

KWireGuard: Design of Kernel Cryptographic Layer for WireGuard Based on Korean Block Ciphers

Min-Woo Park*, Jae-Kyung Sagong*, Ho-Geun Yoon*, Dongju Kim*[○], Sangsoo Park**, Tae-Young Byun**
*School of Computer Software, Daegu Catholic University, **NanoIT Co. Ltd.
e-mail: {minwoo003655, sgjk0213, yunhogeun8}@gmail.com*, deekim@cu.ac.kr*, dori@nanoit.kr**, tybyun@cu.ac.kr**

I. Introduction

- **Background:** WireGuard (Linux 5.6+, Noise_IKpsk2) [1] supports only ChaCha20-Poly1305, blocking deployment where KCMVP algorithms are mandated
- **Scope:** Architectural prerequisites for a validated module (CAVP units, POST, RNG binding), distinct from KCMVP algorithm listing
- **Challenges:** Packet layout preservation, injective nonce-to-IV mapping, kernel memory safety, ARM64 acceleration, and no silent downgrade

II. Related Work

- **Prior Work:** Hardware AES-GCM raised gateway throughput 11-19% [4]; WireGuard outperforms OpenVPN/IPsec [5]; both use foreign standards only
- **Cipher Selection:** ARIA [9] and LEA [10] fit GCM with 128-bit blocks and 256-bit keys [8]; SEED and HIGHT are excluded [6,7]
- **ARM Acceleration:** x86 ARIA reaches 10.6× via AVX-512 [11], but ARM64 kernels lack ARIA [12,13]; ARX-based LEA suits embedded cores [14]

III. KWireGuard Design

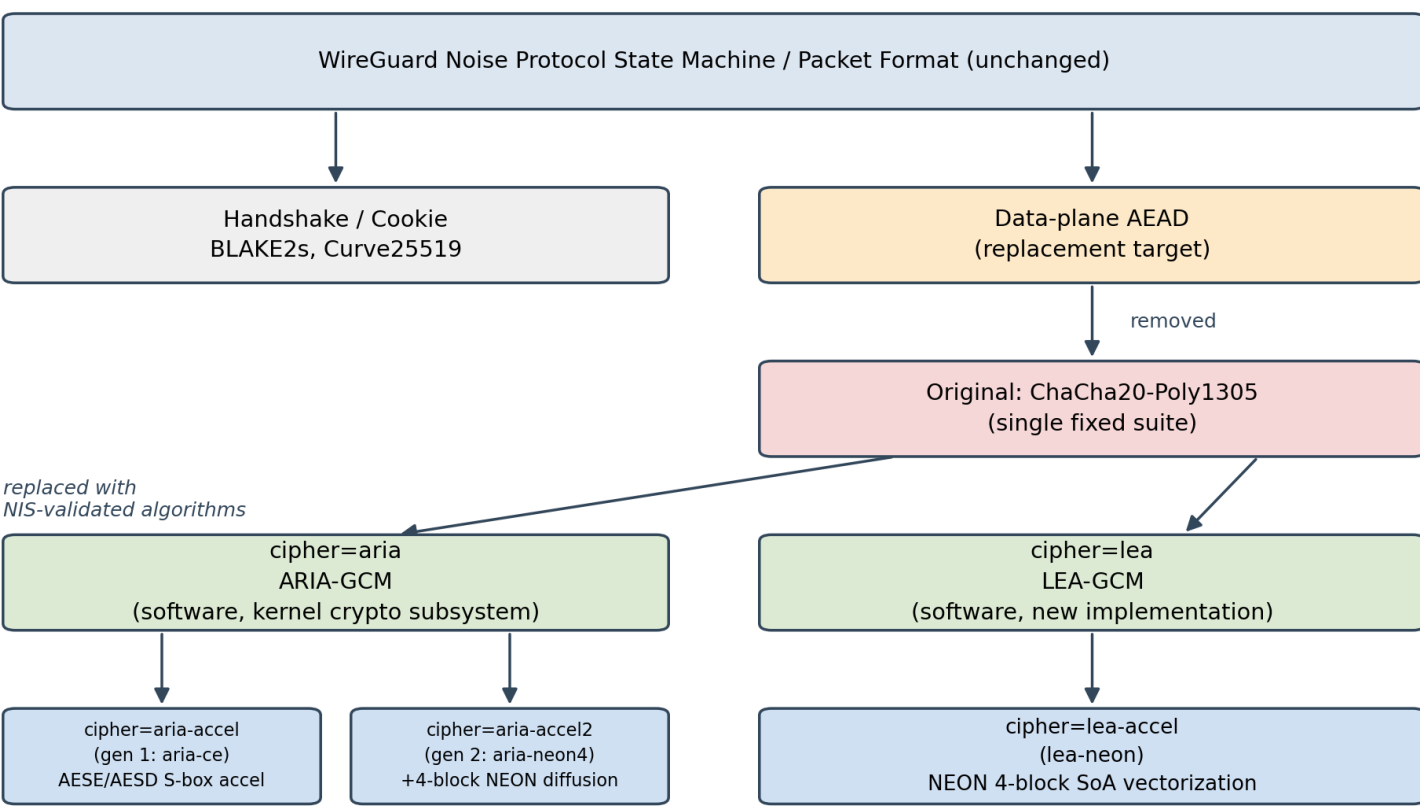


Figure 1. KWireGuard architecture

- **Threat Model:** Dolev-Yao MITM, endpoint hijacking, DoS and Oops, cache side channels, and accidental downgrade to unvalidated code
- **Static Binding:** Read-only cipher fixed at load time removes runtime locks; mismatched peers fail at the first AEAD tag check

Table 1. KWireGuard cipher modes and KCMVP validation units

Cipher Mode	Crypto API Id.	Hardware Constraints	KCMVP Validation Scope
aria	gcm(aria)	Generic pure C (No constraints)	Reference C Algorithm Validation Unit
lea	gcm(lea)	Generic pure C (Static module)	Module Internal Pure C Validation Unit
aria-accel	gcm(aria-ce)	ARMv8 Crypto Extension (AES)	Hardware Instruction Accelerated Unit 1
aria-accel2	gcm(aria-neon4)	ARMv8 Crypto + NEON SIMD	Vector Instruction Accelerated Unit 2
lea-accel (Default)	gcm(lea-neon)	ARMv8 Advanced SIMD (NEON)	Pure SIMD Vector Accelerated Unit

- **Validation Units:** Each implementation is a separate CAVP unit, registered under its own name and gated by CPU feature flags

III. KWireGuard Design (Cont.)

- **Self-Test:** Four-tier load-time KAT including AEAD tamper detection; any failure halts module loading
- **Fail-Closed:** Missing AES/ASIMD flags skip registration, so a forced mode fails AEAD allocation and the handshake is rejected
- **Entropy:** Linux CRNG is not a KCMVP DRBG; deployment binds a certified hardware RNG or the Crypto API DRBG

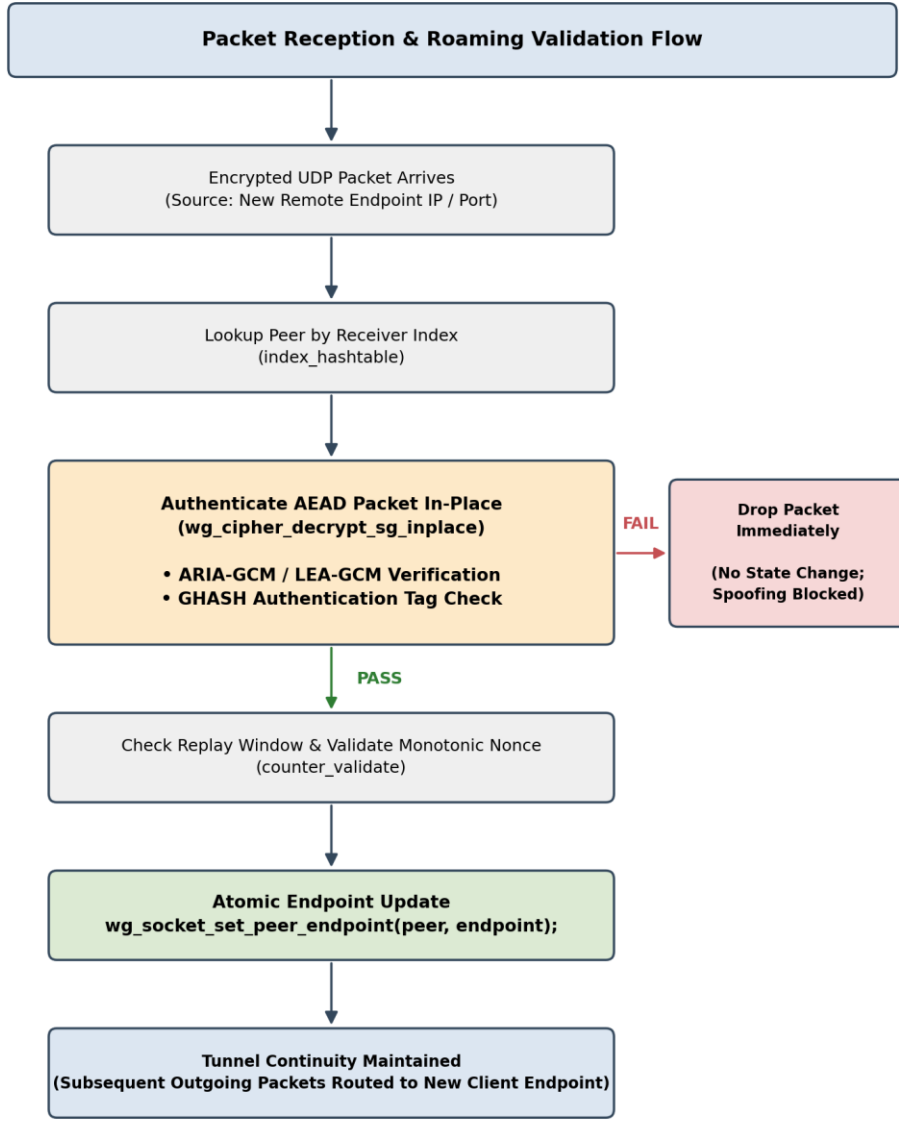


Figure 2. Roaming flowchart based on packet authentication

- **Roaming Integrity:** Endpoint updated atomically only after GHASH tag and replay window checks pass, blocking hijacking
- **Injective Nonce:** $IV = 0x00^4 \parallel le64(n)$, rekey at 2^{60} ; cookie IV takes the low 12 bytes of a counter nonce, so IV reuse is zero
- **Memory Safety:** 32-byte heap bounce buffer prevents Oops on virtually mapped stacks; zero-length plaintext maps to AAD only
- **Zero-Copy:** Per-session AEAD transforms allocated once and shared lock-free; in-place encryption reuses the tag tailroom

Table 2. ARM Cortex-A acceleration modules

Module	Instructions	Batching	Bottleneck / Optimization
ARIA-CE	AESE/AESD	1-block	GPR ↔ NEON transition bottleneck
ARIA-NEON4	Crypto + NEON	4-block SoA	Vectorized diffusion layer, ILP
LEA-NEON	NEON SIMD	4-block SoA	Pure ARX, zero register spill

- **ARIA:** S2/X2 table lookups stall in-order Cortex-A53 [15,16]; full gain needs 16-block transposition [13]
- **LEA-NEON (Default):** 4 blocks hide RAW latency [17] with 10-12 of 32 registers, constant time without S-boxes
- **SIMD Guard:** One NEON enter/exit per 4KB batch, $\approx 4-5 \mu s$ worst case on Cortex-A53, within PREEMPT_RT budget

IV. Conclusion and Future Work

- **Conclusion:** ARIA-GCM/LEA-GCM kernel layer with injective nonces, memory safety, POST, and Fail-Closed policy
- **Future Work:** Kernel implementation on physical ARM boards; measure throughput, latency, footprint, Fail-Closed, and roaming