension Working Group, *“Scalar SHA-3 Instruction-Set Draft,”* v0.4, March 2024.	https://github.com/riscv/riscv-crypto
[14]	A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, A. Shamir, “Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds,” *Cryptology ePrint Archive*, Report 2009/374, 2009.	https://eprint.iacr.org/2009/374.pdf
[15]	T. Gagliardoni, A. Hülsing, C. Schaffner, "Semantic Security and Indistinguishability in the Quantum World," CRYPTO 2016, LNCS 9816, 2016, pp. 60–89.	https://doi.org/10.1007/978-3-662-53015-3_3
[16]	X. Bonnetain, M. Naya-Plasencia, “Hidden Shift Quantum Cryptanalysis and Implications,” ASIACRYPT 2018, LNCS 11272, pp. 560–592, 2018.	https://doi.org/10.1007/978-3-030-03326-2_19
[17]	Y. Liu, Y. Shi, D. Gu, B. Dai, F. Zhao, W. Li, Z. Liu, Z. Zeng, “Improved impossible differential cryptanalysis of large-block Rijndael,” *Science China Information Sciences*, vol. 62, no. 3, art. 32101, 2019.	https://doi.org/10.1007/s11432-017-9365-4
[18]	The CEX++ post-quantum cryptographic library.	https://github.com/QRCS-CORP/CEX
[19]	The QSC post-quantum cryptographic library	https://github.com/QRCS-CORP/QSC
[20]	The RCS cipher specification.	https://qrcs-corp.github.io/QSC/pdf/RCS_Specification.pdf