Efficient and Side-Channel Resistant Ed25519 on ARM Cortex-M4

Daniel Owens, Rabih El Khatib, Mojtaba Bisheh-Niasar, Reza Azarderakhsh, *Member, IEEE*
and Mehran Mozaffari Kermani, *Senior Member, IEEE*

***Abstract*—As the cryptographic community turns its focus toward post-quantum cryptography, the demand for classical cryptographic schemes such as Elliptic Curve Cryptography (ECC) remains high. In this work, we present an optimized implementation of the Edwards Curve Digital Signature Algorithm (EdDSA) operations Keygen, Sign, and Verify using the Ed25519 parameter on the ARM Cortex-M4 using optimized assembly code. We discuss the optimization of field and group arithmetic to produce high-throughput cryptographic primitives. Then, we present the first SCA-resistant implementation of the Signed Comb method, and Test Vector Leakage Assessment (TVLA) measurements. Our fastest implementation performs Ed25519 Keygen in 200,000 cycles, Sign in 240,000 cycles, and Verify in 720,000 cycles on the ARM Cortex-M4.**

I. Introduction

It is estimated that by 2023, about 14.7 billion Internet of Things (IoT) devices will be connected to the Internet. The proliferation of IoT devices is creating a revolution in connectivity worldwide, while simultaneously creating difficult security and privacy problems [1].

Elliptic Curve Cryptography (ECC) is a popular public-key system [2] that is commonly used in IoT devices. Ed25519 is the Edwards Digital Signature Algorithm (EdDSA) configured to use the elliptic curve Edwards25519 and SHA-512 introduced in 2011 [3]. It was standardized in 2019 by the NIST in FIPS 186-5 [4] and is used to produce public keys, compute digital signatures of messages, and verify signature and message pairs. However, all ECC systems, including Ed25519, are vulnerable to Shor's algorithm [5] once a suitably powerful quantum computer is created.

In 2022 the National Institute of Standards and Technology (NIST) announced it had chosen the first group of winners from its six-year competition to find new quantum-resistant cryptographic algorithms [6]. Although NIST's first standards for post-quantum cryptography (PQC) algorithms are not expected until 2024 [7], the transition to PQC standards has already begun. During this phase, hybrid systems which combine classical and PQC will be needed to maintain regulations and standards for the duration of the transition period [8]. Hence, continued research that improves latency and reduces resource utilization for ECC in constrained devices is still relevant.

Side-channel attacks. IoT devices may be deployed in a manner that exposes them to physical access with few restrictions. As a result, a physical attack model should be considered when writing cryptographic software for these devices. There are two types of such attacks: passive and active. Active attacks are fault attacks, whereas passive attacks, performed via side-channel analysis (SCA), include power and timing attacks [9], among others. We focus on defense against passive attacks in this work.

Our contributions. Ed25519 has been thoroughly researched and discussed. However, there is little literature about the use of the Signed Comb method of scalar multiplication proposed by [10]. This work bests the prior work [11] with an up to 52% reduction in latency of the Ed25519 cryptographic primitives using the Signed Comb Method. For the first time, side-channel countermeasures are evaluated while using the Signed Comb scalar multiplication method.

Organization. This paper is organized as follows: In §2, we discuss the basics about Ed25519, the ECC operational structure, and introduce the target architecture. In §3 we describe implementation details about finite field arithmetic and scalar multiplication. Finally, in §4 we discuss side-channel countermeasures and present Test Vector Leakage Assessment (TVLA) results.

II. Preliminaries

A. Ed25519

The Edwards-Curve Digital Signature Algorithm (EdDSA) with parameter Ed25519 is defined in [12], where the points satisfying the equation $Ed/\mathbb{F}_p : ax^2 + y^2 = 1 + dx^2y^2$ lay on the twisted Edwards curve over a finite field, defined as $\mathbb{F}_p$ with $p = 2^{255} - 19$ and

$$d = 37095705934669439343138083508754565 18...$$
$$...95421138798032190163887855330859402835 55$$

where $P = (x, y)$ and $x, y \in \mathbb{F}_p$. A combination of finite field operations performed on values in the field $\mathbb{F}_p$ such as long integer addition, subtraction, and multiplication form the basis for group operations that are applied to elements on the curve that form the EdDSA cryptographic primitives Keygen, Sign, and Verify.

B. ARMv7-M Architecture

To evaluate and analyze our implementation's performance, we use the ARM Cortex-M4 based STM32F407VG micro controller, which is a reduced instruction set computer (RISC) [13]. It features 192 KB of RAM and 1MB of flash memory,

sixteen (thirteen usable) 32-bit General Purpose Registers (GPRs), and a further 32, 32-bit (or 16, 64-bit) Floating Point Registers (FPRs) intended for use with the platform's built-in Floating Point Unit (FPU).

Importantly, the ARMv7-M ISA includes the low-latency Multiply ACcumulate (MAC) instructions UMUL and UMAAL, which allow the execution of 32×32-bit multiplication (UMAAL adds two additions) needing only a single clock cycle each. These operations are crucial for efficient finite field arithmetic.

III. Implementation Details

A. Finite Field Arithmetic

High throughput, low latency operation of Ed25519 is only possible with efficient finite field operations. Open source code provided by the `X25519-Cortex-M4` project created by [14] was utilized in our work to perform modular addition and subtraction, as well as $\mathbb{F}_p$ multiplication. This project offers the fastest known $\mathbb{F}_p$ arithmetic routines for ARMv7-M architectures.

Modular Addition and Subtraction: Modular addition is implemented using the `ADC` and `ADS` instructions, needing only 1 cycle to add and propagate the carry. A weak reduction $\mathbb{F}_{2^{256}-38}$ is used to avoid the final carry propagation and defer the full $\mathbb{F}_p$ reduction until a multiplication is performed to save cycles. Modular addition is implemented in a similar manner with the instructions `SBC` and `SBS`.

Multi-precision Multiplication: An efficient multi-precision multiplication design is a necessity for a low-latency implementation. The approach utilized by `X25519-Cortex-M4`, illustrated in Figure 1, is similar to the Operand Caching techniques described by *Fuji et al.* [11] and *Anastasova et al.* [15] where the required operations are split into two rows and processed in-order. Execution proceeds from right to left, starting from `r0`, with horizontal connections indicating multiplication and vertical indicating addition.

Modular Inversion: While the `X25519-Cortex-M4` project includes code for inversion, we implemented the Itoh-Tsuji method proposed in [16] which uses 11 modular multiplications and 254 modular squares to compute $a^{p-2} \equiv a^{-1} \bmod p$ in 53% less cycles than [11].

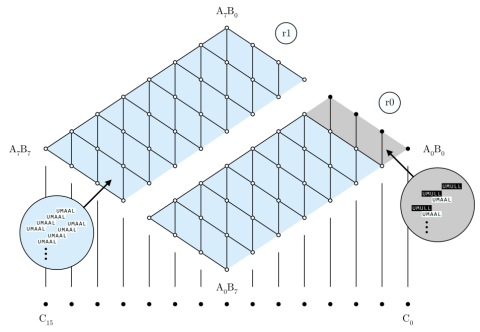

Fig. 1. Rhombus representation of the multiplication strategy.

B. The Signed Comb Method

Currently the most efficient constant time scalar multiplication algorithm, the Signed Comb method [10] is used for Ed25519 Keygen and Sign in our implementation. Three parameters are required: n the number of combs (with one comb per block), t the number of "teeth" in each comb, and s the number of bits between each teeth. The number of bits in a block is determined by $t \cdot s$. This work uses the parameters $n = 3$, $t = 5$, and $s = 17$, where $n \cdot t \cdot s = 255 = \log_2(p)$.

Signed Binary Conversion: Before performing the comb operations the scalar e must be converted to its signed binary form d using Equation 1 provided by [10]

$$d = \frac{e + 2^D - 1}{2} \tag{1}$$

where $d_i \in \{\pm 1\}$ and $D > \log_2(l)$. D can be set to the length of the input scalar e.

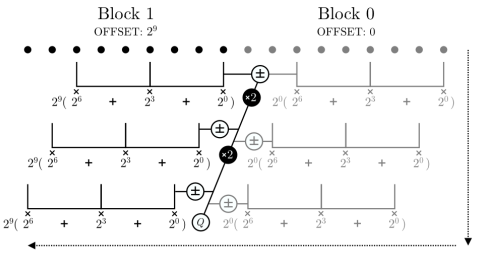

Fig. 2. Example of a comb operation where $n = 2$, $t = 3$, and $s = 3$

Signed Comb Precomputation: The Signed Comb method uses precomputed multiples of the Ed25519 base point G to significantly reduce the number of Point Additions and Point Doubles needed to perform the multiplication. This work precomputes the multiples offline and stores them as a table for use during operation.

To precompute the multiples, first observe that there are $j = n - 1$ combs and $k = t - 1$ teeth, and an offset o for each block j of the scalar d. Then for each tooth k of the jth comb compute $C_{j,k}$, from [10] as

$$C_{j,k} = \sum_{i=0}^{t_j - 1} d_{o_j + s_j i + k} \cdot 2^{s_j i} \tag{2}$$

Finally, compute the $2^{t_j} - 1$ values of $|C_{j,k}| \cdot G$ for each $j \in [0, n)$ [10]. Because d is in signed binary form, a leading coefficient of 1 indicates a negative value. Hence, we can reduce the necessary number of values from $2^{t_j} - 1$ to $2^{t_j - 1} - 1$ by only keeping the positive multiples. During multiplication, if a negative multiple is needed, the inverse of a positive multiple is computed where the inverse is defined as $(-x, y)$. Our comb parameters $n = 3$, $t = 5$, and $s = 17$ require 48 precomputed multiples using 4.5 KB of ROM.

Signed Comb Computation: The computation stage consists of a small amount of point doubles and adds, which vary based on the n, t, s parameters chosen. After multiples of G have been precomputed, for each comb $j \in [0, n)$, double, then add or subtract the appropriate multiple. Let $Q = G$. The

TABLE I
COMPARISON OF 255-BIT SCALAR MULTIPLICATION ALGORITHM COSTS IN TERMS OF POINT ADDITIONS (PA) AND POINT DOUBLES (PD)

Algorithm	Cost in PA/PD
Double-and-Add	255PD+128PA
Montgomery Ladder	255PD/PA
Window (average)	255PD+$((1-2^{-w})255/w)$PA
w-NAF method (average)	255PD+(255/(w + 1))PA
Signed Comb	$(s_{\max}-1)$PD+$(\sum_{j=0}^{n-1} s_j)$PA
Signed Comb where $n=3$, $t=5$, $s=17$	16PD+51PA

computation proceeds as follows in Equation 3, demonstrated in [10]

$$Q_{\text{Final}} = \sum_{k=s-1}^{0} 2Q + \sum_{j=0}^{n-1} Q \pm G_{j,k} \pmod{l} \tag{3}$$

where $2Q$ is point doubling and $\pm$ is point addition between the intermediate point Q and the possibly inverted multiple $G_{j,k}$. Point inversion is always performed, and $G_{j,k}$ or $-G_{j,k}$ is chosen using a masking method. For our parameters, scalar multiplication requires $16\boldsymbol{PD} + 51\boldsymbol{PA}$. Point addition and point doubling are implemented as demonstrated in [12]. For a comparison with other techniques, see Table I.

C. w-NAF

Ed25519 Verify operates only on public information. For this reason the non-constant time w-NAF method was utilized for scalar multiplication, based on the proposal by [17]. To compute the double point multiplication of two points, the base point B is multiplied by the S portion of the signature, and the point A decompressed from the public key multiplied by the hash $H(R, A, M)$. To decompress the public key, we use the fast decompression method proposed in [3] which does not involve an inversion and instead only requires a single exponentiation. To verify the signature, we use the fast single-signature verification method which requires checking if $SB - H(R, A, M)A$ (computed through double point multiplication) and R are the same in affine coordinates, which requires an inversion. [3]

IV. SIDE-CHANNEL ANALYSIS

A. Side Channel Countermeasures

Two popular SCA countermeasures for ECC systems are Scalar Blinding and Point Randomization.

Scalar Blinding: The scalar s is blinded by adding it to a multiple of the group order l by some random integer r, computing $s' = s + r \cdot l$, such that the final result $s' \cdot P = s \cdot P$. In our final implementation we chose to use a 129-bit random value for r which increases the bit length of the scalar from 255 bits to 384. In addition, to accommodate the longer scalar, our comb parameters were changed to $(n = 4, t = 4, s = 24)$ which requires 32 precomputed multiples using 3 KB of ROM.

Point Randomization: The base point G is replaced with G' by selecting a random integer λ, then multiplying G's coefficients by λ. Point Randomization is costly, and introduced leakage when used with the Signed Comb method, therefore it is not a part of our design.

B. Test Vector Leakage Assessment

Test Vector Leakage Assessment (TVLA) is a method to robustly and efficiently detect leakage a passive attacker may be able to observe during cryptographic operations [18]. Using the plot provided in [19], a value of 7 was chosen as our t-threshold, represented as blue lines. For evaluation, non-specific fixed vs. random t tests were performed as presented in [18].

C. TVLA Measurements

While performing side-channel analysis, the SHA-512 was excluded from Keygen. In Sign, the first hash of the secret key was excluded. SHA-512 is known to be insecure as demonstrated by [20] and we consider it outside the scope of this work. For development and benchmarking, an open source version of SHA-512 was used as provided by [21].

Experimental Setup: TVLA experiments were performed with the following equipment and configuration:

- A host PC that sends test vectors to the DUT
- A Picotech PicoScope 3000 series, 200 MHz bandwidth and 8-bit sample resolution.
- The NewAE ChipWhisperer lite board [22]
- The NewAE CW308T-STM32F ARM Cortex-M4 target board, mounted on the NewAE CW308 UFO board [22]

The DUT was run at 25 MHz for all experiments. The power traces were obtained via a passive probe connected to the CW308 UFO board at a rate of 125 MS/s, 5 samples per DUT clock cycle.

Baseline TVLA Measurements: A large amount of leakage over the t threshold was observed in the absence of countermeasures after only 500 TVLA traces were processed in Figure 3.

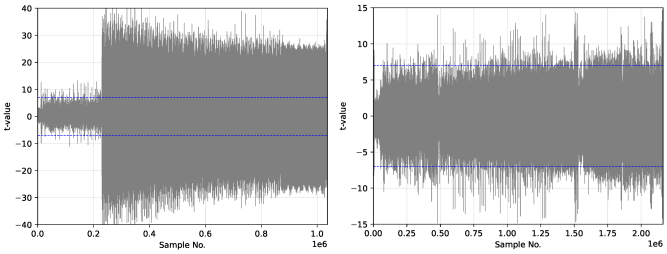

Fig. 3. Left: Unprotected Keygen using the Signed-Comb method (parameters 3,5,17) after performing TVLA over 500 traces. Right: Unprotected Sign using the Signed-Comb method (parameters 3,5,17) after performing TVLA over 500 traces.

TVLA Measurements with Countermeasures: Under the same set of experimental criteria, TVLA was applied to the implementation of Keygen with only Scalar Blinding over

10,000 traces which can be observed in Figure 4. Only 500 traces where recorded for Sign with Scalar Blinding, as seen in Figure 4. Even though the initial hash was removed, the remaining hash functions still leak significant information after only a small amount of traces.

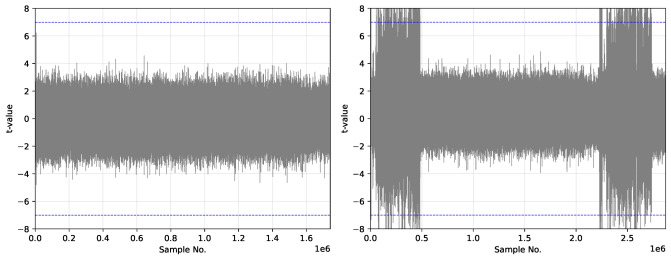

Fig. 4. Left: Protected Keygen using the Signed-Comb method (parameters 4,4,24) with Scalar Blinding after performing TVLA over 10,000 traces. Right: Protected Sign using the Signed-Comb method (parameters 4,4,24) with Scalar Blinding after performing TVLA over 500 traces. SHA-512 is responsible for the leakage present.

V. Performance & Conclusions

See Table II for a comparison of our work compared to the previous best work. All code was compiled with `arm-none-eabi-gcc` version 10.2.1, using the `-O3` optimization flag.

TABLE II
ED25519 DSA PERFORMANCE ON IOT PLATFORMS.
KCC IS KILO-CLOCK CYCLES.

Work	Platform	Freq. [MHz]	Keygen [KCCs]	Sign [KCCs]	Verify [KCCs]
Ed25519 (No SCA) [11]	Cortex-M4	48	347	496	1265
This work	Cortex-M4	24	200	239	722
		168	214	254	760
This work (SCA 129-bit Scalar Blinding)	Cortex-M4	24	325	364	-
		168	344	385	-

In this work, we presented an efficient and side-channel secure implementation of Ed25519 on the ARM Cortex-M4. Our implementation was shown to outperform existing works in latency when performing Keygen, Sign, and Verify due to highly optimized target-specific assembly code. We also presented side-channel analysis results of Keygen and Sign implemented with the Signed Comb Method for the first time. Performance was reported for the design with and without SCA countermeasures.

VI. Acknowledgments

The authors would like to thank the reviewers for their comments. This work is supported in parts by NSF 2147196 grant.

References

[1] E. Ronen and A. Shamir, "Extended functionality attacks on IoT devices: The case of smart lights," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 3–12.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, pp. 417–426.

[3] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," vol. 2, pp. 77–89.

[4] National Institute of Standards and Technology, "FIPS PUB 186-5 (Draft) - Digital Signature Standard (DSS)." [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5-draft.pdf

[5] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*. Ieee, pp. 124–134.

[6] "NIST Announces First Four Quantum-Resistant Cryptographic Algorithms." [Online]. Available: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms

[7] D. Moody, "NIST PQC: LOOKING INTO THE FUTURE." [Online]. Available: https://csrc.nist.gov/csrc/media/Presentations/2022/nist-pqc-looking-into-the-future/images-media/session-1-moody-looking-into-future-pqc2022.pdf

[8] N. Bindel, U. Herath, M. McKague, and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure," Cryptology ePrint Archive, Paper 2017/460. [Online]. Available: https://eprint.iacr.org/2017/460

[9] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology CRYPTO 1996*, N. Koblitz, Ed. Springer Berlin Heidelberg, vol. 1109, pp. 104–113.

[10] M. Hamburg, "Fast and compact elliptic-curve cryptography." [Online]. Available: https://eprint.iacr.org/2012/309

[11] H. Fujii and D. F. Aranha, "Curve25519 for the Cortex-M4 and Beyond," in *Progress in Cryptology – LATINCRYPT 2017*, T. Lange and O. Dunkelman, Eds. Springer International Publishing, pp. 109–127.

[12] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," p. RFC8032. [Online]. Available: https://www.rfc-editor.org/info/rfc8032

[13] ARM, "ARM® Cortex®-M4 Processor Technical Reference Manual Revision: R0p1." [Online]. Available: https://developer.arm.com/documentation/100166/0001

[14] Emill, "X25519-Cortex-M4." [Online]. Available: https://github.com/Emill/X25519-Cortex-M4

[15] M. Anastasova, M. Bisheh Niasar, H. Seo, R. Azarderakhsh, and M. Mozaffari Kermani, "Efficient and side-channel resistant design of high-security ed448 on ARM cortex-M4."

[16] D. J. Bernstein, "Curve25519: New Diffie-Hellman Speed Records," in *Public Key Cryptography - PKC 2006*, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds. Springer Berlin Heidelberg, pp. 207–228.

[17] D. Hankerson, S. A. Vanstone, and A. Menezes, "Guide to elliptic curve cryptography," in *Springer Professional Computing*.

[18] T. Schneider and A. Moradi, "Leakage assessment methodology," vol. 6, no. 2, pp. 85–99. [Online]. Available: https://doi.org/10.1007/s13389-016-0120-y

[19] M. Bisheh Niasar, M. Anastasova, A. Abdulgadir, H. Seo, and R. Azarderakhsh, "Side-Channel Analysis and Countermeasure Design for Implementation of Curve448 on Cortex-M4."

[20] N. Samwel, L. Batina, G. Bertoni, J. Daemen, and R. Susella, "Breaking Ed25519 in WolfSSL," in *Topics in Cryptology CT-RSA 2018*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed. Springer International Publishing, pp. 1–20. [Online]. Available: https://nielssamwel.nl/papers/ctrsa2018_wolfssl.pdf

[21] libtom, "LibTomCrypt." [Online]. Available: https://www.libtom.net/LibTomCrypt/

[22] NewAE, "CHIPWHISPERER." [Online]. Available: https://www.newae.com/chipwhisperer