Highly optimized Curve448 and Ed448 design in wolfSSL and side-channel evaluation on Cortex-M4

***Abstract*—The compact key sizes and the low computational latency of the Elliptic Curve Cryptography (ECC) family of curves sparked high interest in their integration into network protocols. The recently suggested Curve448, assuring 224-bit security, is an ideal curve choice for integrating into cryptographic libraries according to a late study on backdoors on other ECC instances compromising their security, which results in the integration of Curve448 into the TLS1.3 protocol. Curve448 and its birationally equivalent untwisted Edwards curve Ed448, used for key exchange and authentication, respectively, present a perfect fit for low-end embedded cryptographic libraries due to their minimal memory requirements. In this work, we present the first adoption of highly optimized ECDH and EdDSA based on Curve448 and Ed448 into the widely employed IoT-focused cryptographic library wolfSSL. We evaluate the performance of the newly integrated architectures against the NIST recommended Cortex-M4 STM32F407-DK ARM-based platform. Additionally, we perform thorough security evaluation of the side-channel robustness of the implementation via powerful TVLA analysis revealing DPA data leakage. We integrate countermeasures to protect the design, evaluate their effectiveness and analyze the latency overhead. We achieve SCA robust Curve448 and Ed448 at the performance cost of $\sim 1,200KCC$ and $\sim 1.36\times$ the execution time, however, observe faster results when benchmarking wolfSSL, based on the WiFi quipped STM32F413-DK microcontroller, due to the handcrafted assembly implementation.**

Keywords: Elliptic Curve Cryptography, Curve448, Side-Channel Countermeasures, Cortex-M4.

I. Introduction

In recent years, technology has become more integrated into our daily lives, increasing the amount of data exchanged through the Internet. The rapid transmission speed and ease of getting information on low- or high-end nodes improve quality of life. Private information cannot be communicated over an insecure network like the Internet without being compromised. Cryptographic algorithms and network protocols use challenging mathematical issues to safeguard our data during transport.

Public Key Cryptography (PKC), also known as asymmetric cryptography, is a major cryptographic primitive because it ensures data integrity, confidentiality, and authentication without a common/symmetric secret key. RSA's and ECC's security is based on the mathematical problems of factoring big numbers and elliptic curve logarithm. The relatively new ECC provides reduced computational latency and key sizes compared to RSA, resulting in lower transmission costs for same security levels. As an optimal solution for key agreement and digital signature algorithm, ECC has become the most preferred choice for incorporation into network protocols and cryptographic libraries. Despite ideal memory requirements and computational resources, low-end embedded systems with restricted resources struggle to embrace ECC primitives.

The Elliptic Curve family of schemes consists of curves featuring differentÂ security level, CPU and storage requirements. Yet,Â NIST curves have been a source of concern owing to newly acknowledged security flaws [**?**]. Consequently, a new family of curves was developed addressingÂ the existingÂ issues. Curve25519 and Curve448, as well as their birationally equivalent Edwards curves Ed25519 and Ed448, grew increasing trust byÂ the NIST an are among theÂ recommended curves for building a cryptosystem basedÂ on an ECC primitive since both are suitable for key agreement and digital signature algorithms deployment. In 2018, high security level Curve448 (224 bits) was incorporated into the TLS 1.3 [**?**] network protocol, leading to their incorporation into cryptographic libraries. This work focuses on high-security Curve448 and Ed448, side-channel analysis protection, and wolfSSLCrypt. We also perform different experiment based on the wolfSSL TLS 1.3 implementation based on a USART server-client communication and report our findings before and after protecting the design.

To minimize data leakage through physical behavior and enable Side-Channel Analysis (SCA), cryptographic system design is extremely demanding for embedded devices. Resource consumption may allow an attacker to reveal confidential information. In addition to time, electricity use may expose secret information. Simple Power Analysis (SPA) can reveal key-dependent power consumption disparities in a cryptographic method. Differential Power Analysis (DPA) shows a more complex link between hidden bits and power consumption than SPA, which may be eliminated by equalizing power usage independent of input key value. Therefore, further precautions should be considered to protect the code implementation against side-channel vulnerabilities. This work presents a side-channel resistant design of Curve448 and Ed448 that includes the essential countermeasures at the cost of execution overhead, which is reported for both cryptographic primitives and the TLSv1.3 protocol execution. To prove that our countermeasures secure our code, we offer Test Vector Leakage Assessment (TVLA) statistics and compare the unprotected design to the side-channel resilient implementation.

A. Related Work

Cryptographic primitives, that rely on complex mathematical problems, require a significant amount of processing power on low-end devices; hence, several efficient ECC-based architectures are presentedÂ in the literature, along with the adoption of side-channel attack countermeasures. Being the ultimate objective of an optimum and safe implementation design, its incorporation into cryptographic providers is the objective of several academic and industry groups.

The interest in the relatively new Curve25519 proposed by *Bernstein* in [?], and later Curve448 proposed by *Hamburg* in [?] derives mostly from its low execution latency and high security level, respectively. Both curves are useful for key agreement Elliptic Curve Diffie-Hellman (ECDH), where their birationally equivalent (un)twisted Edwards curves Ed25519 and Ed448 aid in the execution of the Edwards curve Digital Signature Algorithm (EdDSA) [?]. Due to their integration into theÂ TLS 1.3 network protocol standard, theyÂ are already incorporated most cryptographic libraries and have been the subject of optimization and security review.

Elliptic Curve Cryptography's pyramid structure, in which each layer comprises of numerous invocations of a lower layer primitive, enables ECC optimization at many levels. Different algorithms presented in the literature include either low execution delay (Window method [?], Signed Comb [?] method), constant time performance (Double-and-Add-always, Montgomery Ladder [?] method), or compact code size (Double-and-Add [?] method). We base out study onÂ a Montgomery Ladder scalar multiplication implementation design sinceÂ it prevents time and SPA attacks and appears to be particularly effective in eliminating Y coordinate operations, allowing us to operate solely on the X coordinates.

Optimizations and enhancements were likewiseÂ implemented forÂ the ECC's bottom layer. When constructing target-specific hand-crafted assembly code, finite field arithmetic allows for considerable performance improvements. The literature provides numerous multi-precision arithmetic techniques aimed at increasing the efficiency of the algorithms.In [?], *Seo* provides an efficient design for the 8-bit AVR and 16-bit MSP platforms, which, to the best of our knowledge, is the first low-end target-specific design for Curve448 arithmetic architecture based on Karatsuba multi-precision multiplication technique. It is worth noting that finite field arithmetic operations are not only exclusive to classical schemes, they also provide the foundation of the isogeny-based post-quantum primitive. As a result, low level pyramid layer optimizations may be applied to various algorithms and adopted to their prime field length. *Hutter et al.* introduced the implementation ofÂ Operand Caching (OC) in [?], where later its variants Consecutive- and Refined-OC are presented by *Seo et al.* in [?], [?] and [?]. The first customized Curve448 design for the Cortex-M4 ARM CPU is published in [?] by *Seo et al.,* where the authors deploy the R-OC technique. The first EdDSA Ed448 deployment on Cortex-M4 is presented in [?] by *Anastasova et al*. Future multi-precision arithmetic solutions are presented by *Anastasova et al*. in [?]. Recently, the same team presented a new speed record for the Curve448-based key agreement and the Ed448 based digital signature algorithm [?], which we use as a starting point for our work aiming at analyzing and protecting the time optimal design while integrating it into the cryptographic library wolfSSL.

The evaluation of ECC algorithms against side-channel attacks is critical, and it has been a matter of study since 1999, when Kocher introduced the notion of [?]. Current implementations of elliptic curve techniques focus on removing physical behavior that reveals the secret value through the deployment of a series of side-channel attack countermeasures. Point randomization [?] and scalar blinding are two of the most effective as they ensure constant-time and secret-independent calculations. The use of DPA countermeasures, on the other hand, tends to increase the execution time of ECC schemes, as documented for special form such asÂ curvesÂ Curve25519 [?], [?], [?], [?] and FourQ [?]. Curve448 has not been well studied on our target platform, thus we focused on protecting the designs of Curve448 and Ed448 on Cortex-M4 by employing point randomization and scalar blinding countermeasures.

B. Contributions

In this paper, we integrate a highly efficient and side-channel protected architecture for Curve448- and Ed448-based key derivation and digital signature algorithms into the embedded device-specific wolfSSL cryptographic library. Our contributions are as follows:

1) We examine the most recent multi-precision multiplication and squaring approaches designed for the ARM-based Cortex-M4 architecture forÂ Curve448 and Ed448 ECC primitives. We compare performance gains to prior research using the NIST-recommended STM32F407-DK and STM32F413-DK, with an inbuilt WiFi module,Â microcontrollers.
2) We present a side-channel-resistant design for Curve448 and Ed448 by including Differential Power Analysis (DPA)Â countermeasures. After employing point randomization and scalar blinding, we obtain a side channel-resistant architecture at $\sim 1.38\times$ execution time for Curve448 ECDH and $\sim 1.37\times$ and $\sim 1.22\times$ performance increase for Ed448 key generation and signature.
3) We confirm the efficacy of the countermeasures by supplying TVLA traces. We provide the outcomes of a practical method for detecting leaks in an unprotected design.
4) We implement the Curve448 and Ed448 primitives into the wolfSSL cryptographic library and report an improved performance of $\sim 0.88\times$ time the original TLS 1.3 client, as compared to the unprotected architecture, where we benchmark the timing results via a USART connection among the embedded device and a computer. We conduct our experiment using the STM32F413-DK board.

The remaining sections are grouped as follows. In **??** we discuss the mathematical problems, base of X448 and Ed448 DSA algorithms, present the platform specifications, and resume the side-channel analysis considerations. **??** presents the multi-precision arithmetic architecture for Curve448 and Ed448. **??** shows the main features of the emerging TLS 1.3 protocol and the optimal wolfSSL cryptographic library. In **??** we show the side-channel analysis setup and countermeasures, the basis of the applied TVLA leakage detection method, and the obtained results after evaluating our protected design. **??** shows the integration of our design into wolfSSL and the performance improvements as a standalone primitive execution inside the library and as part of the TLS 1.3 network protocol. Finally, we conclude our work in **??**.

Algorithm 1 Montgomery ladder

```
Input: $P = (X_P : Z_P)$, $k = \sum_{i=0}^{l-1} k_i 2^i$ where $k_{l-1} = 1$
Output: $R = k \cdot P$
 1: $R \leftarrow (X_R, Z_R) = (1, 0)$
 2: $Q \leftarrow (X_Q, Z_Q) = (X_P, 1)$
 3: for ($i = 447$; $i >= 0$; $i - -$) do
 4:     if $k_i = 0$ then
 5:         $(R, Q) = ladderstep(X_P, R, Q)$
 6:     else
 7:         $(Q, R) = ladderstep(X_P, Q, R)$
 8:         end if
 9:     end for
10: return $x_R = X_R/Z_R$
```

II. Preliminaries

This section reviews the mathematical background of ECC and in particular the Montgomery and Edwards representation of Curve448–Goldilocks. Finally, we summarize the target specifications and the side-channel analysis evaluation and considered countermeasures.

A. ECC Mathematical Background

Untwisted Edwards Curve448–Goldilocks was proposed by *Hamburg* in [**?**] and shortly after was chosen as a NIST recommendation for ECC instantiation and is claimed to be preferred over NIST curves. Edwards curve are defined as: $E_{Ed}/\mathbb{F}_p : ax^2 + y^2 = 1 + dx^2y^2$ where Curve448–Goldilocks features the parameter set of $d = -39081$ and $a = 1$ and prime number $p = 2^{448} - 2^{224} - 1$ defining the finite field $\mathbb{F}_p$. Edwards and Montgomery curves are birationally equivalent, thus Curve448–Goldilocks can be described as: $E_M/\mathbb{F}_p : v^2 \equiv u^3 + 156326u^2 + u$. This allows to simplify and optimize the implementation of scalar multiplication by dropping the Edwards representation and execute a Montgomery Ladder.

Montgomery Ladder **??** is an efficient constant time algorithm for computing the group operation point multiplication $P = [k] \cdot Q$ where the secret value is decomposed and processed in a bit-by-bit fashion. Opposite to other point multiplication algorithms, Montgomery Ladder prevents SPA due to the execution of point doubling and point addition independently of the processed bit value. Besides timing and SPA, when adapting adequate countermeasures such as scalar blinding and point randomization to eliminate the data dependency in the swap step, the Montgomery Ladder algorithm becomes robust agains DPA.

Additional advantage of applying Montgomery Ladder is the reduced computational latency based on projective coordinates point representation and $X-$only formula. The mapping between projective and affine coordinates consists of $x, y = (X \cdot Z^{-1}, Y \cdot Z^{-1})$, which is performed at the end of the Montgomery Ladder execution. Thus, Montgomery Ladder, efficient and resistant to side-channel analysis attacks, is a preferred choice in many ECC implementation architectures and presents the base of this work.

Algorithm 2 X448 algorithm. G represents the value of the base point

Alice	**Bob**
Input: -	***Input:*** -
Output: sk_A, pk_A	***Output:*** sk_B, pk_B
$sk_A \in_R \mathbb{Z}/\mathbb{F}_p$	$sk_B \in_R \mathbb{Z}/\mathbb{F}_p$
$pk_A = [sk_A] \cdot G$	$pk_B = [sk_B] \cdot G$
Input: sk_A, pk_B	***Input:*** sk_B, pk_A
Output: ss_A	***Output:*** ss_B
$ss_A = [sk_A] \cdot pk_B$	$ss_B = [sk_B] \cdot pk_A$
$ss_A = [sk_A] \cdot sk_B \cdot G$	$ss_B = [sk_B] \cdot sk_A \cdot G$

B. X448 and Ed448

Elliptic Curve Diffie-Hellman and Edwards curve Digital Signature Algorithm have as a core operation the point multiplication, where the signature integrates additionally hash functions due to the nature of the arbitrary length message authentication.

The sole operation for the performance of ECDH is scalar multiplication where both communication parties need two invocations of the so called X448 function. At the end of the execution of the entire algorithm **??**, both parties should reach a common shared secret $ss = sk_A \cdot sk_B \cdot G = sk_B \cdot sk_A \cdot G$, which upon success would allow a symmetric key derivation and the application of an efficient encryption cipher.

The key agreement, however, cannot ensure the authenticity of the communication parties. Therefore, in network protocols, an additional cryptographic algorithm is needed, referred to as digital signature. Similar to real-life signatures, the sender should place a unique sign which will allow the recipient to authenticate the addresser of the message. The EdDSA consist of three main functions - Key Generation, Sign and Verify **??**. The execution of key generation is similar to the ECDH step, integrating a deterministic random number generation based on an additional hashing function. Based on the variable (and unbounded) length of the message being transmitted, the signature involves a hash function. At the end of the algorithm, the verifier obtains a true or false output, depending on the success of the authentication.

C. Target Architecture

ARM embedded devices are a target platform for experimental setup and performance assessment due to their high deployment rate in real-time IoT systems owing to their low power and energy consumption. Instruction pipelining without data dependencies or structural hazard stalls is possible using the Reduced Instruction Set Computer (RISC) architecture. For cryptographic algorithm evaluation, NIST recommends the low-end STM32F407VG microcontroller based on Cortex-M4. Using the provided platform, this study reports side-channel analysis countermeasures. Due to the network protocol focus of this work, particularly the wolfSSL cryptographic library and the TLS 1.3 protocol, we choose the Cortex-M4 ARM platform STM32F413-DK, which has a WiFi module for easy network connection and adoption into IoT real-time systems.

Algorithm 3 Ed448 algorithm [?]. H denotes $SHAKE256$. L represents the order of Ed448 curve. G represents the value of the base point

Key Generation

Input: $seed$

Output: $(p, s), pk_A$

$sk_A \in_R^{seed} \mathbb{Z}/\mathbb{F}_p$

$(p, s) \leftarrow H(sk_A)$

$pk_A \leftarrow encode([s] \cdot G)$

Sign

Input: $pk_A, (p, s), M$

Output: $sign \equiv R||S$

$r \leftarrow (H(p||M))(modL)$

$R \leftarrow encode([r] \cdot G)$

$k \leftarrow (H(R||pk_A||M))(modL)$

$S \leftarrow encode((r + k * s)(modL))$

Verification

Input: $pk_A, M, R||S$

Output: $[S] \cdot G == R + [k] \cdot A$

$k \leftarrow H(R||pk_A||M)(modL)$

$A \leftarrow decode(pk_A)$

ARMv7-M 32-bit architecture features 16 General-Purpose Registers (GPRs) `R0-R15` and optionally another 32 32-bit Floating-Point Registers (FPRs) S0-S31. The optimal implementation design relies not only on efficient and scheduled register utilization but also on Multiply ACcumulate (MAC) instruction which perform long accumulative multiplication in a single clock cycle [CC].

The CPU fetches an instruction every cycle where a stall is only produced due to data dependencies while accessing the memory when proper instruction scheduling is missing.

D. Side-Channel Countermeasures and TVLA Analysis

Constant execution time is a primary anti-SPA countermeasure since the system should not show relationship between the secret value being processed and the physical behavior. More complex analysis, such as DPA, might be used to derive a correlation between these two. To ensure that no data is leaked, a set of countermeasures must be added to a cryptographic architecture. In this paper, we use two countermeasures and demonstrate via TVLA that they eliminate the power consumptionÂ dependency with theÂ secret input value.

Point randomization is a DPA countermeasure approach that allows the scalar value to be protected when performing point multiplication. The approach masks the coordinates of the static base point G usingÂ a randomly generated value λ which is bounded by the bit-length of the prime number defining the finite field. The 448-bit λ is then multiplied by the coordinates of the point, where after converting into projective coordinates the base point becomes defined as $G_r = (\lambda \cdot x_p, \lambda)$. After executing the scalar multiplication and obtaining the result, a conversion back to affine representation is executed. In this step the value of λ is being reduced while retrieving the $x_p = X \cdot Z^{-1} = X/Z = \lambda X/\lambda Z$.

Scalar Blinding is another DPA countermeasure that requires the generation of a random number r to conceal the value of the secret scalar. This technique relies on the fact that base point G added to itself l times, where l is the group order, results in the point at infinity: $l \cdot G = \mathcal{O}$. Thus, any multiple of l, e.g. $r \cdot l$, will also end up at $\mathcal{O}$. In particular, when multiplying the value of $r \cdot l$ and adding it to the secret scalar, the resulting point R will remain the same such as $R = (sk + r \cdot l) \cdot G = sk \cdot G + r \cdot l \cdot G = sk \cdot G + \mathcal{O} = sk \cdot G$.

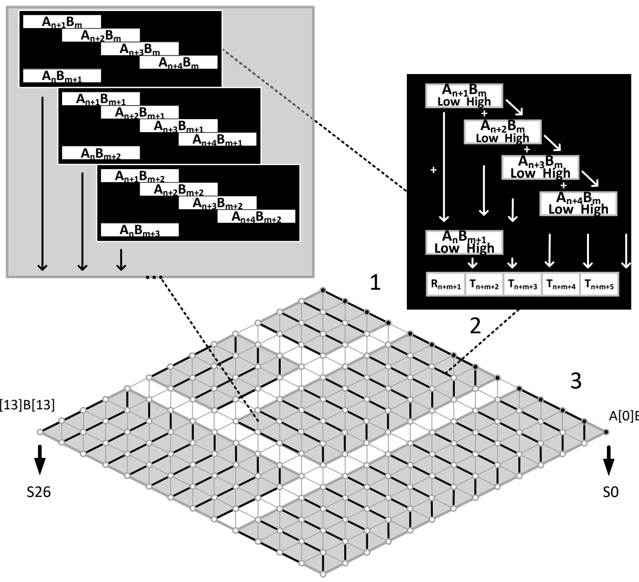


Figure 1. Hybrid^2 architecture for 448-bit multi-precision multiplication. Black lines denote inner loop execution flow. Each black box presents the instructions executed per black line. The white boxes define the instructions executed per white dot.

Algorithm 4 Multi-precision multiplication inner loop execution flow. The horizontal space denotes the PS-like execution flow and the vertical UMAAL instructions show OS-fashion execution flow.

```
VMOV R0, S12                              // R12
UMAAL R0, R10, R2, R6                     // a6b6
    UMAAL R11, R10, R3, R6                // a7b6
      UMAAL R12, R10, R4, R6              // a8b6
         UMAAL R14, R10, R5, R6           // a9b6
LDR R7, [R8, #4*7]                        // b7
UMAAL R0, R9, R1, R7                      // a5b7
VMOV S12, R0                              // R12
```

The recommended value for the blinding factor is around half of the secret scalar bit length $\sim |sk|/2$.

To ensure that our countermeasure design is effective and, in deed, prevents DPA attacks, we use Test Vector Leakage Assessment (TVLA) leak detection mechanism. We perform different measurements and report the graphs based on 10,000 traces.

III. Field Arithmetic Architecture

This work is based on the latest finite field architecture for Curve448 arithmetic targeting Cortex-M4 [?]. The authors of the paper show significant optimization of the X448 and Ed448 performance results by proposing a new multi-precision multiplication and squaring functions.

A. Multi-precision Multiplication

The first mixed multi-precision multiplication is the hybrid variant [?] where the inner loop deploys PS and the outer processes in an OS-like fashion. Later, more optimal variants have been presented in the literature such as the Operand Caching (OC) [?], where the inner loop execution flow changes to

Algorithm 5 Multi-precision squaring inner loop execution flow. The horizontal space denotes the PS-like execution flow.

```
VMOV R0, S12                          // $R_{12}$
LDR R7, [R0, #4*12]                   // $a_{12}$
ADCS R7, R7, R7                       // $2a_{12}$
UMAAL R10, R8, R1, R7                 // $2a_0a_{12}$
   UMAAL R12, R8, R2, R7              // $2a_1a_{12}$
      UMAAL R9, R8, R3, R7            // $2a_2a_{12}$
         UMAAL R14, R8, R4, R7        // $2a_3a_{12}$
            UMAAL R11, R8, R5, R7     // $2a_4a_{12}$
VMOV S12, R10                         // $R_{12}$
```

ensure maximum utilization of the loaded operand limbs. A Consecutive- and Refined-OC (R-OC) [**?**], [**?**] were suggested in the literature again modifying the execution flow for more optimal results.

The first implementation design to combine both multi-precision multiplication strategies inside the scope of the inner loop is the one presented by *Anastasova et al.* in [**?**].

The multi-precision multiplication of this double-hybrid (or hybrid^2) strategy (double since it applies hybrid design - R-OC, first to the entire multiplication, similar to previous implementations, and second to the inner multiplication loop) relies on the idea of boosting the inner multiplication loop by increasing the row size (i.e., the number of accumulatively computed 32×32-bit multiplications in each iteration of the inner loop).

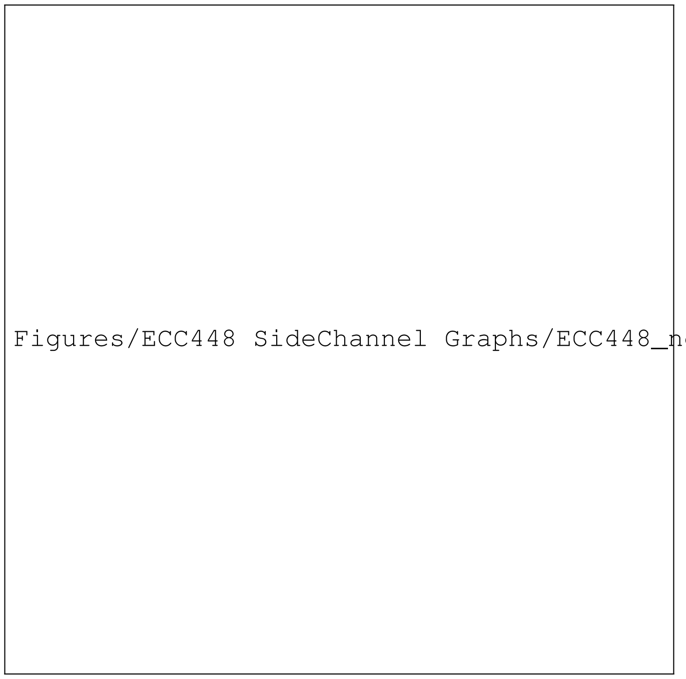
Visual representation of the described steps is presented in **??** where the rows are highlighted in grey color and the inner loop execution flow is denoted by a bolded back line, where each line dotes a single iteration of the loop. The execution flow and the instruction scheduling per inner loop iteration is presented in **??**. A more detailed representation of the computational execution flow is shown in the upper side of **??**, where each one of the $k+1$, with $k = 4$, 32×32-bit multiplication per inner loop iteration are shown.

B. Multi-precision square

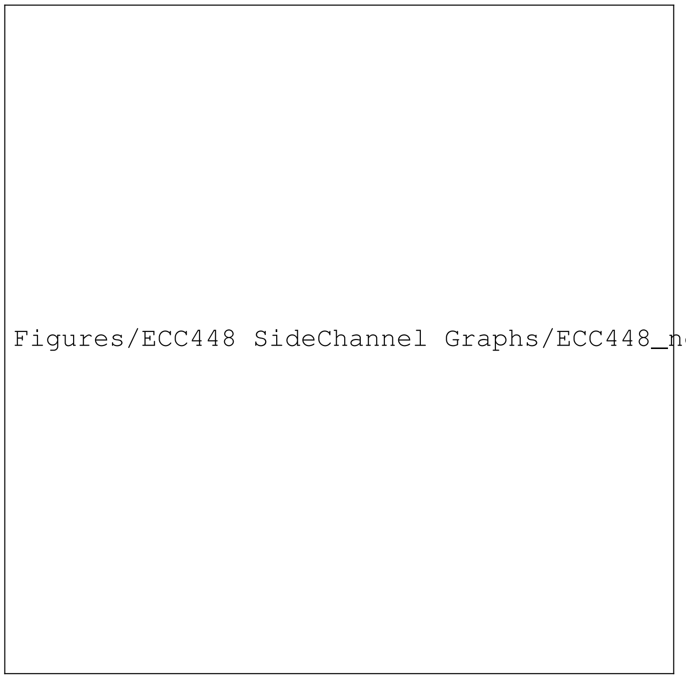
Multi-precision squaring is also a fundamental building component for ECC calculation. It has similar qualities to multiplication; however, when a number is multiplied by itself, the limbs of both operands coincide. As a consequence, many of the 32×32-bit multiplications may be removed simply by doubling the results (i.e., the accumulative multiplication of limb n and m of the operand, when $m = n$, is going to the the double of only one of the multiplications, or shifting right by 1). As a result, squaring may be implemented at a far lower cost than multi-precision multiplication.

The authors of [**?**] offer the first and, to our knowledge, quickest multi-precision squaring design. The architecture is built on a process similar to Product Scanning. However, they combine it with the notion of Refined-OC for execution flow backward, therefore their design still includes the curve in the square's *midpoint*.

The design mixes two separate row-types, one of which is implemented in a sub-squaring way and the other in a sub-multiplication form, similar to [**?**]. The sub-squaring blocks



(a) TVLA t-test.



(b) Magnified t-test values for 100,000 samples.

Figure 2. TVLA graphs showing data leak for the unprotected Montgomery Ladder execution using 10,000 traces.

Figures/ECC448 SideChannel Graphs/ECC448_pointrandom/1st-order-tvla-10000-pointrandom-eps-conv

Figure 3. TVLA graphs showing data leak for the protected Montgomery Ladder design based on point randomization DPA countermeasure using 10,000 traces.

Figures/ECC448 SideChannel Graphs/ecc-448-SB-10000-eps-converted-to.pdf

Figure 4. TVLA graphs showing data leak for the protected Montgomery Ladder design based on scalar blinding DPA countermeasure using 10,000 traces.

Figures/ECC448 SideChannel Graphs/ecc-448-both-10

(a) TVLA t-test.

Figures/ECC448 SideChannel Graphs/ECC448_scalarbl

(b) Magnified t-test values for 100,000 samples.

Figure 5. TVLA graphs showing data leak for the protected Montgomery Ladder design based on point randomization and scalar blinding DPA countermeasure using 10,000 traces.

Table I
CURVE25519 AND CURVE448 ECDH AND EDDSA SCA UNPROTECTED VS. PROTECTED IMPLEMENTATIONS [KCC]

Work	Freq. [MHz]	X448	Ed448 KeyGen	Ed448 Sign	Ed448 Verify	Protected
FourQ[1]		543	-	-	-	U
Curve25519[2]	84	894	390	544	1,331	U
Curve25519[3]	16	625	-	-	-	U
	168	656	-	-	-	
Curve25519[4]	24	2,339	-	-	-	F
Curve448[5]	24	6,218	-	-	-	UU
	168	6,286	-	-	-	
Ed448[6]	24	-	4,069	6,571	8,452	U
	168	-	4,195	6,699	8,659	
Curve448[7]	24	3,221	3,536	6,038	7,404	U
	168	3,975	4,282	6,787	8,854	
This work	24	**3,503**	**3,826**	**6,328**	**7,404**	PR
		4,151	**4,510**	**7,012**	**7,404**	SB
		4,465	**4,841**	**7,343**	**7,404**	F
	168	**4,344**	**4,669**	**7,173**	**8,854**	PR
		5,128	**5,472**	**7,975**	**8,854**	SB
		5,538	**5,913**	**8,417**	**8,854**	F

Refer to:[1] [?],[2] [?],[3] [?],[4] [?], [5] [?], [6] [?], [7] [?]

are placed at the start and end of each row and continue until the row reaches its maximum length. Following that, the sub-multiplication block is used, and the result is added to the sub-squaring block result.

The implementation of multi-precision multiplication and squaring in [?] shows significant speed record and, thus, is focus of this work adding the required SCA countermeasures and benchmarking into wolfSSL TLS 1.3.

IV. TLS 1.3 AND WOLFSSL

Transport Layer Security (TLS) is the most frequently used network protocol for creating secure communication, and it is implemented and supported by every major cryptographic library. The protocol's widespread adoption opened gap for many attacks to the TLS1.2. The Internet Engineering Task Force (IETF) standardized the next 1.3 version [?] of the protocol in 2018, which modified the execution paradigm, increased security, modularity, and execution speed.

Thus, most cryptographic libraries provide TLS 1.3 which simplifies and cleans up the existing TLS 1.2 version, boosting security and implementation strength while drastically reducing computational and communicational delay due to the one round trip required for the full handshake. TLS 1.3, most crucially in the context of this study, supports Curve448 and Ed44.

Key exchange, server parameters, and authentication are the three steps of TLS 1.3, sometimes known as the TLS handshake. During the key exchange, the client sends a `ClientHello` message with session and feature information. The server learns the highest available TLS version from \texttt{ClientHello}. Finally, the client offers a list of supported preferred ciphersuites. It's important to note that the server can bypass ciphersuites if they are not preferred and proceed with others.

The server sends a `ServerHello` message with a nonce and legacy version. This message's ciphersuite, which encrypts transmission, is very important. Extensions include server name, supported groups, signature techniques and certificates, CAs, pre-shared keys, etc. TLS 1.3 encrypts anything after the `ClientHello`. Thus, some extensions send their Server Parameters phase data in ciphertext.

The `ServerHello` message initiates authentication by transmitting its own certificate and possibly requesting client authentication. It's worth noting that the server's signature covers the entire handshake message set, not just the certificate. This stage sends a completion message, which includes a MAC of the complete data length, giving key confirmation or authentication in PSK mode, and the application data may begin to be sent from server to client. The client completes the authentication by checking the server's certificate and, if necessary, submitting its own. Following that, a completion message is sent, and both communication parties can safely share application data.

WolfSSL is a popular cryptography library for low-end embedded devices with limited computation, memory, battery life, and bandwidth. Due to its portable C implementations, wolfSSL reduces execution delay and optimizes code size. Client programs benefit from the library's straightforward APIs, rich documentation, and current crypto primitives. WolfSSL supports TLS 1.3 and experimental post-quantum and hybrid ciphersuites. Other efforts [?] have improved cryptographic algorithm time (Curve25519) and side-channel safety for low-end devices.

This article is the first to combine side-channel protection, TVLA analysis of countermeasure efficacy, and handcoded ARMv7 assembly implementation of Curve448 arithmetic. We propose secure Curve448 and Ed448 at a lower cost than the original wolfSSL design, but at the risk of non-portable platform specific assembly implementation.

V. SCA COUNTERMEASURES, TVLA AND PERFORMANCE IMPACT

Side-channel analysis (SCA) is based on observing a relation between physical behavior of a system and the secret value. Data leak may be produced by non uniform execution time, power consumption, or electromagnetic emissions. Based on the processor resource utilization, a malicious party could recover secret information about a communication party. Therefore, careful analysis should be performed even when constant time implementation is promised.

A. Setup

This study examines the latest Montgomery Ladder-based implementation of the Curve448 key agreement and Ed448 digital signature technology, which use constant-time multi-precision multiplication and squaring. We collected power use data and used TVLA based on t-statistic to assess distinguishability to study the implementation design and potential DPA threats. Welch's t-test calculates a t-statistic from TVLA traces' mean and standard deviation, where a threshold indicates information leakage. We cautiously adjusted the cutoff value at 6 based on [?] and [?] to reduce false positive values.

For the setup of the system we use NewAE CW308T-STM32F board, which features the target Cortex-M4 platform, along with NewAE CW308 UFO. The configuration is linked to a NewAE Chipwhisperer Lite board, which allows the target

Table II
CURVE448 ECDH AND EDDSA SCA UNPROTECTED VS. PROTECTED IMPLEMENTATIONS IN WOLFSSL BENCHMARK TEST AND AS PART OF THE TLS 1.3 HANDSHAKE.

Work	Operation	Curve448 ECDH		Ed448DSA			Protected
		keygen	agree	keygen	sign	verify	
wolfSSL[1]	ops	3	4	6	6	2	U
	sec	1.278	1.706	1.071	1.142	1.020	
	avg ms	426.000	426.500	178.500	190.333	510.000	
	ops/sec	2.347	2.345	5.602	5.254	1.961	
	TLS 1.3 Client	3.411987 [ms] / 57911551 [CCs]					
Curve448[2]	ops	5	6	5	6	2	U
	sec	1.051	1.255	1.063	1.491	1.141	
	avg ms	210.200	209.167	212.600	248.500	570.500	
	ops/sec	4.757	4.781	4.704	4.024	1.753	
	TLS 1.3 Client	2.996094 [ms] / 50802387 [CCs]					
This work	ops	5	6	5	4	2	PR
	sec	1.146	1.365	1.153	1.067	1.142	
	avg ms	229.200	227.500	230.600	266.750	571.000	
	ops/sec	4.363	4.396	4.337	3.749	1.751	
	TLS 1.3 Client	3.043091 [ms] / 51524459 [CCs]					
This work	ops	4	4	4	4	2	SB
	sec	1.012	1.008	1.020	1.165	1.149	
	avg ms	253.000	252.000	255.000	291.250	574.500	
	ops/sec	3.953	3.968	3.922	3.433	1.741	
	TLS 1.3 Client	3.067017 [ms] / 51940135 [CCs]					
This work	ops	4	4	4	4	2	F
	sec	1.086	1.082	1.094	1.236	1.150	
	avg ms	271.500	270.500	273.500	309.000	575.000	
	ops/sec	3.683	3.697	3.656	3.236	1.739	
	TLS 1.3 Client	3.110107 [ms] / 52738762 [CCs]					

Refer to:[1] [?],[2] [?]

ARM platform to interact with the PC. The test results are based on USB3 oscilloscope Picoscope 3000. To carefully measure the implementation, we operate the target board at 25MHz. To guarantee that the traces are indistinguishable, we randomly choose between using a fixed input scalar value or a random scalar.

When the implementation is unprotected, as seen in [?], we publish the gathered TVLA graphs based on the t-test in **??**. It is simple to discover that the obtained traces indicate data leak. We gather another 10,000 traces after applying the point randomization SCA countermeasure and display the TVLA result in **??**. The data leak is visually reduced, but not eliminated; so, this single countermeasure is insufficient to secure the design. We incorporate scalar blinding protection into the design and publish the TVLA findings in **??**, where, as with point randomization, the effort is insufficient to guarantee the user a SCA protected implementation, however, it is noticeable that the design is better protected. Finally, we provide the gathered data base on the integrated countermeasure design **??**, which enables full SCA protected architecture. As can be seen, there is no association between data processing and power usage in the observed numbers.

Only after securing the design, we could proceed to the integration of the code into the cryptographic library wolfSSL since deploying code in industry requires exhaustive security analysis of the design.

B. Protected Design Performance

To evaluate the performance impact of our adopted countermeasure, we test our design on the SMT32F407-DK microcontroller running @24MHz in order to provide precise latency eliminating false stalls produces by memory control unit stalls. We also report out results @168MHz in order to provide a real scenario boosting the speed to the maximum board frequency.

We report the obtained results in **??**, where we report other implementation for comparison purposes. We notice that the short prime Curve25519 and FourQ show much more efficient implementation (orders of magnitudes especially when comparing the unprotected design with out protected design). However, we should notice that Curve448 operates on almost double size integers and provides much higher security level. Therefore, this discrepancy is expected.

We observe $\sim 300KCC$, $\sim 900KCC$, and $\sim 1,200KCC$ of execution overhead when considering X448 running @24Mhz applying scalar blinding, point randomization and full SCA countermeasure protection to the design. For Ed448 key generation we observe similar number of clock cycle overhead and $\sim 1.08\times$, $\sim 1.28\times$, and $\sim 1.36\times$ increased performance for the three SCA countermeasure scenarios, respectively. Signing SCA protection comes at a similar cost.

VI. WOLFSSL TLS 1.3 PERFORMANCE EVALUATION

In this work, we report our results after integrating the the unprotected code in the wolfCrypt cryptographic engine, we analyze the performance and compare it with their previous design. Afterwards, we add the proposed countermeasures and again measure the performance.

?? show the benchmarking results when running the wolfSSL test on STM32F413-DK microcontroller. We notice that the results obtained after integrating the Curve448 and Ed448 arithmetic operations from [?] result in reduced latency, thus, allow more operations to execute in the same tame slot. We notice that the X448 performance shows x2 optimized implementation. The execution of Ed448 does not simply rely on point multiplication but rather integrates other cryptographical primitives such as hashing, therefore, is not impacted as much as the Curve448 ECDH computational latency.

After applying point randomization we observe that the design of X448 drops by $\sim 19ms$, in both the key generation and the agreement. Additionally, the Ed448 point randomization also comes at the same cost $\sim 18ms$ for both the key generation and the signing functions. The stand alone adoption of scalar blinding requires around additional $\sim 40ms$ per function. Finally, the fully protected design comes at a relative cost of $\sim 60ms$ per function.

We increase the implementation time of X448, Ed448 key generation and signature with $\sim 1.29\times$ for all the functions. However, we provide protected design for the highly deployed in the IoT world Cortex-M4 platform.

Finally, we report the timing and the number of clock cycles required per TLS 1.3 handshake roundtrip, running the client of the STM32F413-DK board and the server on the PC. We use the USART serial connection as a communication channel, therefore, the results are higher then expected, based on the communication latency.

VII. Conclusion

In this paper, we examine the most recent Curve448 and Ed448 Montgomery Ladder-based architectures for ECDH and EdDSA, with a focus on the Cortex-M4 ARM platform and side-channel analysis attacks. To assess leakage in the unprotectedÂ scheme, we set up an experimental scenario and conductÂ the TVLA test. We secure the Curve448 by using scalar blinding and point randomization DPA countermeasures and analyzing the TVLA findings to confirm that our design is secure. Finally, we incorporate our protected assembly versions of Curve448 and Ed448 into wolfSSL and test their performance as a standalone primitive and as part of the TLS 1.3 protocol.