Efficient and Side-Channel Resistant Design of High-Security Ed448 on ARM Cortex-M4

Mila Anastasova, Mojtaba Bisheh-Niasar, Hwajeong Seo, Reza Azarderakhsh, *Member*, *IEEE*, and Mehran Mozaffari Kermani, *Senior Member*, *IEEE*

***Abstract*—The demand for classical cryptography schemes continues to increase due to the exhaustive studies on their security. Thus, constant improvement of timing, power consumption, and memory requirements are needed for the most widely used classical Elliptic Curve Cryptography (ECC) primitives, suiting high- as well as low-end devices. In this work, we present the first implementation of the Edwards Curve Digital Signature Algorithm (EdDSA) based on the Ed448 targeting the ARM Cortex-M4-based STM32F407VG microcontroller, which forms a large part of the Internet of Things (IoT) world. We report timing and memory consumption results based on portable C and target-specific hand-crafted assembly code implementations of the low-level finite filed arithmetics. We optimize the high-level group operations by implementing the efficient scalar multiplication over the Ed448 isogenous map to reduce the computation complexity. Furthermore, we provide a side-channel analysis (SCA) and fault attack protected design by developing point randomization, scalar blinding techniques, and repeated signature, and evaluate the performance. Our optimized architecture performs a signature and verification in 39.88ms and 51.54ms, respectively, where SCA protection can be achieved at less than 6.4% cost of performance overhead.**

***Index Terms*—ARM Cortex-M4, digital signature algorithm, Ed448, elliptic curve cryptography, side-channel**

I. Introduction

Classical crypto schemes such as RSA [1] and Elliptic Curve Cryptography (ECC) [2] form the main protection behind widely used applications and are the base of the most famous cryptographic libraries. The ECC family of schemes, based on elliptic curve point manipulations, provides faster computations and smaller key sizes, ensuring the same security levels, compared to RSA, therefore, is the most commonly used cryptographic scheme. The advantages of ECC and the increasing interest in deploying it lead to continuous work on optimizing its performance results and resource requirements.

The use of Edwards curves was first proposed by *Bernstein et al.* in [3] where the name of the new signature algorithm comes from - Edwards curve Digital Signature Algorithm (EdDSA). The authors suggest the use of Ed25519, ensuring a 128-bit level of security. The improved performance results and the optimized memory use, when applying Edwards curves, leads to continuing the work in the field, and in 2015 *Hamburg* proposed a higher security level EdDSA, based on the curve Ed448-Goldilocks [4]. Ed448 provides 2^{224} security target, rather than the 2^{128} security, provided by most of the previously used curves such as the NIST P256 and Curve25519, ensuring only AES-128 equivalent bit security level. The work on Ed448 is, to the best of our knowledge, limited to software-based design [4], where *Hamburg* proposes an efficient implementation of the arithmetic operations providing portable C/C++ code and hardware-based design [5], where *Bisheh-Niasar et al.* show area-time efficient results on Xilinx Virtex 7 platform.

The low-level arithmetics performed for the key exchange protocol X448 overlap with the Ed448 routines. Targeting ARMv7-M architecture, *Seo et al.* [6] proposed the first implementation of X448 on the Cortex-M4 platform, reaching point multiplication performance results of 6,218,135 clock cycles, offering constant time and cache protected design. Other Internet of Things (IoT) target platforms have been also used for deploying key exchange based on Curve448, particularly the work presented by *Seo et al.* [7] where 8-bit AVR ATmega and 16-bit MSP430 low-end processors are targeted. Recently, a number of hardware implementations have been introduced to implement a point multiplication core over Curve448 [8], Curve25519 [9], and Ed25519 digital signature algorithm [10].

The similar finite field bit length of Curve448 and the security level I of the (compressed) Supersingular Isogeny Key Encapsulation mechanism [11] leads to similar implementation techniques of the lowest level arithmetic operations. Several works [12], [13], [14], [15], [16] show performance, power consumption, and memory use software-based optimization strategies, targeting the IoT 32-bit ARM Cortex-M4 processors, and hardware-based improvements targeting Xilinx Virtex 7 platform [17], [18]. The literature also shows multiple research lines focusing on the optimal, side-channel, and fault attack secure lower security level Curve25519 and Ed25519 protocols on the Cortex-M4 embedded device [19], [20], [21] or other target processors [22], [23], [24]. However, to the best of our knowledge, there has not been any work on Ed448 for ARMv7-M architecture. In this work, we propose the first implementation of the Ed448 DSA on the target architecture and report our optimized performance results.

The paper is organized as follows: Section II reviews the preliminaries and the target platform features. The finite

M. Anastasova, Mojtaba Bisheh-Niasar and R. Azarderakhsh are with the Computer and Electrical Engineering and Computer Science Department and I-SENSE at Florida Atlantic University, Boca Raton, FL. Email addresses: (manastasova2017, mbishehniasa2019, razarderakhsh)@fau.edu.

H. Seo is with the Division of IT Convergence Engineering, Hansung University, Seoul 136-792, Korea. Email address: hwajeong84@gmail.com.

R. Azarderakhsh is with PQSecure Technologies, LLC, Boca Raton, FL, USA.

M. Mozaffari Kermani is with the Computer Engineering and Science Department at University of South Florida, Tampa FL. Email address: mehran2@usf.edu.

Public Parameters: $p = 2^{448} - 2^{224} - 1$, $E/\mathbb{F}_p = ax2 + y2 = 1 + dx2y2$, G

Alice		**Bob**
Key Generation **Input:** $seed$ **Output:** sk_A, pk_A		
1. $sk_A \in_R^{seed} \mathbb{Z}/\mathbb{F}_p$ 2. $(p, s) \leftarrow H(sk_A)$ 3. $pk_A \leftarrow [s] \cdot G$		**Verifying** **Input:** $pk_A, M, sign$ **Output:** $true/fasle$
Signing **Input:** p, sk_A, M **Output:** $sign \equiv R\|\|S$	$sign$ $\longrightarrow$	1. $k \leftarrow H(R\|\|pk_A\|\|M)$ **Return** $[S] \cdot G == R + [k] \cdot pk_A$
1. $r \leftarrow (H(p\|\|M))(modL)$ 2. $R \leftarrow [r] \cdot G$ 3. $k \leftarrow (H(R\|\|pk_A\|\|M))(modL)$ 4. $S \leftarrow (r + k * s)(modL)$ **Return** $R\|\|S$		

Figure 1. Ed448 algorithm [25]. H denotes $SHAKE256$. L represents the order of Ed448 curve. G represents the value of the base point.

field arithmetics and side-channel countermeasures are briefly presented in Section III and Section IV. Section V presents evaluation and analysis of the performance results obtained. Finally, we conclude this work in Section VI.

II. Mathematical Background and Target Architecture

This section describes the EdDSA signature protocol and reviews the characteristics of the target platform.

A. Preliminaries

The Edwards-Curve Digital Signature Algorithm (EdDSA) is defined over Ed448 in [25], where the points satisfying the equation $Ed/\mathbb{F}_p : ax^2 + y^2 = 1 + dx^2y^2$ are laying on the twisted Edwards curve over a finite field, defined as $\mathbb{F}_p$with $p = 2^{448} - 2^{224} - 1$ and $d = -39081$, where $P = (x, y)$ and $x, y \in \mathbb{F}_p$. An Edwards curve Ed448 is birationally equivalent to a Montgomery curve. The elliptic curve-based protocols have a pyramid-like structure where the computational layers consist of finite field arithmetics, manipulating long integers, and group operation applied on the curve elements. For the efficient and side-channel secure design of the ECC protocols, including the EdDSA algorithm, multiple optimization strategies and countermeasures should be added to the operation layers of the scheme.

The group operation, base of ECC, is the scalar multiplication, where it can be implemented in different ways such as Double-And-Add (and its side-channel protected variants). The Montgomery Ladder technique over Montgomery curves, however, is the most efficient scalar multiplication algorithm, computing a point addition and a point doubling in a single ladder step. This algorithm requires $4\boldsymbol{M} + 4\boldsymbol{S} + 1\boldsymbol{k} + 8\boldsymbol{A}$ with $\boldsymbol{M}$, $\boldsymbol{S}$, $\boldsymbol{k}$, and $\boldsymbol{A}$ are the multiplication, squaring, multiplying by a constant, and addition cost, respectively. Furthermore, the group operations are performed on the projective coordinates (X, Y, Z), while the map to affine space (x, y) is defined as $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$.

Graphical representation of the Ed448 DSA protocol, executed by both computational parties, is shown in Figure 1.

Table I
ARMv7-M ISA [26] for memory access and MAC instructions.

Instruction	**Functionality**	**Timing (CC)**
LDR/STR	$R_n \leftarrow$memory memory$\leftarrow R_n$	2
UMULL	$Rd_1, Rd_2 \leftarrow R_n \times R_m$	1
UMAAL	$Rd_1, Rd_2 \leftarrow R_n \times R_m + Rd_1 + Rd_2$	1

The key generation procedure receives a seed value and outputs a key pair sk_A and pk_A. The signing function receives the value of the secret key sk_A along with a message value M and returns the signature, composed by R and S. Finally, the verification subroutine receives the public key pk_A, the message M, and the signature of the message, to verify the authenticity of the signing party.

B. ARMv7-M Architecture

The simple and highly efficient ARMv7-M architecture, implementing Reduced Instruction Set Computer (RISC), converts the Cortex-M4 into the most widely deployed platform. To evaluate and analyze our implementation performance, we use the Cortex-M4-based STM32F407VG microcontroller, recommended by NIST as a target platform for benchmarking the PQ primitives on low-end devices, which features 192KB of RAM and another 1MB of flash memory.

The target platform offers performance free of structural hazards and data dependencies, where the only exception, resulting in an additional clock cycle, is the use of memory access instructions. Thus, loaded data should be kept in the 16 32-bit General Purpose Registers (GPRs) – `R0-R15` as long as possible.

The ARMv7-M ISA includes the low-latency Multiply ACcumulate (MAC) instructions, which allow the execution of 32×32-bit multiplication, possibly along with another two additions, requiring a single clock cycle. Table I details their functionality and latency.

III. Finite Field Arithmetic

For the optimal performance of the Ed448 protocol, as well as the minimized memory usage, we have focused on implementing the low-level arithmetic operations, where the pyramid-like structure of the ECC cryptography ensures that the optimizations of each particular layer result in overall execution improvements.

Modular Addition: For the implementation of the modular addition, we have carefully scheduled the order of the memory access instructions so that consecutive memory addresses are accessed without interrupting the loading sequence of instructions. Our proposed design ensures that the additional clock cycle, the stall, is absorbed by the following instruction due to the 3-stage pipeline of the Cortex-M4 processor. Therefore, accessing n consecutive $32-$bit data segments, requires $n+1$ cycles when properly scheduled.

Multi-precision Multiplication: The invocation rate of the multi-precision multiplication routine leads to the need for optimal design. The long integers required for the storage of the

Table II
FINITE FIELD OPERATIONS FOR CURVE448/ED448 TARGETING ARMV7-M.

Ref.	Timing [cc]						
	Curve448/Ed448 operations						
	$\mathbb{F}_p$				Group		
	Add	Sub	Multiply	Invert	Add	Double	Multiply
Seo et al. [6]	164	161	821	363,485	6,566	6,567	6,218,135
This work [C]	337	350	2,962	1,369,7543	34,075(total)		15,200,3398
This work [ASM]	139	137	705	325,997	8,465(total)		3,703,755
Ref.	**Ed448 Specific operations**						
	Mod l	y-Recovery	Shake256	Decode	Point Multiplication		
This work [C]	745,082	41,119	13,966	2,681,880	12,558,965		
This work [ASM]	-	10,581	-	643,611	3,503,308		

operands, however, convert it into a challenging task, since it suggests more complex and time-consuming implementation.

We propose the Refined-Operand Caching (R-OC) technique illustrated in Figure 2, where the size of the rows, i.e. the number of 32-bit words cached in the register set and operated on, is increased from 3 to 4 by increasing the register utilization. According to Figure 2, different strategies are implemented for the beginning (marked with light grey color), the middle, and the end (marked in blue color) of each row. Due to the use of `UMULL` and `UMALL` instructions (marked with black and white dots/rectangles, respectively), the zero initialization of the registers is omitted. Finally, careful scheduling of the register use has been performed, eliminating data dependencies always when possible.

Reduction: A modular reduction is required to bring back the values of the arithmetic operations in the finite field. The special shape of the prime p allows the implementation of a highly optimized reduction technique, proposed in [7]. We also propose an efficient Montgomery reduction for reducing the SHAKE256 output with respect to the group order. To reduce the double-length multiplication, the value is split into chunks, where continuous addition with careful propagation of the carry bit returns the reduced value.

Table II details the results obtained after running C and assembly ARMv7-M language implementations. Our design is more than 100 CC faster for the $\mathbb{F}_p$ multiplication, $1.5\times$ more efficient for point addition and point doubling, and almost $2\times$ better for point multiplication compared to the previous best work targeting Curve448 [7]. Additionally, we report the latency of the Ed448 specific functions.

IV. SIDE-CHANNEL AND FAULT ATTACK CONSIDERATIONS

The side-channel analysis (SCA) attack uses data leakage based on timing, power consumption, or electromagnetism information. Therefore, our design shows constant-time execution, preventing simple power analysis (SPA) attacks. To protect against differential power analysis (DPA) attacks, we have deployed additional countermeasures.

Scalar Blinding: We apply scalar blinding, which *hides* the value of the secret scalar by removing the point swapping data dependency during point multiplication. We randomly generate

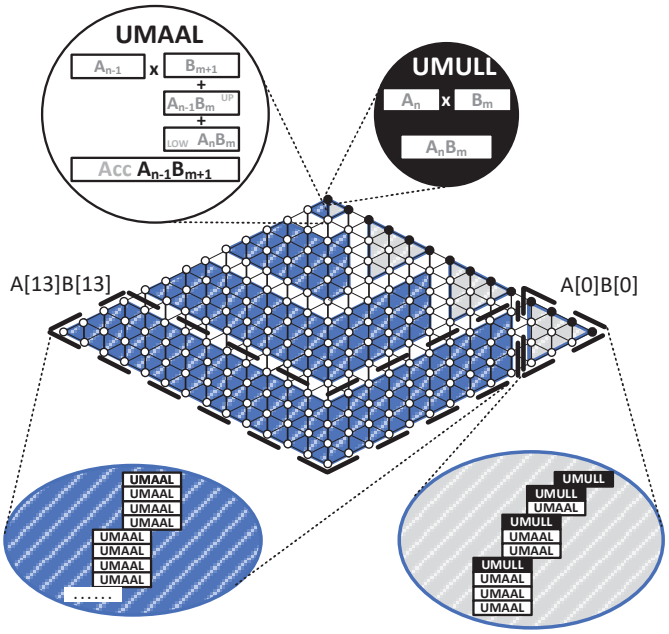

Figure 2. Rhombus representation of the multiplication strategy.

r and multiply it by the group order, such that $k \leftarrow sk_A + r \cdot \#Ed$, where due to the associative property, it will result in the point at Infinity and does not modify our result, however, comes at the cost of an additional multiplication.

Point randomization: Another DPA countermeasure technique that we have integrated into our design is point randomization. In particular, we use an integer λ and obtain the randomized base point $G_{rand} = (\lambda \cdot x, \lambda)$ and $G_{rand} = (\lambda \cdot X, \lambda \cdot Y, \lambda)$ in affine and projective coordinates, respectively. Upon conversion of the resulting point coordinates back to affine space, the value of λ is discarded as $x = \frac{\lambda X}{\lambda Z} = \frac{X}{Z}$. Thus, our resulting point $(x, y) = sk_A \cdot G_{rand} = sk_A \cdot G$ does not change.

Fault attacks are based on modifications in the environment such as the voltage supply or the frequency of the device's clock. They can also be a result of maliciously modified information (e.g. parameter values), where the erroneous output serves to obtain the private data.

Repeated Sign: To protect our EdDSA implementation design, we apply the repeated execution of the signing procedure, which, considering the EdDSA deterministic property, ensures that the signature is not faulty. Some literature sources suggest checking the output of the hash function only, however, it will not ensure protection when the input value is attacked, thus, we do not consider it in our analysis.

V. PERFORMANCE EVALUATION

Table Table IV reports the performance results of, to the best of our knowledge, the first implementation of Ed448 DSA on the Cortex-M4 target processor, using the STM32F407VG microcontroller. The target platform offers operation mode @24MHz and @168MHz, where the former shows precise latency results ensuring zero wait state, eliminating MCU stalls, and the latter provides a real-world scenario measurement.

Table III
ED25519/ED448 DSA PERFORMANCE ON IOT PLATFORMS.

Work	Platform	Freq. [MHz]	KeyGen [KCCs]	KeyGen [ms]	Sign [KCCs]	Sign [ms]	Verify [KCCs]	Verify [ms]
Ed25519 [20]	Cortex-M4	84	389.5	4.64	543.7	6.47	1,331.4	15.85
Ed448 [6]	AVR	32	103,229	3,225.9	-	-	-	-
Ed448 [6]	MSP	25	73,478	2,939.1	-	-	-	-
This work [C]	Cortex-M4	24	11,326	471.91	13,828	576.16	22,062	919.25
		168	11,694	69.60	14,198	84.51	22,730	135.29
This work [ASM]	Cortex-M4	24	4,069	169.54	6,571	273.79	8,452	352.16
		168	4,195	**24.97**	6,699	**39.87**	8,659	**51.54**

Table IV
ED448 DSA SIDE-CHANNEL PROTECTED DESIGN PERFORMANCE ON STM32F407VG.

Scheme	Timing [ms]			Memory
	KeyGen	Sign	Verify	[B]
No SCA	24.97	39.88	51.54	3,612
Scalar Blinding	25.42	40.33	51.54	3,612
Point Randomization	27.19	42.10	51.54	3,612
Both Countermeasures	27.69	42.60	51.54	3,612

We mark the related work on Ed25519 and Ed448 along with the target platforms in Table III. Aiming at optimizing the design for Cortex-M4, our key generation, sign and verify functions complete in 24.97ms, 39.87ms, and 51.54ms, respectively, when running @168MHz.

Table IV reports the additional latency when deploying SCA countermeasures. We note that when applying point randomization and scalar blinding, the added timing is around 3ms for both - the key generation and the signing function. We report the constant memory use in bytes.

VI. CONCLUSIONS

In this work, we presented the first implementation of the Ed448 DSA protocol targeting the highly demanded low-end device ARM-based Cortex-M4. We evaluate the performance results based on pure C code implementation design and target-specific assembly language. Finally, we provide side-channel and fault attack protected design and report the performance obtained.

We keep evaluating our proposed countermeasures using the TVLA technique over Cortex-M4 as future work. We also would like to report our design's energy and power consumption in different scenarios.

VII. ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their comments. This work is supported in parts by NSF-1801341.

REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*. Springer, 1985, pp. 417–426.

[3] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of cryptographic engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[4] M. Hamburg, "Ed448-Goldilocks, a new elliptic curve," *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 625, 2015.

[5] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Area-time efficient hardware architecture for signature based on Ed448," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021.

[6] H. Seo and R. Azarderakhsh, "Curve448 on 32-Bit ARM Cortex-M4," in *International Conference on Information Security and Cryptology*. Springer, 2020, pp. 125–139.

[7] H. Seo, "Compact implementations of Curve Ed448 on low-end IoT platforms," *ETRI Journal*, vol. 41, no. 6, pp. 863–872, 2019.

[8] M. Bisheh Niasar, R. Azarderakhsh, and M. Mozaffari kermani, "Efficient hardware implementations for elliptic curve cryptography over Curve448," in *21st International Conference on Cryptology, Indocrypt 2020, India, December 13-16, 2020*, 2020.

[9] M. Bisheh Niasar, R. E. Khatib, R. Azarderakhsh, and M. Mozaffari Kermani, "Fast, small, and area-time efficient architectures for key-exchange on Curve25519," in *27th IEEE Symposium on Computer Arithmetic, ARITH 2020, Portland, OR, USA, June 7-10, 2020*, 2020, pp. 72–79.

[10] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Cryptographic accelerators for digital signature based on Ed25519," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 29, no. 7, pp. 1297–1305, 2021.

[11] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, J. Renes, V. Soukharev, and D. Urbanik, "Supersingular Isogeny Key Encapsulation," Submission to the NIST Post-Quantum Standardization Project, 2017. [Online]. Available: https://sike.org/

[12] H. Seo, Z. Liu, P. Longa, and Z. Hu, "SIDH on ARM: faster modular multiplications for faster post-quantum supersingular isogeny key exchange," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 1–20, 2018.

[13] H. Seo, "Memory efficient implementation of modular multiplication for 32-bit ARM Cortex-M4," *Applied Sciences*, vol. 10, no. 4, p. 1539, 2020.

[14] H. Seo, M. Anastasova, A. Jalali, and R. Azarderakhsh, "Supersingular Isogeny Key Encapsulation (SIKE) Round 2 on ARM Cortex-M4," *IEEE Transactions on Computers*, 2020.

[15] M. Anastasova, R. Azarderakhsh, and M. M. Kermani, "Fast strategies for the implementation of sike round 3 on arm cortex-m4," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 10, pp. 4129–4141, 2021.

[16] M. Anastasova, M. Bisheh-Niasar, R. Azarderakhsh, and M. M. Kermani, "Compressed SIKE Round 3 on ARM Cortex-M4," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2021, pp. 441–457.

[17] R. Elkhatib, R. Azarderakhsh, and M. Mozaffari-Kermani, "Accelerated RISC-V for post-quantum SIKE," *Cryptology ePrint Archive*, 2021.

[18] Elkhatib, Rami and Azarderakhsh, Reza and Mozaffari-Kermani, Mehran, "High-Performance FPGA Accelerator for SIKE," *IEEE Transactions on Computers*, 2021.

[19] F. De Santis and G. Sigl, "Towards side-channel protected X25519 on ARM Cortex-M4 processors," *Proceedings of Software performance enhancement for encryption and decryption, and benchmarking, Utrecht, The Netherlands*, pp. 19–21, 2016.

[20] H. Fujii and D. F. Aranha, "Curve25519 for the Cortex-M4 and beyond," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2017, pp. 109–127.

[21] J. Xu, M. Li, L. Liang, Y. Zhang, S. Xiang, and Z. Ma, "Profiling attacks against ecc: Side channel analysis based on deep learning for curve-25519," in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*. IEEE, 2021, pp. 274–278.

[22] Y. Romailler and S. Pelissier, "Practical fault attack against the Ed25519 and EdDSA signature schemes," in *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2017, pp. 17–24.

[23] K. Chalkias, F. Garillot, and V. Nikolaenko, "Taming the many EdDSAs," in *International Conference on Research in Security Standardisation*. Springer, 2020, pp. 67–90.

[24] N. Samwel and L. Batina, "Practical fault injection on deterministic signatures: the case of EdDSA," in *International Conference on Cryptology in Africa*. Springer, 2018, pp. 306–321.

[25] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," RFC 8032, Jan. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8032.txt

[26] ARM., "Cortex-M4 ISA." last accessed on Jan 18, 2022 from https://developer.arm.com/documentation/ 100166/0001.