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*— An estimated 14.7 billion Internet of Things (IoT) devices will be connected to the Internet by 2023. The ubiquity of these devices creates exciting new opportunities, while at the same time introducing new concerns about privacy and security. To address these concerns, efficient cryptographic algorithms are needed to secure communication between IoT devices. In this work, we present an optimized implementation of one such algorithm, the Edwards Curve Digital Signature Algorithm (EdDSA) with operations Keygen, Sign, and Verify using the Ed25519 parameter on the ARM Cortex-M4 implemented in assembly code. The ARM Cortex-M4 is used in millions of devices world-wide, and is a popular choice for a wide range of IoT applications. We discuss the optimization of field and group arithmetic on this platform 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.**

***Index Terms*— Digital signatures, message authentication, elliptic curve cryptography, public key cryptography, embedded systems, side channel attacks, cryptography.**

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 has started a revolution in connectivity worldwide, while simultaneously creating difficult security and privacy problems [1], [2], [3]. Fast, efficient and secure cryptosystems such as Elliptic Curve Cryptography (ECC) are needed as part of the solution.

Manuscript received 25 September 2023; revised 1 February 2024; accepted 16 March 2024. Date of publication 15 April 2024; date of current version 30 May 2024. This work was supported by NSF Resilient & Intelligent NextG Systems (RINGS) under Grant 2147196. This article was recommended by Associate Editor C. H. Chang. *(Corresponding author: Daniel Owens.)*

Daniel Owens and Rabih El Khatib are with the Computer and Electrical Engineering and Computer Science Department and the Institute for Sensing and Embedded Network Systems Engineering (I-SENSE), Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: jowens17@fau.edu; relkhatib2019@fau.edu).

Mojtaba Bisheh-Niasar is with Microsoft, Redmond, WA 98052 USA (e-mail: mojtabab@microsoft.com).

Reza Azarderakhsh is with the Computer and Electrical Engineering and Computer Science Department and the Institute for Sensing and Embedded Network Systems Engineering (I-SENSE), Florida Atlantic University, Boca Raton, FL 33431 USA, and also with PQSecure Technologies LLC, Boca Raton, FL 33431 USA (e-mail: razarderakhsh@fau.edu).

Mehran Mozaffari Kermani is with the Computer Science and Engineering Department, University of South Florida, Tampa, FL 33620 USA (e-mail: mehran2@usf.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2024.3384414.

Digital Object Identifier 10.1109/TCSI.2024.3384414

Ed25519, one such system, is the Edwards Digital Signature Algorithm (EdDSA) configured to use the elliptic curve Edwards25519 and SHA-512 introduced in 2011 [4]. It was standardized in 2019 by the NIST in FIPS 186-5 [5] and is used to produce public keys, compute digital signatures of messages, and verify signature and message pairs. It is widely adopted, having been included in modern global applications such as the Transport Layer Security (TLS) protocol version 1.3 and the Secure Shell (SSH) protocol.

Ed25519 was designed for low latency Keygen, Sign, and Verify while providing a high security level with small (64 byte) signatures. [4] The version standardized by the NIST has been proven by [6] to provide *strong unforgeability under chosen message attacks* (SUF-CMA) security, which is a desired property of modern digital signature schemes. [6] These factors position Ed25519 as an excellent choice for IoT devices, which are typically computationally and spatially constrained. However, like other ECC cryptosystems, the large number of multi-precision mathematical operations pose a significant hurdle in the way of achieving a low-latency implementation. This is especially true for many IoT devices which utilize processors with 32-bit or lower architectures and therefore require many cycles to process 256- or 512-bit numbers.

The target processor, the 32-bit ARM Cortex M4, is widely used in many embedded and IoT applications due to its powerful ALU, high CPU frequency, and relatively spacious RAM and ROM especially when compared to older platforms like the AVR ATmega or MSP430. [7] The ARM Cortex-M processors have been shipped in more than 47 billion chips world-wide, and are used for a wide range of embedded and IoT applications from medical sensors to UAV flight control systems [8], [9]

Although Ed25519 was only recently standardized in 2019, in 2022, the first winners of the National Institute of Standards and Technology (NIST) competition [10] to find quantum-resistant cryptographic algorithms was announced [11]. During the transition period from classical algorithms like Ed25519 to post-quantum cryptography (PQC) algorithms, hybrid cryptosystems such as [12] will be needed to maintain regulations and standards [13], [14]. Hence, continued research that improves latency and reduces resource utilization for ECC in constrained devices is still relevant.

A. 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: active and passive. Active attacks are fault attacks, whereas passive attacks, performed via side-channel analysis (SCA), include power and timing attacks [15], among others. In this work we focus on defense against passive attacks.

B. 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 [16] which was employed and briefly described by [17] in their implementation of Ed25519. Using this method, we further reduce the latency of Ed25519 cryptographic primitives compared to the prior fastest effort by up to 52%.

For the first time, side-channel countermeasures are evaluated while using the Signed Comb scalar multiplication method. Software from previous works using the Signed Comb method have not included countermeasures against side-channel attacks. Additionally, our software for the ARM Cortex-M4 is open source and has been made available online. [18]

C. Disclaimer

The software is provided “as-is” and readers are encouraged to use it at their own discretion.

D. Organization

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

II. Preliminaries

A. Ed25519

The Edwards-Curve Digital Signature Algorithm (EdDSA) with parameter Ed25519 is defined in [19], 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

$$d = 37095705934669439343138083508754565189542113879843219016388785533085940283555$$

$$d = 370957059346694393431380835087545 6518\ldots$$
$$\ldots 9542113879803219016388785533085940283555$$

with E as the order of the curve. A point P on the curve is defined as $P = (x, y)$, $P \in E$ and $x, y \in \mathbb{F}_p$. A combination of finite field operations are performed on values in the field $\mathbb{F}_p$ such as long integer addition, subtraction, and multiplication. With these, the group operations Point Addition and Doubling can be constructed which form the basis of scalar multiplication: $Q = kP$ where $k \in \mathbb{Z}$ and $P, Q \in E$. As the order of E grows very large, such as the group order l in Ed25519, it becomes infeasible to solve for k if given Q and P [20].

Algorithm 1 Ed25519 Keygen

```
Input: $sk_{\mathbb{Z}} \in \{0, 1\}^{256}$
Output: $pk$
1 $sk' \leftarrow \texttt{prune}: sk_{\mathbb{Z}} \rightarrow sk_{\mathbb{Z}/\mathbb{F}_p}$
2 $s \leftarrow H(sk')_{\langle 0,1,\ldots,31 \rangle}$          {Lower 32 bytes}
3 $pk \leftarrow [s] \cdot B$
4 return $pk$
```

Algorithm 2 Ed25519 Sign

```
Input: $sk_{\mathbb{Z}}$, $pk_{\mathbb{F}_p}$, message $M$
Output: $sign \equiv R||S$
1 $sk' \leftarrow \texttt{prune}: sk_{\mathbb{Z}} \rightarrow sk_{\mathbb{Z}/\mathbb{F}_p}$
2 $s \leftarrow H(sk')_{\langle 32,33,\ldots,63 \rangle}$          {Upper 32 bytes}
3 $r \leftarrow H(p||M) \mod l$
4 $R \leftarrow [r] \cdot B$
5 $k \leftarrow H(R||pk||M) \mod l$
6 $S \leftarrow (r + k \cdot s) \mod l$
7 return $R||S$
```

Algorithm 3 Ed25519 Verify

```
Input: $pk_{\mathbb{F}_p}$, $M$, $sign$
Output: true / false
1 $A \leftarrow \texttt{decompress}(pk)$
2 $k \leftarrow H(R||A||M)$
3 $v \leftarrow [S] \cdot B - [k] \cdot A \equiv R$
4 return $v$
```

Group operations form the basis the EdDSA cryptographic primitives Keygen, Sign, and Verify (see Algorithms 1, 2, and 3 respectively, from [19], [21]). In these algorithms, H denotes SHA-512, $[x] \cdot y$ denotes scalar multiplication of the scalar $[x]$ with the point y, and $||$ denotes bit-string concatenation. l denotes the Ed25519 group order, and B the Ed25519 base point [19].

B. The ECC Operational Structure

The operational dependencies of elliptic curve based cryptography are structured like a pyramid, where each layer is composed of the operations below it, see Fig. 1. Finite field arithmetic of long integers forms the foundation, which are used to create group operations, which are then used to form cryptographic primitives such as Keygen.

To obtain an efficient and secure design for EdDSA or any ECC protocol, multiple optimization strategies and side-channel countermeasures must be added to each operational layer. Since each stratum of operation is tightly coupled, reductions in latency of field operations can have a large impact on the latency of group operations and cryptographic primitives. $\mathbb{F}_p$ long integer multiplication and inversion are especially costly. At the group level, scalar multiplication is responsible for the largest amount of latency in any given cryptographic primitive, see Fig. 2. As such, this work is mainly focused on the optimization of these operations to reduce latency.

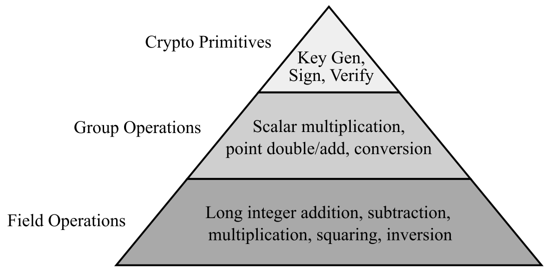

Crypto Primitives
Key Gen, Sign, Verify
Group Operations
Scalar multiplication, point double/add, conversion
Field Operations
Long integer addition, subtraction, multiplication, squaring, inversion

Fig. 1. Ed25519 Operational Structure.

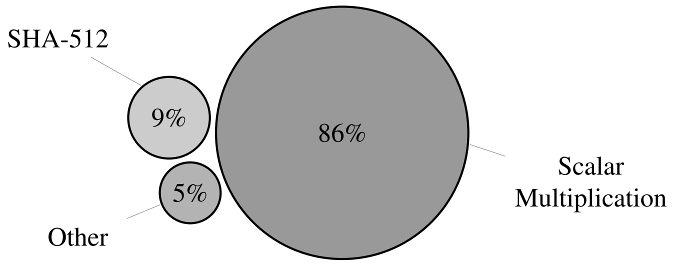


Fig. 2. Ed25519 Keygen algorithm steps as a percentage of total Keygen clock cycles, using Montgomery Ladder Method for scalar multiplication. No SCA mitigation present.

TABLE I
COST OF 255-BIT SCALAR MULTIPLICATION METHODS IN 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* (average)	255PD+(255/(w + 1))PA
Signed Comb*	($s_{\max}-1$)PD+($\sum_{j=0}^{n-1} s_j$)PA
Signed Comb ($n=3$, $t=5$, $s=17$)*	16PD+51PA
Signed Comb ($n=4$, $t=4$, $s=24$)*	23PD+96PA

*Implemented and included in the source code.

1) Scalar Multiplication: There are many scalar multiplication methods, see Table I for a handful of the most well-known. The Signed Comb Method and Montgomery Ladder Method are included in this work, with the latter presented mostly as a point of comparison and alternative to the less studied and implemented Signed Comb Method.

C. 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) [22]. 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).

Most instructions on the target platform execute in one clock cycle. The most notable exceptions are instructions that access or write to memory, which require 2-3 cycles to execute. Therefore, it is desirable to keep data in the GPRs `R0-R15` as long as possible. Notably, access to the FPRs requires only 1 cycle for each register being read or written to, and data sent between an FPR and GPR is not modified in any way. Non-floating point instructions cannot be performed on data in the FPRs, which positions them as an efficient alternative to the stack in some scenarios.

TABLE II
LATENCY OF ARMv7-M ISA MEMORY ACCESS AND MAC INSTRUCTIONS IN CLOCK CYCLES (CC) [24]

Instruction	**Functionality**	**Timing (CC)**
`LDR/STR`	$R_n \leftarrow$ memory memory $\leftarrow R_n$	2
`VMOV`	$R_d \leftarrow S_m$ $S_d \leftarrow R_m$	1
`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 ARM Cortex-M4 features a three-stage pipeline that enables the processing of up to three instructions at once after the pipeline has been filled, unless the instructions are 16-bit, in which case the pipeline is restricted to two instructions at a time [22], [23]. Using the pipeline, repeated memory operations to or from registers can be optimized such that n accesses requires only $n+1$ cycles to perform [17]. Careful instruction ordering is used throughout our implementation to keep the pipeline filled, reducing memory access overhead.

1) Portability: In order to achieve the lowest possible latency, all $\mathbb{F}_p$ operations, both scalar multiplication methods, and some supporting routines such as Point Addition and Point Doubling are implemented in assembly. C is used to direct program flow and to implement smaller, less frequent operations. We acknowledge that this reduces code portability; however the ARM Cortex-M4 platform is a very popular choice in IoT applications, ensuring the relevancy of this work.

2) Memory Caching: The platform used for development as well as the platform used for side-channel analysis both lacked a memory cache. Therefore, this work assumes that the ARM Cortex-M4 running the software does not have a memory cache. No attention is paid to obfuscating the reading of secret addresses in memory using techniques such as those presented in [4].

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 [25] and written in ARM assembly, was utilized in our work to perform $\mathbb{F}_p$ addition, subtraction, and multiplication. This project offers the fastest known $\mathbb{F}_p$ arithmetic routines for ARMv7-M architectures.

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. See Table II for a comparison of costs in clock cycles for various ARMv7-M assembly instructions important to this implementation.

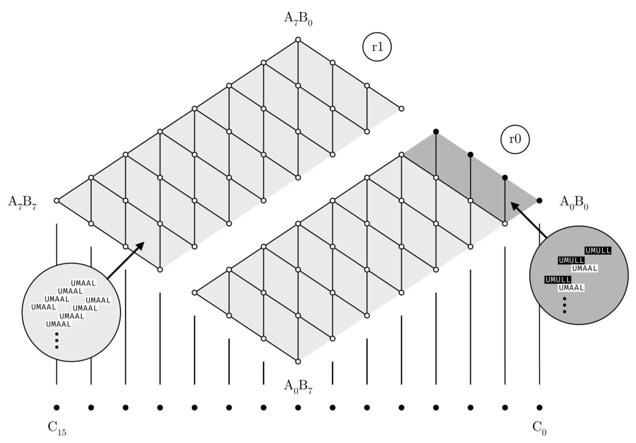


Fig. 3. Rhombus representation of the multiplication strategy. Execution proceeds right to left, r_0 first then r_1.

1) 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 subtraction is implemented in a similar manner with the instructions `SBC` and `SBS`.

2) 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 3, is similar to the Operand Caching techniques described by [17], [26], and [27] where the multiplication is performed in row-order. Fig. 3 illustrates the operational structure. Execution proceeds from right to left, starting from `r0`, with horizontal connections indicating multiplication and vertical indicating addition.

3) Modular Inversion: Although the `X25519-Cortex-M4` project includes code for inversion, we chose to implement the Itoh-Tsuji method proposed in [28] which uses 11 modular multiplications and 254 modular squares to compute $a^{p-2} \equiv a^{-1} \bmod p$ in 53% less cycles than [17].

B. The Signed Comb Scalar Multiplication Method

Currently the most efficient constant time scalar multiplication algorithm, the Signed Comb method [16] is used for Ed25519 Keygen and Sign in our implementation. It trades latency for space by using precomputed multiples of the point P to be used in scalar multiplication.

For any given Signed Comb with parameters n, t, s, the algorithm proceeds by splitting the total bits of the scalar to be processed $n \cdot t \cdot s$ into n blocks which are $t \cdot s$ bits wide. Each of these blocks is processed in-turn by a "comb" with t teeth spaced s bits apart. For each block, the comb is shifted one bit a time t times; at each iteration a bit-string x is formed using the bits of the scalar at the positions of the teeth of the comb. A multiple of the desired point is then chosen using the value of x for each block, which are then added together then doubled. For an illustration of this process, see Figure 4. For Signed Comb without SCA mitigation, this work uses the parameters $n = 3$, $t = 5$, and $s = 17$, where

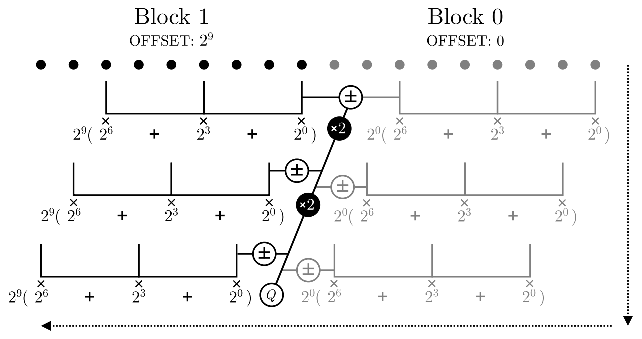


Fig. 4. Example of a comb operation where $n = 2$, $t = 3$, and $s = 3$.

$n \cdot t \cdot s = 255 = \log_2(p)$. With SCA mitigation, the parameters $n = 4$, $t = 4$, and $s = 24$ are chosen. See §IV.

The Signed Comb Method is composed of three processes:

1) Offline Precomputation
2) Online Scalar Signed Binary Conversion
3) Online Scalar Multiplication

1) Offline Precomputation: The Signed Comb method uses precomputed multiples $M_{j,k}$ of a point P 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 $M_{j,k}$, first observe that for each $j \in [0, n)$ combs, each comb has $k = t_j$ teeth spaced s_j bits apart, and an offset o_j for each block j of the scalar d. [16] For each comb n_j with teeth t_j, the total possible values of the bit-string x composed of t_j bits is 2^{t_j}. The base point is then doubled and added or subtracted to itself the corresponding amount of times to produce a multiple xE for each block n [16]. Finally, xE is doubled o_j times to produce the final multiple. [16] Additionally, because d is in signed binary form, we can exploit the fact that $+1 \equiv -(-1)$ and discard every multiple where the leading coefficient of the bit-string x is 1, reducing the number of total multiples in half from 2^t to 2^{t-1} [16]. See Algorithm 4 for implementation details.

Finally, our comb parameters $n = 3$, $t = 5$, and $s = 17$ require 48 precomputed multiples using 4.5 KB of ROM. With SCA mitigation, parameters $n = 4$, $t = 4$, and $s = 24$ require 32 precomputed multiples using 3.1 KB of ROM. Indeed, there are many possible values of (n, t, s) with a wide range of trade-offs. In our work, $(3, 5, 17)$ was chosen for the variant with no mitigation due to the desirable balance between computational and space complexity, as shown in [16]. See §IV for details on the parameters $(4, 4, 24)$.

2) Online 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 [16]:

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

where $d_i \in \{\pm 1\}$ and $D > \log_2(l)$. D can be set to the length of the input scalar e. 2^D is precomputed and stored in ROM.

3) Online Scalar Multiplication: The computation stage consists of a small amount of point doubles and adds, which

Algorithm 4 Signed Comb Precomputation

```
Input: A point $P \in E$, comb parameters $n, t, s$
Output: $M$ multiples of the point $P$
1  $R \leftarrow P$
2  for $i = t - 1$ down to 0 do
3      for $j = 0$ up to $t - i$ do
4          $R \leftarrow 2R$
5      $\text{Powers}_{t-1-i} \leftarrow R$
6  for $i = 0$ up to $n$ do
7      for $j = 0$ up to $2^{t-1}$ do
8          $Q \leftarrow P$
9          $x \leftarrow 2^{t-1} + j$
10         for $k = 0$ up to $t$ do
11             bit = $(x \gg k)$ & 1
12             if $bit = 1$ then
13                 $Q \leftarrow Q + \text{Powers}_k$
14             else
15                 $Q \leftarrow Q - \text{Powers}_k$
16     offset $\leftarrow s \cdot t \cdot i$
17     for 0 up to offset do
18         $Q \leftarrow 2Q$
19     $M_{i,j} \leftarrow Q$
20 return $M$
```

Algorithm 5 Signed Comb Scalar Multiplication

```
Input: signedBinary:$[e] \mapsto [d]$, $M$ multiples of the point $P$
Output: $[e] \cdot P$
1  $Q \leftarrow P$
2  for $i = s - 1$ down to 0 do
3      $Q \leftarrow 2Q$
4      for $j = 0$ up to $n - 1$ do
5          idx $\leftarrow 0$
6          for $k = 0$ up to $t - 1$ do
7              bit $\leftarrow i + s(k + j \cdot t)$
8              idx $\overset{+}{\leftarrow} d_{\text{bit}} \cdot 2^k$
9          $Q \leftarrow Q \pm M_{j,\text{idx}}$
10 return $Q$
```

Algorithm 6 Montgomery Ladder Scalar Multiplication

```
Input: An $m$-bit scalar $[s]$, and the $x$-coordinate $x_P$ of a point $P$
Output: $x_P = [s] \cdot P$
1  $D \leftarrow x_P$; $R_0 \leftarrow (1, 0)$; $R_1 \leftarrow (x_P, 1)$
2  for $j \leftarrow m - 1$ down to 0 do
3      $R_0, R_1 \leftarrow$ Cswap$(R_0, R_1, \text{bit}_j)$
4      $R_0, R_1 \leftarrow$ LadderStep$(D, R_0, R_1)$
5  return $R_0$
```

vary based on the (n, t, s) parameters chosen. After multiples of P have been precomputed, for each comb $j \in [0, n)$, double, then add or subtract the appropriate multiple. Let $Q = P$. The computation proceeds as follows in Equation 2, demonstrated in [16]:

$$Q_{\text{Final}} = \sum_{j=s-1}^{0} 2Q + \sum_{k=0}^{n-1} Q \pm M_{j,k} \pmod{l} \qquad (2)$$

where $2Q$ is Point Doubling and $\pm$ is Point Addition between the intermediate point Q and the possibly inverted multiple $M_{j,k}$. Point inversion is always performed, and $M_{j,k}$ or $-M_{j,k}$ is chosen using the addition masking method proposed in [29], see Algorithm 5 for implementation details. Point Addition (PA) and Point Doubling (PD) are implemented as demonstrated in [30]. For a comparison of PA/PD costs between scalar multiplication techniques, see Table I.

C. The w-NAF Scalar Multiplication Method

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 [31].

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 is decompressed from the public key multiplied by the hash $H(R, A, M)$, see Algorithm 3 for details. To decompress the public key, we use the fast decompression method proposed in [4] 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. [4]

Although fast, the Signed Comb Method is not a viable option for Verify. In Keygen and Sign, the point used for scalar multiplication is always the Ed25519 base point B, which allows precomputation. In addition to B, Verify uses another point A which is not known before runtime. Although it is possible to compute the multiples of A on-the-fly, and ignoring the additional space requirement, the time cost was found to be $\approx 700,000$ clock cycles. This places the approach behind the slower alternative Montgomery Ladder before multiplication can even begin.

D. The Montgomery Ladder Scalar Multiplication Method

The Montgomery Ladder, proposed by Pollard in 1987 [32], is a scalar multiplication method that has many desirable properties [33], has been well studied and implemented [4], [33], [34], and has attained a high level of trust and familiarity. For these reasons, we wanted to include the Montgomery Ladder in our implementation as a trusted point of reference to compare against the Signed Comb Method, and as an alternative to it for those wishing to use our software.

Our implementation of the Montgomery Ladder is presented in Algorithm 6, which is similar to Algorithm 3 presented in [35]. The primary difference between them is the manner of the `Cswap()` function which swaps the points R_0 and

```
    mls r8,r1,r2,r8
    mls r9,r1,r2,r9
    mla r10,r1,r2,r10
    mla r11,r1,r2,r11
```

Fig. 5. Constant time `Cswap()` in ARM assembly. `r1,r2` hold the value of bit from Algorithm 6 and the constant `#64` respectively. `r8,r9` hold pointers to the x- and y-coordinates of the point R_0 and `r10,r11` hold pointers to the x- and y-coordinates of the point R_1. For example, in the first line of the presented code, `r1` is multiplied by `r2` and the result is subtracted from `r8`.

R_1 before each execution of `LadderStep()`. Our implementation of `Cswap()` is similar to the `Cswap2()` function presented in [35], but implemented in assembly using the two ARM instructions `MLS` and `MLA` which multiply two values then add or subtract a third value. The code for this operation can be viewed in Figure 5. The points R_0, R_1 are 64 bytes apart in our implementation, which allows this optimized technique. The end result is that the inputs R_0, R_1 to `LadderStep()` conditionally swap in constant time. As mentioned in §II, it is assumed that no memory cache is present. The presence of a cache may introduce side-channel leakage.

The details of the `LadderStep()` function are excluded for brevity. It is exactly as presented by [35] in their Algorithm 2.

E. Extended Twisted Edwards Coordinates

The specification for Ed25519 [19] instructs the implementer to project affine points of the form (x, y) to Twisted Edwards Curve points of the form $(X : Y : Z)$ as described in [36], which decreases the latency of group operations like Point Addition. Further optimization is possible by alternatively using extended twisted Edwards coordinates of the form $(X : Y : Z : T)$ as described in [30], where an auxiliary coordinate $t = xy$ is introduced to extend affine coordinates, which map to projective as $(x, y, t) \mapsto (x : y : t : 1)$ [30]. An extended twisted Edwards coordinate corresponds to the extended affine point $(X/Y, Y/Z, T/Z)$ with $Z \neq 0$.

In this work, all group operations are performed on extended twisted Edwards coordinates as described in [30]. Here, there is an obvious trade-off between time and space complexity, since an extra coordinate T is necessary for every point. The extra space requirements are negligible in implementations using scalar multiplication methods that do not make use of precomputed data, like Double and Add or Montgomery Ladder. However, in the worst case for our implementation, 48 precomputed points are required, which introduces an extra 1.54 KB of data. To alleviate the space requirement, we chose to take $Z = 1$ for all precomputed points, allowing its exclusion from storage.

IV. Side-Channel Analysis

Side-channel attacks exploit minute similarities in some measure, such as power consumption, between many discrete cryptographic operations to recover secret information [37], [38], [39]. In our analysis, we assume that an attacker is able to observe and record with great accuracy minute fluctuations in power consumption, referred to as leakage traces, while cryptographic operations are performed. It is also assumed that an attacker can gather an arbitrary number traces with the chosen input and corresponding output, and that the attacker can generate templates.

A. Scope of Analysis

Although our implementation covers all Ed25519 primitives, our primary focus for SCA evaluation are the scalar multiplication portions of Keygen and Sign. Verify is omitted since it is only used to process public information. Therefore, discussion and results will center around the observation of various scalar multiplication algorithms with different countermeasures and parameters. In addition, the hash function SHA-512 is known to be insecure as demonstrated by [40]. Although it is possible to make SHA-512 secure against DPA attacks by padding the input message with random bytes [41], it breaks compatibility with FIPS 186-5, so this solution was not implemented. We consider securing SHA-512 out of scope for this work, but for development and benchmarking, an insecure and open source version of SHA-512 was used as provided by [42].

B. Types of Attacks

First, we present the types of attacks we consider in this work, then how our countermeasures apply to each.

1) Passive Attacks: We consider these ECC passive attacks: Simple/Differential power analysis (SPA) [37], Horizontal collision/correlation attacks ([43], [44], [45], [46]), Template attacks (TA) [46], Deep-learning attacks [46], Online template attacks (OTA) [46], [47], Horizontal `cmov` attacks ([46], [48]), and Soft-analytical side-channel attacks (SASCA) [46].

Address-based attacks such as Address-bit DPA and Address template attacks [46] are out of scope for this work.

2) Active Attacks: Active attacks, such as fault injection, are considered out of scope for this work. However, Weak-curve attacks [46] are relevant to EdDSA. In a weak-curve attack, an attacker may choose a point on another curve as input to the protocol in an attempt to weaken the security of the implementation [46]. In EdDSA it is impossible to select an arbitrary point on the curve as input. Therefore, EdDSA and by extension Ed25519 are not vulnerable to weak-curve attacks.

C. Countermeasures

1) SPA/DPA Countermeasures: As suggested by [49], coordinate randomization and scalar blinding are performed for every execution. Constant-time operations are also utilized at every level of the design as suggested by [37].

2) Horizontal Correlation/Collision Attack Countermeasures: This attack relies on identical field values being used across two or more scalar multiplication loop iterations [46] which occurs in every version of our Montgomery Ladder algorithm. One possible solution is to randomize the coordinate in every loop iteration as proposed and implemented by [46] in their Ephemeral X25519 implementation. In our implementation of the Signed Comb Method with Coordinate Randomization, the coordinates are randomized before every iteration by necessity, see §IV-E for more detail.

3) Horizontal `cmov`*, Template, Deep-Learning, SASCA, and Online Attack Countermeasures:* According to [46] and [50], projective (coordinate) randomization is enough to defeat these attacks, which is present in our implementation. However, it is unclear what frequency of randomization is required. The authors of [46] express doubt as to whether this countermeasure can stop all single-trace attacks, which we must express by extension for our implementation.

D. Scalar Blinding

Scalar Blinding is the process of algebraically randomizing a scalar $s \rightarrow s'$ such that $[s'] \cdot P \equiv [s] \cdot P$ [49]. The process ideally changes every bit of s, and is meant to be performed prior to every scalar multiplication, reducing the probability that a correlation between power consumption and the bits of s can be discerned over many traces.

To perform scalar blinding, a scalar s is added to the multiplication of the group order l by some random $r \in \mathbb{Z}$, computing $s' = s + r \cdot l$, without reduction. This increases the bit-length of s from $\log_2(p)$ to $\approx \log_2(p) + \log_2(r)$, which can have a large negative impact on scalar multiplication latency given a sufficiently large r. Therefore, it is important to choose the smallest possible length of r for which the blinding is effective.

Determining this length is not obvious. *Coron* [49] suggests that 20 bits for r is enough. Although their countermeasures are broader in scope, Batina et. al. [46] use 64 bits to perform blinding. Liu et. al. [51] state that r must be greater than $\approx k/2$, where k is the power of the group order of their curve. They state this is due to the unique structure of k, which contains a large portion of 1s or 0s in succession; thus, the most significant bits of s' are those of s, which is undesirable. The Ed25519 group order l also exhibits this characteristic, allowing us to discern that for our case it must be that $\log_2(r) > \log_2(l)/2$. In our implementation, r is generated using the hardware psuedorandom number generator (PRNG) of the ARM Cortex-M4 and used to calculate s' at runtime. It is also possible, and preferred [46], to perform the blinding outside of the cryptographic core. The computational cost for the blinding process itself is low, requiring approximately one 512 bit multiplication. However, the increased length of s' greatly increases latency regardless of scalar multiplication method, see §V.

1) Signed Comb Method: After experimentation and simulation, it was determined that r could be at minimum 133 bits long to effectively blind the scalar using the Signed Comb Method. To accommodate the change, new Signed Comb parameters ($n = 4, t = 4, s = 24$) were chosen and a separate set of multiples generated, see §III and §V for the impact on implementation. For TVLA results, see Fig. 6.

2) Montgomery Ladder: As our focus was on the Signed Comb Method, a 133 bit r was also used for the blinded version of the Montgomery Ladder for simplicity and comparison. For TVLA results, see Fig. 7.

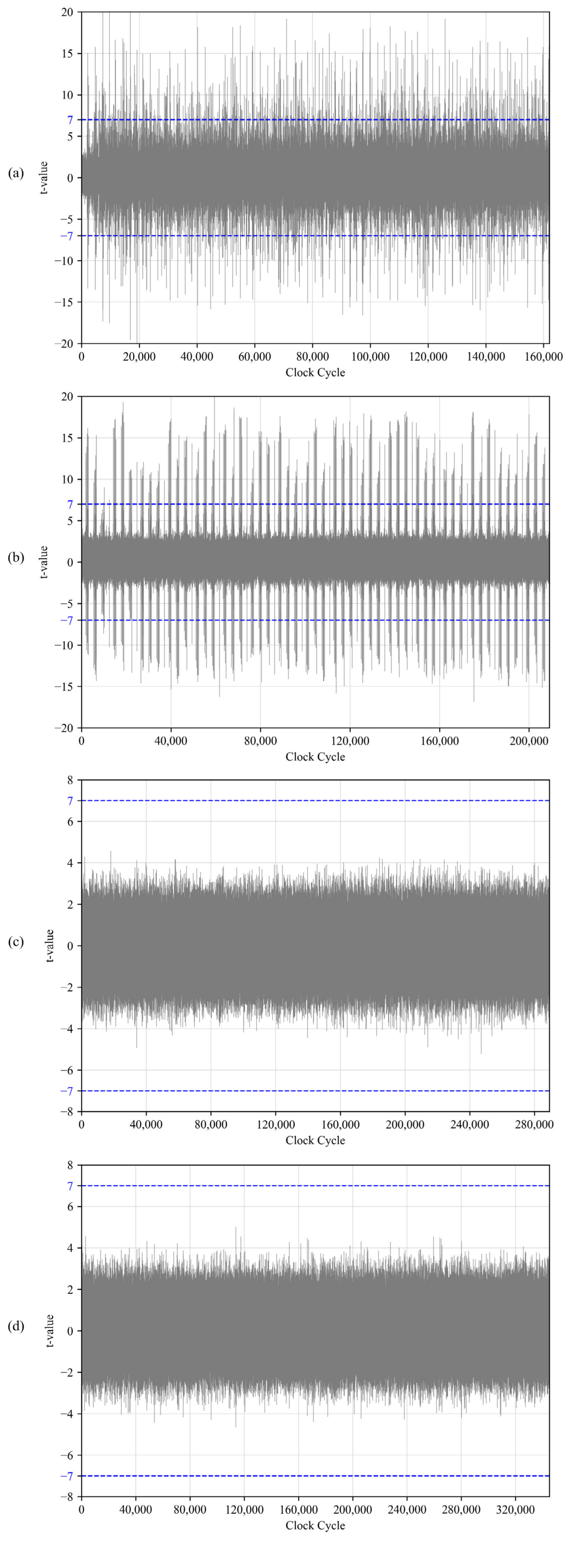


Fig. 6. TVLA results of the Signed Comb Method with various countermeasures. Threshold $T = 7$ for all observations. **(a)**: No SCA countermeasures. 1,000 traces were recorded. **(b)**: Coordinate Randomization. 1,000 traces were recorded. **(c)**: Scalar Blinding. 10,000 traces were recorded. **(d)**: Scalar Blinding and Coordinate Randomization. 10,000 traces were recorded.

E. Coordinate Randomization

Similar to Scalar Blinding, Coordinate Randomization [49] algebraically randomizes the coordinates of a point $P \rightarrow P'$

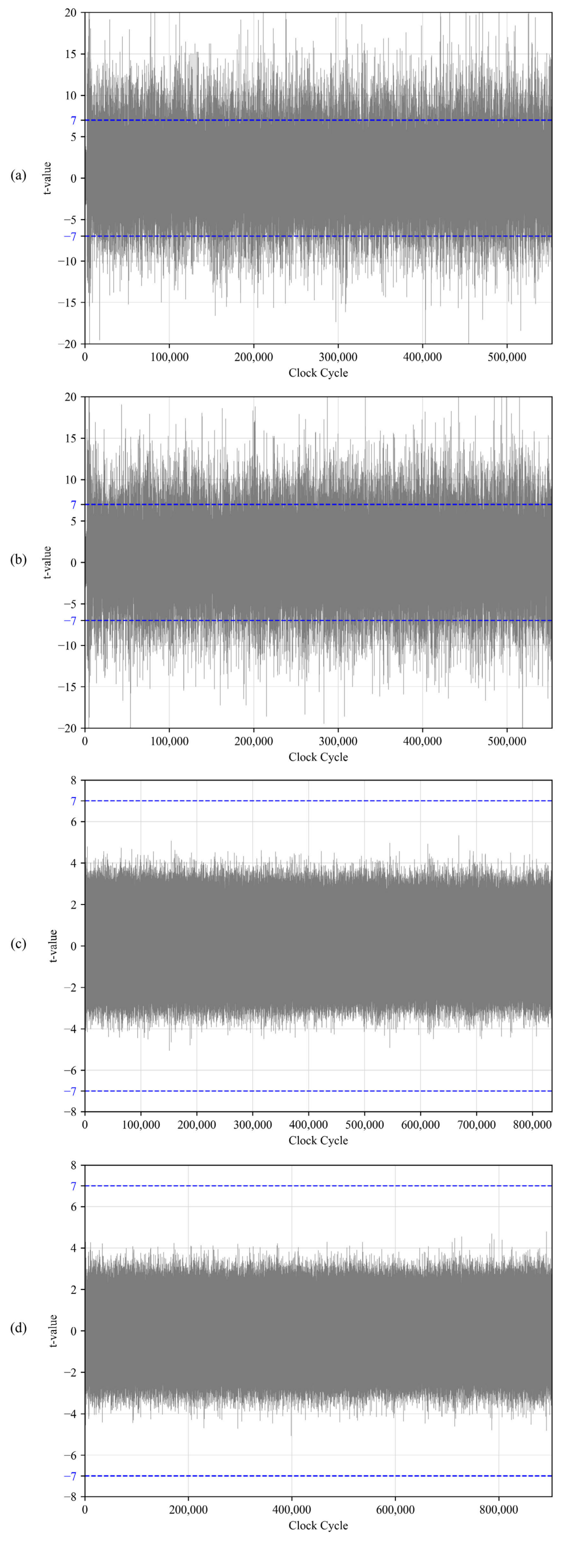


Fig. 7. TVLA results the of Montgomery Ladder Method with various countermeasures. Threshold $T = 7$ for all observations. **(a)**: No SCA countermeasures. 1,000 traces were recorded. **(b)**: Coordinate Randomization. 1,000 traces were recorded. **(c)**: Scalar Blinding. 10,000 traces were recorded. **(d)**: Scalar Blinding and Coordinate Randomization. 10,000 traces were recorded.

such that $[s'] \cdot P \equiv [s] \cdot P$, with the same goal as Scalar Blinding of reducing or removing the correlation between power consumption and the bits of s as they are processed.

This is achieved by selecting a random integer λ, then taking P' as the point $(\lambda \cdot x, \lambda)$ in affine coordinates and $(\lambda \cdot X, \lambda \cdot Y, \lambda, \lambda \cdot T)$ in extended projective [49]. Unlike Scalar Blinding, the multiplications with λ occur in the finite field $\mathbb{F}_p$. The correct result is recovered after scalar multiplication during the *unprojection* of the point $[s]P$ back to the affine with $x = \frac{\lambda X}{\lambda Z} = \frac{X}{Z}$, which occurs whether Coordinate Randomization is present or not. The computational cost of Coordinate Randomization is low. Assuming P is in the projective form, and $Z = 1$, only 3 field multiplications are required. Additionally, since the length of s is not modified, the latency of the scalar multiplication is not impacted.

1) Signed Comb Method: Coordinate Randomization is typically performed once prior to multiplication [49]. However, the Signed Comb Method makes use of many precomputed multiples of the Ed25519 base point B. Thus, generating B' prior to multiplication does not randomize a stored multiple xB required during operation. To address this, one random λ is generated prior to multiplication. During execution, after a multiple $B_{j,k}$ is retrieved from memory, $B'_{j,k} = (\lambda X : \lambda Y : \lambda Z : \lambda T)_{B_{j,k}}$ is calculated then used in Point Addition with the intermediate point Q.

2) Montgomery Ladder: Montgomery Ladder accesses the base point B once at the beginning of operation, which is multiplied by a random λ to produce B'. See Fig. 7

F. Test Vector Leakage Assessment

Ed25519 can be used in a *static* or *ephemeral* key-exchange scenario [46]. In the ephemeral case the key is changed after every use, but in the static case the key remains constant, which favors a passive attacker who can gather data from the device over many discrete observations. For evaluation, multiple non-specific fixed vs. random t tests were performed as presented in [52].

Test Vector Leakage Assessment (TVLA) is a method to efficiently detect leakage a passive attacker may be able to observe during cryptographic operations without the need to test and prove security for individual attacks [52]. It has three components: (1) the device under attack (DUT), (2) a high frequency oscilloscope, and (3) a host PC to store and process the recorded data as it is sent from the oscilloscope. [52]

The goal of performing TVLA is to determine if two sets of data are indistinguishable using statistical analysis. The statistical function used is Welch's $t-$test, which provides a quantitative measure for any given recorded sample if there is a difference between two populations of data Ω_0 and Ω_1. In the case of our non-specific fixed vs. random t tests, one population is the set of traces obtained using the fixed $sk_{\mathbb{Z}}$, and the other population the set of traces obtained using a random $sk_{\mathbb{Z}}$. The t-test is defined in [52] as:

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}}} \tag{3}$$

where μ_0 and μ_1 are the sample mean, s_0^2 and s_1^2 the sample variance, and n the number of traces gathered during the test.

The resultant t value is then compared against a threshold value T to determine if there is an observable difference between the two sets Ω_0 and Ω_1 at that point. The authors in [52] state that a threshold value of 4.5 is sufficient to achieve a confidence of > 0.99999, but the authors of [46] encountered many false positives when a high number of samples were recorded. Using the plot provided in [53], and following in the steps of [46], 7 was chosen as our threshold.

1) 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 [54].
- The NewAE CW308T-STM32F ARM Cortex-M4 target board, mounted on the NewAE CW308 UFO board [54].

The DUT was configured to operate 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.

G. TVLA of Implemented Scalar Multiplication Methods

1) TVLA with no Countermeasures: See Figs. 6 and 7, graph (a). A large amount of leakage over the T threshold can be observed for both the Signed Comb and Montgomery Ladder methods after just 1,000 traces, which is the expected result with no countermeasures implemented.

2) TVLA with Coordinate Randomization: See Figs. 6 and 7, graph (b). Here, only Coordinate Randomization was implemented. While the results for Montgomery Ladder appear identical to the version with no countermeasures, there is a noticeable difference compared to the previous Signed Comb graph. Interestingly, the operational structure of the Signed Comb Method can be clearly discerned in this graph. Comparing both results after just 1,000 traces, Coordinate Randomization is not effective enough for either Signed Comb or Montgomery Ladder.

3) TVLA with Scalar Blinding: See Figs. 6 and 7, graph (d). Here, only Scalar Blinding was implemented. Both TVLA results show leakage well under the $T = 7$ threshold after 10,000 traces. Note the increase in clock cycles for both scalar multiplication methods.

4) TVLA with Scalar Blinding and Coordinate Randomization: See Figs. 6 and 7, graph (d). Here, both countermeasures were included. Similar to Coordinate Randomization by itself, there is no significant reduction in leakage.

V. Performance and Conclusion

A. Comparison With Prior Work

Table III offers a latency comparison of our implementation versus prior implementations for $\mathbb{F}_p$ arithmetic, and cryptographic primitives. The works from *De Groot* [55] and *De Santis* [56] focus primarily on optimizing $\mathbb{F}_p$ operations. They also offer latency for X25519, but these measurements likely include an inversion, so they are not comparable to scalar multiplication only. The work by Fuji et. al. [17] presents

TABLE III
Latency Comparisons of Arithmetic in $\mathbb{F}_{2^{255}-19}$, and Ed25519 Cryptographic Primitive Operations Between Works in Clock Cycles (CC)

Reference	$\mathbb{F}_{2^{255}-19}$ Add/Sub	Square	Mult.	Invert
De Groot [55]	73	631	563	151,997
De Santis [56]	106	546	362	96,337
Fuji et. al. [17]	86	273	243	64,425
This work	44	151	111	30,665

Reference	Cryptographic Primitive Keygen	Sign	Verify
Fuji et. al. [17]	347,225	496,039	1,265,078
This work (No CM)	200,511	239,322	715,981
This work (CM)	324,527	363,826	-

CM is Countermeasures and includes Scalar Blinding and Coordinate Randomization. The lowest reported clock cycles from each source were chosen for comparison. Each number taken is the average of 512 executions.

TABLE IV
Clock Cycles for the Montgomery Ladder and Signed Comb Scalar Multiplication Methods on Edwards Curve25519

SCA Countermeasure	Freq. (MHz)	Clock Cycles Mont. Ladder	Clock Cycles Signed Comb
None	24	530,180	155,479
None	168	575,844	165,590
SB	24	800,322	277,644
SB	168	870,026	294,994
CR	24	574,780	210,152
CR	168	630,155	250,125
SB and CR	24	867,149	330,407
SB and CR	168	950,858	349,365

SB is Scalar Blinding. CR is Coordinate Randomization. All measurements taken on the STM32F470G. Each number taken is the average of 512 executions.

a full implementation of Ed25519 on the Cortex-M4 using the Signed Comb Method, but they do not provide specific measurements for scalar multiplication only. No prior work on the ARM Cortex-M4 reports latency with SCA countermeasures implemented. For full results of our scalar multiplication implementations see Table IV.

Where our work differs from these prior works, and most relevantly [17] is: (1) The use of [25] to obtain significantly faster $\mathbb{F}_p$ operations, (2) full disclosure of our Signed Comb and Montgomery Ladder implementation details, (3) side-channel countermeasure implementation and evaluation for both Signed Comb and Montgomery Ladder, and 4) our work is open source.

Compared to the previous fastest work of Fuji et. al. [17], our Keygen is 42% faster, Sign is 52% faster, and Verify is 43% faster.

B. Implementation Results

1) Scalar Multiplication: Clock cycles for scalar multiplication only are provided in Table IV to compare the Signed Comb and Montgomery Ladder methods with various SCA countermeasures included. Most notably, the Signed Comb method with both Scalar Blinding and Coordinate Randomization enabled is faster than Montgomery Ladder with no countermeasures.

TABLE V
LATENCY AND POWER MEASUREMENTS OF ED25519 CRYPTOGRAPHIC PRIMITIVES WITH VARIOUS COMBINATIONS OF SCALAR MULTIPLICATION ALGORITHM AND SCA COUNTERMEASURES ON THE STM32F470G AND NUCLEO-F411RE ARM CORTEX-M4 PLATFORMS

Device	Primitive	Scalar Mul Method	Countermeasure	Frequency (Mhz)	Clock Cycles	Latency (ms)	Power (mW)	Energy (μJ)
STM32F470G	Keygen	Montgomery Ladder	None	24	578,448	24.10	–	–
				168	628,797	3.74	–	–
			Scalar Blinding	24	850,020	35.42	–	–
				168	928,192	5.52	–	–
			Scalar Blinding & Coordinate Randomization	24	916,844	38.20	–	–
				168	994,046	5.92	–	–
		Signed Comb	None	24	200,511	8.35	–	–
				168	213,721	1.27	–	–
			Scalar Blinding	24	324,527	13.52	–	–
				168	344,052	2.05	–	–
			Scalar Blinding & Coordinate Randomization	24	377,175	15.72	–	–
				168	397,592	2.37	–	–
	Sign	Montgomery Ladder	None	24	617,623	25.73	–	–
				168	669,900	3.99	–	–
			Scalar Blinding	24	888,655	37.03	–	–
				168	968,622	5.77	–	–
			Scalar Blinding & Coordinate Randomization	24	955,482	39.81	–	–
				168	1,034,495	6.16	–	–
		Signed Comb	None	24	239,322	9.97	–	–
				168	254,426	1.51	–	–
			Scalar Blinding	24	363,826	15.16	–	–
				168	385,281	2.29	–	–
			Scalar Blinding & Coordinate Randomization	24	416,041	17.34	–	–
				168	438,230	2.61	–	–
	Verify	W-NAF	–	24	715,981	29.83	–	–
				168	767,180	4.57	–	–
NUCLEO-F411RE	Keygen	Montgomery Ladder	None	100	579,737	6.04	74.98	452.84
			Scalar Blinding	100	851,061	8.87	72.98	647.01
			Scalar Blinding & Coordinate Randomization	100	917,638	9.56	72.26	690.70
		Signed Comb	None	100	202,982	2.11	76.73	162.23
			Scalar Blinding	100	326,818	3.40	75.87	258.29
			Scalar Blinding & Coordinate Randomization	100	382,021	3.98	74.77	297.54
	Sign	Montgomery Ladder	None	100	613,825	6.39	74.62	477.11
			Scalar Blinding	100	884,847	9.22	73.12	673.97
			Scalar Blinding & Coordinate Randomization	100	951,628	9.91	73.04	724.06
		Signed Comb	None	100	237,665	2.48	76.33	188.100
			Scalar Blinding	100	361,627	3.77	75.06	282.75
			Scalar Blinding & Coordinate Randomization	100	416,110	4.33	74.77	324.10
	Verify	w-NAF	–	100	724,184	7.54	75.40	568.80

Numbers taken for this work are the average of 512 executions.

2) Cryptographic Primitives: Our implementations of Ed25519 Keygen, Sign, and Verify without SCA countermeasures achieve a new latency record on the ARM Cortex-M4 when compared to previous works as shown in Table III. Table V provides comprehensive latency and power measurements for each cryptographic primitive, using each scalar multiplication method, using combinations of SCA countermeasures. The measurements for Keygen, Sign, and Verify encompass everything presented in Algorithms 1, 2 and 3.

3) SCA Countermeasures: As discussed in §IV, SCA countermeasures can introduce a large amount of latency into the design. Using Signed Comb, about 175,000 clock cycles were added to Keygen and Sign with Scalar Blinding and Coordinate Randomization enabled. For the same countermeasures but using Montgomery Ladder, about 335,000 clock cycles were added to Keygen and Sign. Montgomery Ladder's runtime increases linearly with the length of the input scalar, which accounts for the large discrepancy between the two methods. For further details see Table I.

In the best case scenario, using the Signed Comb Method for scalar multiplication, an SCA protected ED25519 Keygen and Sign was achieved with a 53% and 58% overhead respectively.

4) Frequency and Latency: In Tables IV and V two frequencies are provided for the STM32F470G: 24 MHz and 168 MHz. 24 MHz is a "benchmarking" frequency where the CPU is never left waiting on the memory. 168 Mhz is the maximum frequency of the chip on that platform and is representative of a more realistic configuration for the M4. For the NUCLEO-F411RE device only one frequency was used to simplify measurements. See §V-C.

All code was compiled with `arm-none-eabi-gcc` version 10.2.1, using the `-O3` optimization flag.

C. Power Measurements

Power measurements were included as an additional point of comparison and evaluation. They were obtained using an X-NUCLEO-LPM01A "Power Shield", which monitored a NUCLEO-F411RE target device running our software. It was chosen for its direct compatibility with the X-NUCLEO-LPM01A to enable the collection of accurate and detailed power measurements. Like the STM32F507G, the NUCLEO-F411RE is a Cortex-M4 based platform with an FPU, ensuring compatibility with our software. Because the STM32F507G target device does not interface with the X-NUCLEO-LPM01A, power measurements were not obtained for it.

D. Conclusion

In this work, we presented an efficient and side-channel secure implementation of Ed25519 on the ARM Cortex-M4 using both the Montgomery Ladder and Signed Comb scalar multiplication methods utilizing optimized assembly throughout our design. It was shown to outperform existing works in latency when performing Keygen, Sign, and Verify by up to 52%. We also presented discussion and evaluation of various side-channel countermeasures for the Signed Comb and Montgomery Ladder scalar multiplication methods. Finally, latency and power consumption was reported for the design with and without side-channel countermeasures.

Acknowledgment

The authors would like to thank the reviewers for their comments.

References

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

[2] H. Mokari, E. Firouzmand, I. Sharifi, and A. Doustmohammadi, "Deception attack detection and resilient control in platoon of smart vehicles," in *Proc. 30th Int. Conf. Electr. Eng. (ICEE)*, 2022, pp. 29–35.

[3] H. Mokari, E. Firouzmand, I. Sharifi, and A. Doustmohammadi, "DoS attack detection and resilient control in platoon of smart vehicles," in *Proc. 9th RSI Int. Conf. Robot. Mechatronics (ICRoM)*, Nov. 2021, pp. 144–150.

[4] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *J. Cryptograph. Eng.*, vol. 2, no. 2, pp. 77–89, Sep. 2012.

[5] *FIPS PUB 186-5 (Draft)—Digital Signature Standard (DSS)*, National Inst. Standards Technology, Gaithersburg, MD, USA, 2019.

[6] J. Brendel, C. Cremers, D. Jackson, and M. Zhao, "The provable security of ed25519: Theory and practice," in *Proc. IEEE Symp. Secur. Privacy (SP)*. San Francisco, CA, USA: IEEE, May 2021, pp. 1659–1676.

[7] H. Seo and R. Azarderakhsh, "Curve448 on 32-bit ARM cortex-M4," in *Proc. Inf. Secur. Cryptol. ICISC*, vol. 12593, D. Hong, Ed. Cham, Switzerland: Springer, 2020, pp. 125–139.

[8] Y. Zefeng, L. Fang, R. Yibiao, C. Yaowen, and F. Zhun, "Design of a multichannel biomedical signal acquisition system based on cortex-M4 processor," in *Proc. IEEE Int. Conf. Cyber Technol. Autom., Control, Intell. Syst. (CYBER)*, Jun. 2015, pp. 1023–1027.

[9] S. Incicco, F. S. De Ferraris, and N. Real, "Embedded navigation and control system for UAVs ARM cortex m4," in *Proc. XVI Workshop Inf. Process. Control (RPIC)*, Oct. 2015, pp. 1–6.

[10] (Dec. 2016). *NIST Asks Public to Help Future-Proof Electronic Information*. [Online]. Available: https://www.nist.gov/news-events/news/2016/12/nist-asks-public-help-future-proof-electronic-information

[11] *NIST Announces First Four Quantum-Resistant Cryptographic Algorithms*, NIST, Gaithersburg, MD, USA, Jul. 2022.

[12] E. Bursztein and F. Kaczmarczyck. (2023). Toward Quantum Resilient Security Keys. Google Online Security Blog. Accessed: Aug. 19, 2023. [Online]. Available: https://security.googleblog.com/2023/08/toward-quantum-resilient-security-keys.html.

[13] N. Bindel, U. Herath, M. McKague, and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure," Cryptol. ePrint Arch., Int. Assoc. Cryptol. Res. (IACR), Tech. Rep. 2017/460, 2017.

[14] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Instruction-set accelerated implementation of CRYSTALS-Kyber," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 11, pp. 4648–4659, Nov. 2021.

[15] P. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Advances in Cryptology CRYPTO 1996*, vol. 1109, N. Koblitz, Ed. Berlin, Germany: Springer, 1996, pp. 104–113.

[16] M. Hamburg, "Fast and compact elliptic-curve cryptography," Int. Assoc. Cryptol. Res. (IACR), 2012. [Online]. Available: https://eprint.iacr.org/2012/309

[17] H. Fujii and D. F. Aranha, "Curve25519 for the cortex-M4 and beyond," in *Prog. Cryptol. LATINCRYPT*, T. Lange and O. Dunkelman, Eds. Cham, Switzerland: Springer, 2019, pp. 109–127.

[18] *Thecooldaniel/Ed25519-ARM-Cortex-M4*. Accessed: Mar. 3, 2024. [Online]. Available: https://github.com/thecooldaniel/Ed25519-ARM-Cortex-M4

[19] S. Josefsson and I. Liusvaara, *Edwards-Curve Digital Signature Algorithm (EdDSA)*, document RFC8032, Jan. 2017.

[20] R. Elkhatib, B. Koziel, R. Azarderakhsh, and M. M. Kermani, "Accelerated RISC-V for post-quantum SIKE," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 6, pp. 2490–2501, Jun. 2022.

[21] L. Chen, D. Moody, A. Regenscheid, and A. Robinson, "Digital signature standard (DSS)," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST FIPS 186-5, Feb. 2023.

[22] *ARM® Cortex®-M4 Processor Technical Reference Manual Revision: R0p1*, ARM, Cambridge, U.K., 2020.

[23] J. Yiu, *Definitive Guide to ARM Cortex-M23 and Cortex-M33 Processors*. Kidlington, U.K.: Elsevier, 2021.

[24] Arm Ltd., *Arm V7-M Architecture Reference Manual*, 2021. [Online]. Available: https://developer.arm.com/documentation/ddi0403/ee

[25] Emill, *X25519-Cortex-M4*, GitHub. 2018. [Online]. Available: https://github.com/Emill/X25519-Cortex-M4

[26] M. Anastasova, M. Bisheh-Niasar, H. Seo, R. Azarderakhsh, and M. M. Kermani, "Efficient and side-channel resistant design of high-security Ed448 on ARM cortex-M4," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Jun. 2022, pp. 93–96.

[27] H. Seo, P. Sanal, A. Jalali, and R. Azarderakhsh, "Optimized implementation of SIKE round 2 on 64-bit ARM cortex—A processors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 8, pp. 2659–2671, Aug. 2020.

[28] 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. Berlin, Germany: Springer, 2006, pp. 207–228.

[29] M. Anastasova, R. Azarderakhsh, and M. M. Kermani, "Fast strategies for the implementation of SIKE round 3 on ARM cortex-M4," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 10, pp. 4129–4141, Oct. 2021.

[30] H. Hisil, K. K.-H. Wong, G. Carter, and E. Dawson, "Twisted Edwards curves revisited," Cryptol. ePrint Arch., Int. Assoc. Cryptol. Res. (IACR), 2008. [Online]. Available: https://eprint.iacr.org/2008/522

[31] D. R. Hankerson, A. J. Menezes, S. A. Vanstone, D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography* (Springer Professional Computing). New York, NY, USA: Springer, 2004.

[32] P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Math. Comput.*, vol. 48, no. 177, pp. 243–264, 1987.

[33] M. Joye and S.-M. Yen, "The Montgomery powering ladder," in *Cryptographic Hardware and Embedded Systems—CHES 2002*, vol. 2523, G. Goos, J. Hartmanis, J. Van Leeuwen, B. S. Kaliski, Ç. K. Koç, and C. Paar, Eds. Berlin, Germany: Springer, 2003, pp. 291–302.

[34] D. R. L. Brown, "Multi-dimensional Montgomery ladders for elliptic curves," Cryptol. ePrint Arch., Int. Assoc. Cryptol. Res. (IACR), 2006. [Online]. Available: https://eprint.iacr.org/2006/220

[35] K. Nath and P. Sarkar, "Constant time Montgomery ladder," Int. Assoc. Cryptol. Res. (IACR), 2020. [Online]. Available: https://eprint.iacr.org/2020/956

[36] D. J. Bernstein and T. Lange, "Faster addition and doubling on elliptic curves," Cryptol. ePrint Arch., Int. Assoc. Cryptol. Res. (IACR), Tech. Paper 2007/286, 2007.

[37] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99*, M. Wiener, Ed. Berlin, Germany: Springer, 1999, pp. 388–397.

[38] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems–CHES 2004*, vol. 3156, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. Joye, and J.-J. Quisquater, Eds. Berlin, Germany: Springer, 2004, pp. 16–29.

[39] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems CHES 2008* (Lecture Notes in Computer Science), E. Oswald and P. Rohatgi, Eds. Berlin, Germany: Springer, 2008, pp. 426–442.

[40] N. Samwel, L. Batina, G. Bertoni, J. Daemen, and R. Susella, "Breaking Ed25519 in WolfSSL," in *Topics in Cryptology CT-RSA 2018* (Lecture Notes in Computer Science), N. P. Smart, Ed. Cham, Switzerland: Springer, 2018, pp. 1–20.

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

[42] libtom, GitHub. (2010). *LibTomCrypt*. [Online]. Available: https://github.com/libtom/libtomcrypt

[43] A. Bauer, E. Jaulmes, E. Prouff, and J. Wild, "Horizontal collision correlation attack on elliptic curves," in *Selected Areas in Cryptography–SAC 2013* (Lecture Notes in Computer Science), T. Lange, K. Lauter, and P. Lisoněk, Eds. Berlin, Germany: Springer, 2014, pp. 553–570.

[44] A. Bauer, E. Jaulmes, E. Prouff, J.-R. Reinhard, and J. Wild, "Horizontal collision correlation attack on elliptic curves," *Cryptogr. Commun.*, vol. 7, no. 1, pp. 91–119, Mar. 2015.

[45] I. Kabin, Z. Dyka, D. Klann, and P. Langendoerfer, "Horizontal attacks against ECC: From simulations to ASIC," in *Computer Security* (Lecture Notes in Computer Science), A. P. Fournaris, M. Athanatos, K. Lampropoulos, S. Ioannidis, G. Hatzivasilis, E. Damiani, H. Abie, S. Ranise, L. Verderame, A. Siena, and J. Garcia-Alfaro, Eds. Cham, Switzerland: Springer, 2020, pp. 64–76.

[46] L. Batina, L. Chmielewski, B. Haase, N. Samwel, and P. Schwabe, "SCA-secure ECC in software—Mission impossible?" Int. Assoc. Cryptol. Res. (IACR), 2021. [Online]. Available: https://eprint.iacr.org/2021/1003

[47] L. Batina, Ł. Chmielewski, L. Papachristodoulou, P. Schwabe, and M. Tunstall, "Online template attacks," *J. Cryptograph. Eng.*, vol. 9, no. 1, pp. 21–36, Apr. 2019.

[48] E. Nascimento, Ł. Chmielewski, D. Oswald, and P. Schwabe, "Attacking embedded ECC implementations through CMOV side channels," in *Selected Areas in Cryptography—SAC 2016* (Lecture Notes in Computer Science), R. Avanzi and H. Heys, Eds. Cham, Switzerland: Springer, 2017, pp. 99–119.

[49] J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Cryptographic Hardware and Embedded Systems*, vol. 1717, Ç. K. Koç and C. Paar, Eds. Berlin, Germany: Springer, 1999, pp. 292–302.

[50] D. Naccache, N. P. Smart, and J. Stern, "Projective Coordinates Leak," in *Advances in Cryptology—EUROCRYPT 2004*, vol. 3027, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, M. Y. Vardi, C. Cachin, and J. L. Camenisch, Eds. Berlin, Germany: Springer, 2004, pp. 257–267.

[51] Z. Liu, P. Longa, G. Pereira, O. Reparaz, and H. Seo, "FourQ on embedded devices with strong countermeasures against side-channel attacks," Cryptol. ePrint Archive, Int. Assoc. Cryptol. Res. (IACR), Paper 2017/434, 2017.

[52] T. Schneider and A. Moradi, "Leakage assessment methodology," *J. Cryptograph. Eng.*, vol. 6, no. 2, pp. 85–99, Jun. 2016.

[53] 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," in *Proc. 11th Int. Workshop Hardw. Architectural Support Secur. Privacy*, 2022, pp. 10–17.

[54] *CHIPWHISPERER*, NewAE, Dartmouth, NS, Canada, Jul. 2022.

[55] W. de Groot, "A performance study of X25519 on Cortex-M3 and M4 de Groot, W. (Author)," Master's thesis, Eindhoven Univ. Technol., Eindhoven, The Netherlands, Sep. 2015.

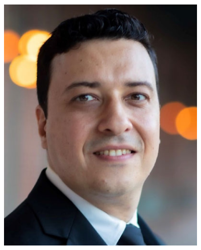
[56] F. De Santis and G. Sigl, "Towards side-channel protected X25519 on ARM cortex-M4 processors," *Proc. Softw. Perform. Enhancement Encryption Decryption, Benchmarking*, Utrecht, The Netherlands, 2016, pp. 19–21.

Daniel Owens received the M.Sc. degree in computer science from Florida Atlantic University (FAU) in 2022, where he is currently pursuing the Ph.D. degree. His current research interests are post-quantum cryptography, hardware implementations of cryptographic standards, and embedded devices, with a focus on side-channel analysis and security.

Rabih El Khatib is currently pursuing the M.Sc. degree with Florida Atlantic University. His research focuses on embedded security through implementation and optimization of classical and post-quantum cryptography, in addition to applying mitigation techniques against side-channel attacks.

Mojtaba Bisheh-Niasar received the Ph.D. degree in computer engineering from Florida Atlantic University in 2022. He is currently a Senior Hardware Security Engineer with Azure Hardware Security Architecture (AHSA), Microsoft, Redmond. His research interests include hardware implementation of applied cryptography, post-quantum cryptography, designing highly optimized computation architecture in industrial IoT and embedded devices, and side-channel protection.

Reza Azarderakhsh (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Western University in 2011. He is currently a Professor with the Department of Electrical Engineering and Computer Science (EECS), Florida Atlantic University. His current research interests include secure protocols design, finite field and its applications, elliptic curve cryptography, lattice-based cryptography, cryptographic engineering, side-channel analysis, and post-quantum cryptography. He was a recipient of the NSERC Post-Doctoral Research Fellowship. He is also a Public Speaker about post-quantum security and has given more than 50 talks in various venues to academia, industry, and government sectors. He was a former Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 2016 to 2023. He serves as a TPC member for various conferences, such as HOST, ARITH, DAC, and CANS.



Mehran Mozaffari Kermani (Senior Member, IEEE) received the B.Sc. degree from the University of Tehran, Tehran, Iran, in 2005, and the M.E.Sc. and Ph.D. degrees from the University of Western Ontario, London, Canada, in 2007 and 2011, respectively. In 2012, he joined the Electrical Engineering Department, Princeton University, NJ, USA, as a NSERC Post-Doctoral Research Fellow. From 2013 to 2017, he was a Faculty Member of Rochester Institute of Technology. In 2017, he was an Associate Professor with the Computer Science and Engineering Department, University of South Florida. Currently, he is serving as an Associate Editor for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the *ACM Transactions on Embedded Computing Systems*, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS. He has been a TPC member of the HOST (Publications Chair), the CCS (Publications Chair), DAC, DATE, RFIDSec, LightSec, WAIFI, FDTC, and DFT. He was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellowship in 2011, the Texas Instruments Faculty Award (Douglas Harvey) in 2014, Outstanding Research Award at College of Engineering, USF, in 2018, Nexus Initiative Global Award in 2019, and USF University-Wide Faculty Outstanding Research Achievement Award in 2021.