Brief Comments on Rijndael-256 and the Standard RISC-V Cryptography Extensions

(June 26, 2025 – minor corrections)

Markku-Juhani O. Saarinen

Tampere University, Finland
markku-juhani.saarinen@tuni.fi

Abstract. We evaluate the implementation aspects of Rijndael-256 using the ratified RISC-V Vector Cryptography extension Zvkn. A positive finding is that Rijndael-256 can be implemented in constant time with the existing RISC-V ISA as the critical AES and fixed crossbar permutation instructions are in the DIEL (data-independent execution latency) set. Furthermore, simple tricks can be used to expand the functionality of key expansion instructions to cover the additional round constants required. However, due to the required additional byte shuffle in each round, Rijndael-256 will be significantly slower than AES-256 in terms of throughput. Without additional ISA modifications, the instruction count will be increased by the required switching of the EEW ("effective element width") parameter in each round between 8 bits (byte shuffle) and 32 bits (AES round instructions). Instruction counts for 1-kilobyte encryption and decryption with Rijndael-256 are factor 2.66× higher than with AES-256. The precise amount of throughput slowdown depends on the microarchitectural details of a particular RISC-V ISA hardware instantiation, but it may be substantial with some high-performance vector AES architectures due to the breakdown of AES pipelining and the relative slowness of crossbar permutation instructions.

1 Introduction

In 2024, the U.S. NIST announced its plan [10] to standardize a wider variant of AES (FIPS 197 [9]), specifically, a variant of Rijndael [2] with a 256-bit block size and a single key size of 256 bits. Following NIST, we will refer to this block cipher as Rijndael-256 in this work.

The main motivation for Rijndael-256 adoption given in the announcement [10] lies in the observation that common block cipher modes of operation impose relatively low upper limits for the amount of data that can be securely processed with a 128-bit block cipher, such as AES [5]. For example, Galois/Counter Mode (GCM) [4] limits the size of its plaintext to $2^{39}-256$ bits (roughly 64 GiB.) Other standard modes of operation have various upper bounds, but they are all well under 2^{64} blocks of data.

> *Overcoming this per-key limit requires either a "beyond birthday bound" mode of operation or a wider block cipher (e.g., a block cipher with a block size of 256 bits).* [5]

1.1 Rijndael-256

One of the stated selection criteria for the Advanced Encryption Standard (in addition to cryptanalytic security, performance, IP issues, etc) was flexibility, including the ability *"to handle key and block sizes beyond the minimum that must be supported"* [6]. The original Rijndael proposal for AES allowed the block length and key length to be independently specified as 128, 192, or 256 bits. The algorithm was later defined for 160 and 224 as well [1].

However, upon the publication of FIPS 197 in 2001, the Advanced Encryption Standard, the block size was limited to 128 bits only [7]. The 2023 update [9] didn't contain technical changes. The larger-block variants have been mostly ignored by the research and engineering community for over 20 years.

A reference implementation of Rijndael, supporting all key and block sizes, as well as some test vectors, is available in the appendices of the book "The Design of Rijndael" by its designers, Joan Daemen and Vincent Rijmen [2].

Key Expansion Rijndael-(256,256) has 14 rounds and 15 round keys (same as with AES-256). Each round key is 32 bytes, double the size of AES. The key schedule of Rijndael-256 is the same as that of AES-256, except that two consecutive AES-256 round keys comprise one Rijndael-256 round key, and twice as many bytes are required overall. In software, the implementor needs to increase the loop count and make more round constants available in the routine to generate the second half of the expanded key. However, the limitations of the "key expansion assist" instructions complicate this process.

Round Function The Rijndael-256 state is organized into eight columns, each 4 bytes in length. The ShiftRows() constants are $\{0, 1, 3, 4\}$ – See Fig. 1. Other round function steps – SubBytes(), MixColumns(), AddRoundKey() – are just as in other variants of Rijndael, except wider; there are 32 parallel S-boxes in SubBytes() and eight mixing function invocations in MixColumns().

Input Bytes

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

→

After ShiftRows()

0	4	8	12	16	20	24	28	no shift
5	9	13	17	21	25	29	1	↩ 1 position
14	18	22	26	30	2	6	10	↩ 3 positions
19	23	27	31	3	7	11	15	↩ 4 positions

Fig. 1. Rijndael-256 ShiftRows(), the only round function step that "crosses lanes" as data is exchanged between a 128-bit lane boundary. Bytes are serialized "column-first" into the state matrix in Rijndael; the input consists of 32 bytes $0, 1, 2, \ldots, 31$.

1.2 RISC-V Cryptographic Extensions 1.0

The current RISC-V specifications [12] contain two sets of ratified AES extensions, one that utilizes the 32- and 64-bitscalar register file ("scalar cryptography extension": Zkne - NIST Suite: AES Encryption) and a second one that utilizes vector register file ("vector cryptography extension": Zvkned - NIST Suite: Vector AES Block Cipher). Furthermore, the "constant-time" extensions Zkt (Data Independent Execution Latency) and Zvkt (Vector Data-Independent Execution Latency) expand the ISA contract to assert latency properties that prevent timing attacks.

While Rijndael-256 is also implementable with scalar cryptography instructions (in constant time), we will focus on the higher-performance vector extension in this report.

The Zvk Vector Cryptography extension [11] was produced by the RISC-V Cryptographic Extensions Task Group (CETG) over several years and was ratified in late 2023. As an official extension, its instructions have permanently allocated opcodes, and it has mainline toolchain (LLVM, GCC) and simulator (QEMU, Spike, others) support.

The functionality provided by Zvk mirrors that of the scalar cryptography extension Zk (ratified in 2021), as well as by SIMD ISAs from Intel, ARM, and other vendors. The focus is on symmetric key algorithms used for bulk data processing: The U.S. NIST Advanced Encryption Standard (AES [9]), Secure Hash Algorithm 2 (SHA2 [8], which has two main variants, SHA256 and SHA512), and their PRC standard counterparts, the block cipher SM4 [14], and hash function SM3 [13]. The Galois Counter Mode (GCM [4]) Authenticated Encryption mode is supported by finite field multiplication operations. A hardware Entropy Source [15] interface is provided by the Zkr extension to support true random number generation.

2 Implementing Rijndael-256 with Zvk 1.0 Instructions

As observed by Drucker and Gueron on the Intel ISA in [3], one can implement Rijndael-256 using AES round instructions by adding an appropriate 256-bit "byte shuffle" to each round. The 32 S-Boxes and other components of Rijndael-256 are implemented by running two 128-bit AES instructions in parallel. However, the details are quite different on RISC-V from x86-64 as the operation of the instructions is not the same.

An implementation and a testbench for Rijndael-256 using Zvk 1.0 instructions can be found at `https://github.com/mjosaarinen/rij256-rv`

The Rijndael code assumes hardware vector length `VLEN` ≥ 256. The repository also contains AES-256 code for comparison reasons, structured the same way as the Rijndael-256 implementation.

2.1 Key Expansion

As noted in Section 1.1, the first 7 of Rijndael-256's round keys have the same bytes as the first 14 round keys of AES-256 with the same 256-bit secret keys.

A slight hitch is that `vaeskf2.vi` – the AES-256 key schedule instruction – also includes the round constant addition and performs a table lookup from an index provided as an immediate value.

We add XOR into the key schedule process to undo the "built-in" round constant and insert a new one from the table (see function `rij256_exp_key()` in `rij256_intrin.c`).

Key schedule is not a time-sensitive operation, and even with this additional step, the operation is still very fast. Note that (unlike some other ISAs), there is no need to modify the expanded key schedule for decryption.

2.2 Round Function with the Byte Shuffle

The 32 S-Boxes and other components of Rijndael-256 are implemented by running two 128-bit AES instructions in parallel. The permutation (in one step) essentially undoes the ShiftRows of AES-256 (Section 1.1) and then does the ShiftRows operation of Rijndael-256.

On RVV, one can use `vrgather.vv` to perform a byte shuffle with a byte index. The "magical" byte shuffle for Rijndael-256 encryption is

```
{ 0,  17, 22, 23, 4,  5,  26, 27, 8,  9,  14, 31, 12, 13, 18, 19,
  16, 1,  6,  7,  20, 21, 10, 11, 24, 25, 30, 15, 28, 29, 2,  3 };
```

and for Rijndael-256 decryption:

```
{ 0,  1,  30, 31, 4,  5,  2,  19, 8,  9,  22, 23, 12, 29, 26, 27,
  16, 17, 14, 15, 20, 21, 18, 3,  24, 25, 6,  7,  28, 13, 10, 11 };
```

The element width of the byte shuffle (8) differs from the element width (32) of AES. Currently, the code requires `vsetvli` instructions around the `vrgather` to change the element width.

```
vsetvli     zero, a4, e8, m1, ta, ma
vrgather.vv v25, v24, v8
vsetivli    zero, 8, e32, m1, ta, ma
vaesem.vv   v25, v10
```

Future architectures may loosen this limitation; the AES instructions do not really "care" about the element width (as they actually operate on 128-bit quantities). This would reduce the instruction count significantly and speed up implementations somewhat.

2.3 Instruction Counts and Performance

The actual performance of Rijndael-256 will depend on the hardware microarchitecture implementation details. We don't currently have widely deployed RISC-V hardware with AES instructions, so we will only provide instruction counts from the SPIKE simulator in Fig. 2.

	AES-256	Rijndael-256	Ratio
Key Expansion	58	123	2.12 ×
Encrypt 1024 bytes	775	2059	2.66 ×
Decrypt 1024 bytes	775	2059	2.66 ×

Fig. 2. Instruction counts for AES-256 and Rijndael-256 using RISC-V Vector Cryptography Extension 1.0 and vector length 256. The same key expansion procedure can be used for both encryption and decryption. Encryption and decryption instruction counts are for 1024 bytes (64 or 32 blocks) in ECB mode (CTR mode performance would not be very different.)

We note that some existing RISC-V microarchitectures have a notably slow `vrgather` instruction, requiring 16 cycles (or more) to shuffle 32 bytes, while AES instructions can be much faster. Hence, the performance slowdown caused by Rijndael-256 may be significantly higher than the 2.66× indicated by Fig. 2; CPU architects have informally speculated that it may be 5× or more. However, taking Rijndael-256 into account during the design phase (by considering Rijndael-256 microbenchmarks, etc.) will alleviate the impact over time, especially if Rijndael-256 becomes quantitively significant in practice (due to use in most common protocols such as TLS, or in applications such as storage encryption.)

References

1. Daemen, J., Rijmen, V.: AES proposal: Rijndael. Submisison to the NIST AES Competition, Version 2.0 (September 1999), https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf
2. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography, Springer (2020). https://doi.org/10.1007/978-3-662-60769-5
3. Drucker, N., Gueron, S.: Software optimization of Rijndael for modern x86-64 platforms. In: Latifi, S. (ed.) ITNG 2022 19th International Conference on Information Technology-New Generations. pp. 147–153. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-97652-1_18
4. Dworkin, M.J.: Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Special Publication SP 800-38D, NIST (November 2007). https://doi.org/10.6028/NIST.SP.800-38D
5. Mouha, N., Dworkin, M.: Report on the block cipher modes of operation in the NIST SP 800-38 series. NIST Internal Report NIST IR 8459, NIST (September 2024). https://doi.org/10.6028/NIST.IR.8459
6. Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., Roback, E.: Report on the development of the advanced encryption standard (AES). J. Res. Natl. Inst. Stand. Technol. **106**(3), 511–577 (May-June 2001). https://doi.org/10.6028/jres.106.023, https://nvlpubs.nist.gov/nistpubs/jres/106/3/j63nec.pdf

7. NIST: Advanced encryption standard (AES). Federal Information Processing Standards Publication FIPS 197, NIST (November 2001), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
8. NIST: Secure hash standard (SHS). Federal Information Processing Standards Publication FIPS 184-4, NIST (August 2015). https://doi.org/10.6028/NIST.FIPS.180-4
9. NIST: Advanced encryption standard (AES). Federal Information Processing Standards Publication FIPS 197 Update 1, NIST (May 2023). https://doi.org/10.6028/NIST.FIPS.197-upd1, minor update to original published in 2001
10. NIST: NIST proposes to standardize a wider variant of AES. PRE-DRAFT Call for Comments: FIPS 197 (Initial Preliminary Draft), NIST (December 2024), https://csrc.nist.gov/pubs/sp/800/197/iprd
11. RISC-V: RISC-V cryptography extensions volume II, vector instructions. Tech. rep., RISC-V Cryptographic Extensions Task Group (2023), https://github.com/riscv/riscv-crypto/releases/download/v1.0.0/riscv-crypto-spec-vector.pdf, ratified version v1.0.0
12. RISC-V: The RISC-V instruction set manual volume I: Unprivileged architecture. Ratified ISA Release 20240411, RISC-V International (April 2024), https://github.com/riscv/riscv-isa-manual/releases/download/20240411/unpriv-isa-asciidoc.pdf
13. SAC: Information security technology – SM3 cryptographic hash algorithm. GB/T 32905-2016, Standardization Administration of China (2016), http://www.gmbz.org.cn/upload/2018-07-24/1532401392982079739.pdf
14. SAC: Information security technology – SM4 block cipher algorithm. GB/T 32907-2016, Standardization Administration of China (2016), http://www.gmbz.org.cn/upload/2018-04-04/1522788048733065051.pdf
15. Turan, M.S., Barker, E., Kelsey, J., McKay, K.A., Baish, M.L., Boyle, M.: Recommendation for the entropy sources used for random bit generation. Special Publication SP 800-90B, NIST (January 2018). https://doi.org/10.6028/NIST.SP.800-90B