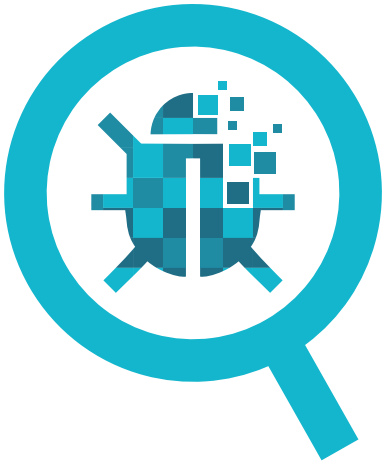


COLDCARD Seed Generation Review

Prepared by

Organization	Keylabs
Contact	contact@keylabs.io
Classification	PUBLIC RELEASE
Client	Coinkite
Version	1.0
Date	October 15, 2026

Executive Summary

This report analyzes the random number generator (RNG) stack in the latest COLDCARD firmware — Mk5 **5.6.3** (`7b4d6cc4`) and Q **1.5.3Q** (`282b8a4f`). The remediation of the July 2026 entropy defect was found to be sound: the fix is confirmed present in the shipped, signed release binaries, and the controls it introduces are assessed as sufficient to close the vulnerability class. The review also covers the immediate predecessors 5.6.2 / 1.5.2Q.

The assessment is grounded in primary sources rather than vendor description. Every conclusion is traced to the firmware source at the audited release tags, to the pinned libngu and MicroPython submodules, and to the shipped, signed release binaries. The remediation is examined at each layer of the entropy path — pre-processor configuration, build system, boot sequence, and seed construction — and confirmed independently in the disassembled binaries for all four targets, with a pre-fix image used as a negative control to show the method detects the defect when it is present. The statistical behavior of the original defective generator was reproduced in simulation to establish why the fault escaped the device's existing health checks.

Contents

1 Background **3**

1.1 Root cause 3

2 Fix Review **5**

2.1 Changes 5

2.2 Fix assessment 7

2.3 Binary verification 8

3 Conclusion **9**

Background

In July 2026 a defect was discovered and disclosed in the COLDCARD firmware that resulted in vulnerable seed generation on affected releases. The cause was a firmware configuration error rather than a hardware fault: seed entropy was drawn from a software pseudo-random generator compiled into the firmware instead of from the hardware random number generator, reducing the unpredictability of the keys produced. The issue has since been fixed, and the corrected firmware is the subject of this report.

The failure was one of entropy provenance, not of the output's statistical quality. A software generator's output is itself statistically uniform, so the deficiency lay in the material used to seed it rather than in any observable property of its output. Output health tests cannot separate a well-seeded generator from a poorly-seeded one; the defect was therefore detectable only by establishing where the entropy actually originated — the question this audit centers on.

Coinkite released corrected firmware across all supported product tracks and published migration guidance. Installing the fixed firmware corrects future seed generation but does not repair a seed already generated on vulnerable firmware; any funds secured by such a seed must be moved to a new seed created on corrected firmware. The current recommended releases are Mk5 5.6.3 and Q 1.5.3Q, released 30 September 2026; corrected firmware has been available since the 5.6.0 and 1.5.0Q releases.

Root cause

The code facts below are verified against the firmware source directly. Impact figures reported elsewhere — including Coinkite's advisory estimate of roughly 72 bits of effective entropy — are not something we independently established and are not relied on here.

```
22 #ifdef MICROPY_PY_STM
23 // ports/stm32/rng.c
24 extern uint32_t rng_get(void);
25 # define CHIP_TRNG_SETUP()
26 # define CHIP_TRNG_32()         rng_get()
27
28 # ifndef MICROPY_HW_ENABLE_RNG
29 # error "get a HW TRNG plz"
30 # endif
31 #endif
```

Listing 1: `libngu ngu/random.c` (pre-fix pin 537519a) — the guard that failed

COLDCARD boards define the macro as `0`. `#ifndef` tests definedness, not value, so the guard passed and `rng_get()` resolved to MicroPython's `pyb_rng_yasmarang` software PRNG, seeded from `UID_low32 ^ SysTick`, `RTC->TR` and `RTC->SSR` — device-identity and timer state, not a hardware noise source. Effective seed entropy was therefore drastically reduced. The defect was present in earlier firmware; on Mk4/Mk5/Q a 32-bit SE reseed narrowed but did not close the gap.

The defective output was nonetheless statistically uniform — we confirmed by simulation that the fallback passes FIPS CRNGT, SP 800-90B RCT/APT, SP 800-22, and COLDCARD's

`assert len(set(seed)) > 4`. The defect was low seed entropy, not non-uniform output, so no output health test detects it; only verification of the source's provenance does.

Fix Review

This review covers the current COLDCARD releases — Mk5 **5.6.3** and Q **1.5.3Q** — and their immediate predecessors 5.6.2 / 1.5.2Q. The fix prevents the faulty linkage — identified under Root cause — at four independent points, as detailed below.

Changes

The remediation is a small set of code changes, shown below, all verified against the audited firmware tree (`2026-09-03T1541-v5.6.2` / `2026-09-30T2026-v5.6.3` and the pinned `libngu @ 4d9af84` / `micropython @ 3d227fe` submodules). 5.6.2 and 5.6.3 carry the same RNG-path C code; 5.6.3's later Python features (Codex32, Shamir backups) do not touch the entropy path.

The guard shown under Root cause is replaced by one that checks the macro's value and a dedicated external-provider flag, with the backend selection moved into a new `random_backend.h`:

```
26 #elif defined(MICROPY_PY_STM)
27 # if MICROPY_HW_ENABLE_RNG != 1 && NGU_STM32_EXTERNAL_RNG_GET != 1
28 #  error "MICROPY_HW_ENABLE_RNG=1 or NGU_STM32_EXTERNAL_RNG_GET=1 required"
29 # endif
```

Listing 2: `libngu ngu/random_backend.h` (fixed) — value-checked guard

The software fallback is also removed from the link entirely: the upstream `stm32/rng.o` is replaced by an empty object and the Yasmarang symbol is poisoned so it cannot be linked (same on Q1):

```
101 $(BUILD)/rng.o: CFLAGS += -Dpyb_rng_yasmarang=error-do-not-want-this
102 $(BUILD)/rng.o:
103     $(ECHO) "SKIP stm32/rng.c"
```

Listing 3: `stm32/COLDCARD_MK4/mpconfigboard.mk` — fallback PRNG neutralized

The boot reseed previously took only the first four bytes (32 bits) of the SE1‖SE2 digest; it now passes the full 32-byte digest, with length asserts:

```
43       a = callgate.read_rng(1)        # SE1
44       b = callgate.read_rng(2)        # SE2
45 +     assert len(a) == 32
46 +     assert len(b) == 8
47
48       n = ngu.hash.sha256d(a+b)
49 -     n, = ustruct.unpack('I', n[0:4])
50
51       ngu.random.reseed(n)
```

Listing 4: `shared/mk4.py rng_seeding()` — 32-bit truncation removed

A build-time check (`rng-code-check`) fails the release build unless the linked objects prove the entropy path reaches the hardware driver (abridged; full recipe continues to the relocation-chain checks):

```
63 rng-code-check:
64     @upstream_symbols="$$($(NM) --defined-only $(BUILD_DIR)/rng.o)" || exit $$?; \
65     if test -n "$$upstream_symbols"; then \
66         echo "ERROR: micropython's stm32/rng.o must not define any symbols"; \
67         exit 1; \
68     fi; \
69     board_symbols="$$($(NM) --defined-only \
70         $(BUILD_DIR)/boards/$(BOARD)/rng.o)" || exit $$?; \
71     if ! printf '%s\n' "$$board_symbols" \
72         | grep -Eq '^[[:xdigit:]]+[[:space:]]+T[[:space:]]+rng_get$$'; then \
73         echo "ERROR: board rng.o does not define global rng_get"; \
74         exit 1; \
75     fi
```

Listing 5: `stm32/shared.mk` — `rng-code-check` (abridged)

Beyond the guard and reseed, the audited firmware uses a SHA-256 Hash_DRBG (SP 800-90A) seeded from 1024 TRNG bits, XORing a fresh TRNG word into each output (`libngu random.c:41-93`); it samples SE1 and SE2 per seed (`shared/seed.py`); and a boot self-test (`COLDCARD_MK4/rng.c:180-210`, `modckcc.c:282-288`) halts the device if `rng_get()` does not reach the hardware path.

Fix assessment

The incident required `ngu.random` to resolve to a software PRNG. Two independent controls prevent this; both are verified in source and in the shipped binary.

Provenance, four independent layers (each sufficient alone):

1. Compile time — value-checked `#error` guard (`random_backend.h:27-29`).
2. Build time — upstream `rng.o` built from `/dev/null`, `pyb_rng_yasmarang` poisoned (`mpconfigboard.mk:99-104`).
3. Build time — `rng-code-check` verifies the relocation chain terminates at the board's hardware `rng_get` (`shared.mk:62-102`).
4. Boot time — `rng_selftest()` draws through `rng_get` before the VM starts and hard-faults on failure (`rng.c:180-210`, `modckcc.c:282-288`).

Construction, removing single-source dependence on the seed path:

- SHA-256 Hash_DRBG seeded from 1024 TRNG bits; each output word XORed with a fresh TRNG word (`libngu random.c:41-93`).
- Per-seed conditioning `sha256d(TRNG || SE1 || SE2)` — TRNG and SE1 fresh, SE2 a static per-device value (`shared/seed.py`).
- 256-bit SE boot reseed (previously 32-bit).

Binary verification

The fix was confirmed in the shipped, signed binaries, not only in source. We disassembled the released DFU images for all four current targets (Mk5 5.6.3 and 5.6.2, Q 1.5.3Q and 1.5.2Q) and, for comparison, a pre-fix release (Mk5 5.5.1), after checking each image's SHA-256 against the PGP-signed `signatures.txt`. The disassembly below was produced with GNU `objdump` and reproduced independently.

The difference is unambiguous. In the current firmware the entropy routine operates on the STM32 RNG peripheral and reads its data register; in the old firmware the generator is self-contained software arithmetic, seeded by a fixed constant and touching no peripheral.

```
0807655c <rng_get_or_fault>:
  807655c:  push   {r0,r1,r4-r8,lr}
  8076564:  ldr    r4, [pc,#168]    ; r4 = 0x50060800  (STM32 RNG base)
  8076570:  ldr    r3, [r4,#4]      ; RNG->SR  status
  8076572:  tst.w  r3, #0x44        ; SEIS | SECS  seed-error check
     ...
  80765ca:  ldr    r0, [r4,#8]      ; read RNG_DR @ 0x50060808  <- hardware TRNG
  80765cc:  cmp    r0, #0           ; reject a zero word
```

Listing 6: Current firmware (Mk5 5.6.3) — `rng_get` reads the hardware RNG data register

```
08069c34 <my_yasmarang>:
  8069c36:  ldr    r4, [pc,#68]     ; generator state in RAM (no peripheral)
  8069c46:  mla    r1, r1, r2, r3   ; Yasmarang step - pure register arithmetic
  8069c4c:  eor.w  r2, r2, r1, ror #30
  8069c50:  mov.w  r0, r1, ror #29
  8069c64:  eor.w  r0, r0, r0, lsr #18
  8069c6a:  eor.w  r0, r0, r2, lsl #5
08115a68 <seed constant>:
  8115a68:  6f e2 8c 0a             ; pad = 0x0a8ce26f
```

Listing 7: Old firmware (Mk5 5.5.1) — the generator is software arithmetic seeded by a constant, no peripheral read

The constant `0x0a8ce26f` that seeds the old generator is present in the pre-fix image and absent from the current firmware, which instead reaches `RNG_DR` through a single `rng_get`. The RNG C code is byte-identical between each product's current and preceding release (5.6.2 / 5.6.3 and 1.5.2Q / 1.5.3Q).

Conclusion

The July 2026 incident arose from a firmware configuration error that drew seed entropy from a software pseudo-random generator instead of the STM32 hardware TRNG. This review examined the current COLDCARD firmware — Mk5 5.6.3 and Q 1.5.3Q — in both source and the shipped, signed binaries, following the entropy path from the hardware RNG through the Hash-DRBG and the secure-element mixing into the seed. The remediation is present and correctly built: the faulty linkage is prevented at four independent points — a value-checked compile-time guard, two build-time checks, and a boot-time self-test — and the boot reseed carries the full secure-element digest. Disassembly of the released images confirms that every entropy consumer reaches the hardware RNG data register and that no software fallback remains on the entropy path. This assessment was scoped to that fix. On the current firmware, the July 2026 seed-generation defect is fully closed.