Efficient Error Detection Cryptographic Architectures Benchmarked on FPGAs for Montgomery Ladder

Kasra Ahmadi, Saeed Aghapour, Mehran Mozaffari Kermani, and Reza Azarderakhsh

***Abstract*— Elliptic curve scalar multiplication (ECSM) is a fundamental element of public key cryptography. The ECSM implementations on deeply embedded architectures and Internet-of-nano-Things have been vulnerable to both permanent and transient errors, as well as fault attacks. Consequently, error detection is crucial. In this work, we present a novel algorithm-level error detection scheme on Montgomery Ladder often used for a number of elliptic curves featuring highly efficient point arithmetic, known as Montgomery curves. Our error detection simulations achieve high error coverage on loop abort and scalar bit flipping fault model using binary tree data structure. Assuming n is the size of the private key, the overhead of our error detection scheme is $O(n)$. Finally, we conduct a benchmark of our proposed error detection scheme on both ARMv8 and field-programmable gate array (FPGA) platforms to illustrate the implementation and resource utilization. Deployed on Cortex-A72 processors, our proposed error detection scheme maintains a clock cycle overhead of less than 5.2%. In addition, integrating our error detection approach into FPGAs, including AMD/Xilinx Zynq Ultrascale+ and Artix Ultrascale+, results in a comparable throughput and less than 2% increase in area compared with the original hardware implementation. We note that we envision using adoptions of the proposed architectures in the postquantum cryptography (PQC) based on elliptic curves.**

***Index Terms*— ARM processor, fault detection, field-programmable gate array (FPGA), Montgomery Ladder, reliability.**

I. Introduction

In 1985, Neal Koblitz and Victor Miller introduced the utilization of elliptic curves within the domain of cryptography [1], [2]. Elliptic curve cryptography (ECC) has attracted considerable interest in the public key cryptographic algorithms because of its ability to use shorter key lengths. Cryptosystems, including those that depend on elliptic curves, have encountered weaknesses and vulnerabilities, making them prone to attacks. As part of ensuring security for ECC implementations, several standards were developed [3], [4], [5], and [6]. Curve25519 and Curve448 [7] have been proposed as viable alternative derivatives of the National Institute of Standards and Technology (NIST) curves [8], as researchers have raised concerns about potential flaws [9].

A. Related Works

Several research works have highlighted the importance of cryptographic applications having resistance to side-channel analysis [10], [11], [12], and [28]. Boneh et al. [14] introduced fault analysis attack which garnered significant attention. This attack relies on inducing transient faults within cryptosystems to disclose sensitive information, e.g., private key. Biehl et al. [15] expanded the application of fault-based attacks to cryptographic systems using elliptic curves. To defend against the mentioned attacks, one can easily confirm that the output lies on a legitimate elliptic curve. Fouque et al. [16] introduced a fault attack targeting the Montgomery Ladder elliptic curve scalar multiplication implementation.

Different operations within finite fields, such as multiplication, addition, inversion, and squaring, can be susceptible to error detection methods. While fault detection techniques have been investigated for cryptographic systems [17], [18], [19], [20], [21], [22], [23], there has been relatively little research dedicated to fault detection specifically at the algorithmic level of Montgomery Ladder elliptic curve scalar multiplication (ECSM). Dominguez-Oviedo and Hasan [24] presented fault detection schemes at the algorithm level for Montgomery Ladder and double-and-add-always scalar multiplication. They considered PV and coherency check (CC) for their proposed error detection scheme. They proposed error detection scheme resist attacks such as safe error (SE) [25] and [26] and sign change fault (SCF) attacks [27].

B. Major Contribution

We have proposed an algorithm-level error detection scheme for Montgomery Ladder in the presence of scalar blinding. The proposed scheme achieves near-perfect fault coverage with minimal hardware and software overhead. This scheme is envisioned for postquantum cryptography (PQC) variants based on elliptic curves with relevant modifications. We implemented the proposed approach on ARMv8 (Cortex-A72) and two field-programmable gate arrays (FPGAs), i.e., Zynq Ultrascale+ and Artix Ultrascale+, using Curve448 parameters. It achieves close to 100% error coverage with low overhead (5.2% clock cycles on ARMv8 and about 2% on FPGAs).

II. Preliminaries

The Montgomery Ladder was first introduced as a scalar multiplication algorithm for a specific type of elliptic curves known for their highly efficient point arithmetic referred to as Montgomery curves. The Montgomery approach also achieves additional acceleration by exclusively calculating projective coordinates (X, Z) of intermediate points. This optimization is feasible because the Montgomery Ladder incorporates a technique called differential addition, which computes the sum of two points with a known difference. The central concept of the algorithm is to simultaneously compute two values that have a P difference. This algorithm is resistant against timing [10] and SPA attacks [11]. The Montgomery Ladder ECSM is described in Algorithm 1.

A. Scalar Blinding

To enhance security against differential power analysis (DPA), it is necessary to incorporate supplementary techniques such as base-point

Manuscript received 17 January 2024; revised 26 March 2024, 15 May 2024, and 5 June 2024; accepted 20 June 2024. Date of publication 5 July 2024; date of current version 24 October 2024. This work was supported by U.S. National Science Foundation (NSF) under Grant SaTC-1801488. *(Corresponding author: Mehran Mozaffari Kermani.)*

Kasra Ahmadi, Saeed Aghapour, and Mehran Mozaffari Kermani are with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: ahmadi1@usf.edu; aghapour@usf.edu; mehran2@usf.edu).

Reza Azarderakhsh is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: razarderakhsh@fau.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2024.3419700.

Digital Object Identifier 10.1109/TVLSI.2024.3419700

Algorithm 1 Montgomery Ladder ECSM

```
Input: $P \in E(\mathbb{F}_q), k_{bits}$(high to low)
Output: $kP$
1: $Q_0 = \mathscr{O}$, $Q_1 = P$
2: for $bit$ in $k_{bits}$:
3:     if $bit = 0$:
4:         $Q_1 = $ point_add$(Q_0, Q_1)$
5:         $Q_0 = $ point_double$(Q_0)$
6:     else:
7:         $Q_0 = $ point_add$(Q_0, Q_1)$
8:         $Q_1 = $ point_double$(Q_1)$
9: return $Q_0$
```

randomization and scalar blinding. Scalar blinding can be achieved by adding multiple group order to k such that $k_r = k + r \times \#E$ where $GF(p)$ is finite field, E is the elliptic curve over $GF(p)$, $P \in E(GF(p))$, and $\#E$ is the cardinality of the group of points $E(GF(p))$. Based on Hesse's theorem, $\#E$ is close to p and bounded by $(\sqrt{p} - 1)^2 \leq \#E \leq (\sqrt{p} + 1)^2$. To generate a random value r, linear feedback shift registers (LFSRs) can be used in simple and resource-constrained applications. For enhanced security purposes, physical unclonable functions (PUFs) are more suitable [28]. The correctness of the approach can be proven as follows:

$$k_r \cdot P = (k + r \times \#E) \cdot P = k \cdot P + r \cdot \mathscr{O} = k \cdot P.$$

III. Error Detection in Montgomery Ladder

In this section, we describe the proposed error detection scheme in Montgomery Ladder ECSM that is described in Algorithm 2.

A binary tree based on the private key (k_{bits}) is created in Line 1 of Algorithm 2. Another binary tree (k') will be created during Montgomery Ladder Algorithm (Lines 6, 7, 11, and 12). It is important to note that each node in the binary tree has only one unique path from the root. Using this guidance, we have assurance that the primary loop of the Montgomery Ladder adheres to the structure of its scalar value.

Our presented error detection scheme relies on the construction and comparison of two binary trees: One generated prior to executing the Montgomery Ladder algorithm, and the other generated during its execution. The primary reason for using the phrase "binary tree" in this context is to illustrate the concept of having two choices, either 0 or 1, at every stage of the process. To simplify matters, we used a binary string instead of binary tree for evaluating the chosen path in the implementation. The arrangement of the binary string is determined by the scalar bit in each stage of the Montgomery Ladder algorithm. Architecture of Error_detection module at Line 15 in Algorithm 2 is depicted in Fig. 1.

As our presented error detection scheme is based on the scalar k, to apply our presented error detection scheme effectively in scenarios involving scalar blinding, we need to incorporate modular reduction based on $\#E$ before comparing k' with k within our scheme. To enhance security in ECC using a specialized prime field like the Solinas prime, it is advised to use larger blinding factors r, which should be at least half the size of the field. Consequently, as our field size is 448-bit, a blinding factor r with a length of 224 bits results in a k_r factor of 672 bits. Our presented error detection scheme is presented in Fig. 1.

A. Fault Model

To the best of our knowledge, any fault injection or error occurrence over P will get covered with the help of PV and specified CC

Algorithm 2 Proposed Error Detection Scheme in Montgomery Ladder ECSM

```
Input: $P \in E(\mathbb{F}_q), k_{bits}$(high to low)
Output: $kP$
1: TreeNode root_pre_computation = build_tree$(k_{bits})$
2: TreeNode $k'$ = create_node()
3: $Q_0 = \mathscr{O}$, $Q_1 = P$
4: for $bit$ in $k_{bits}$:
5:     if $bit = 0$:
6:         $k'$.left = create_node()
7:         $k' = k'$.left()
8:         $Q_1 = $ point_add$(Q_0, Q_1)$
9:         $Q_0 = $ point_double$(Q_0)$
10:    else:
11:        $k'$.right = create_node()
12:        $k' = k'$.right()
13:        $Q_0 = $ point_add$(Q_0, Q_1)$
14:        $Q_1 = $ point_double$(Q_1)$
15: error = Error_detection$(k, k')$
16: if error:
17:     return "Error detected"
18: else:
19:     return $Q_0$
```

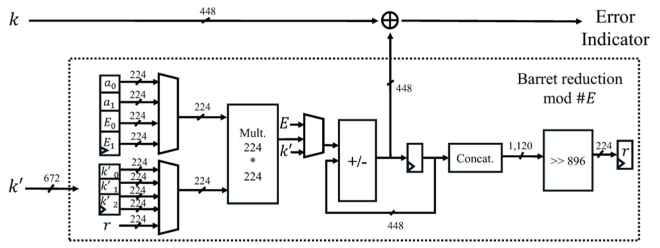

Fig. 1. Proposed error detection module based on Barret reduction.

function [24]. However, the Montgomery Ladder ECSM calculation in Algorithm 2 may face a rather critical compromise which is the occurrence of random or burst errors over scalar (k) or the loop counter. Prior error detection schemes were incapable of identifying errors related to early or extended loop termination or the flipping of scalar bits, which is the focus of our fault model.

B. Practical Scenario

The following example provides details of the proposed fault detection scheme. For simplicity, let the scalar in Algorithm 2 be $k = (1, 0, 1)_2$. In Line 1 of Algorithm 2, the binary tree associated with the scalar value is created. The generated binary tree is depicted in Fig. 2. Another binary tree with the same node generation policy will get produced during the main loop of Algorithm 2. Two binary trees will get compared in the Error_detection function (Line 15) of Algorithm 2 (Fig. 1).

1) Faulty Scalar: Given that our genuine scalar is $k_{\text{genuine}} = (1, 0, 1)_2$, and our faulty scalar is $k_{\text{faulty}} = (1, 0, 0)_2$, the destination nodes of both the binary trees are different. By comparing these destinations, our error detection method can identify the occurrence of an error during Montgomery Ladder ECSM. Fig. 2 depicts the generated nonfaulty and faulty binary trees from faulty scalar, respectively.

2) Extended or Early Loop Termination: In situations where extended or early loop termination occurs, the depths of the two binary trees are not equal, and their respective destination nodes vary.

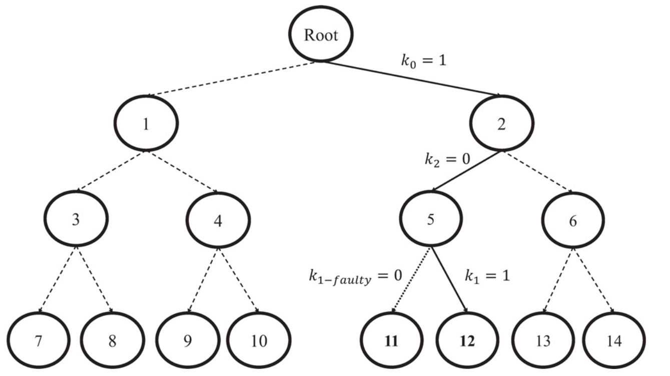

Fig. 2. Binary tree which shows a nonfaulty path using the secret key $k_{\text{genuine}} = (1, 0, 1)_2$ and faulty path using a faulty secret key, $k_{\text{faulty}} = (1, 0, 0)_2$. The nonfaulty path has successfully reached node 12 as its endpoint, while the faulty path, generated using a faulty secret key, reached at node 11.

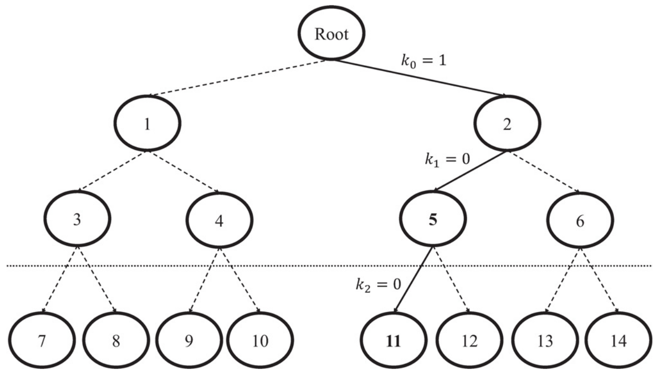

Fig. 3. Binary tree which shows faulty path generated from early loop termination. The binary tree has reached node 5 as its destination from early loop termination. However, nonfaulty destination is node 12.

Through comparison of these destination nodes, our error detection approach can identify the occurrence of an error during the execution of Montgomery Ladder in ECSM architectures. Fig. 3 depicts the generated faulty binary tree from early loop termination.

C. Effectiveness Against Fault Attacks

While previous work has protected against fault attack on changing the public point P, our presented error detection scheme can protect the Montgomery Ladder algorithm from any fault attack which focuses on scalar altering. Our presented error detection scheme can protect from [30] which could achieve the least or most significant bits of the secret scalar by performing differential fault attacks with scalar randomization.

IV. FPGA and ARM Implementation Benchmarks

To show the efficiency and overhead of our proposed error detection method, we assessed its performance by applying our presented error detection scheme to Curve448, which is currently integrated into TLS 1.3. We used projective coordinates formula for point doubling and point addition on ECC curves due to less computational overhead on both hardware and software.

We conducted a benchmark test using the Cortex-A72 processor for software implementation and two different FPGAs, namely, Zynq Ultrascale+ and Artix Ultrascale+, for hardware implementation to assess the additional load introduced by our approach. This analysis demonstrates that our approaches maintain an appropriate overhead while effectively attaining a high error coverage.

By applying our design to ARMv8 and FPGA architectures, we acquired valuable insights into the effectiveness of our schemes on a variety of platforms. The baseline and the proposed error detection scheme used identical optimization and clock settings. The baseline work lacks any error detection scheme. Table I displays the clock overhead of the proposed error detection scheme on software. Table II presents the area, timing, power, and energy derivations of our proposed error detection scheme on two different AMD/Xilinx FPGAs. Our error detection scheme and the baseline work have been synthesized and executed using Xilinx Vivado 2023.1. The provided results are obtained after post place-and-route. In the Vivado synthesis tool context, the notion of area encompasses a combination of Slices and DSPs, with an equivalence ratio of one DSP being equivalent to 100 CLBs.

A. Software Implementation

For software implementation, we selected the Raspberry Pi 4 as our platform for the ARMv8 architecture, featuring four Cortex-A72 cores running at 1.5 GHz each. To assess performance on the ARMv8 architecture, we used the Performance Application Programming Interface (PAPI), a widely recognized framework designed for gauging system performance. We used the GMP library in the C programming language to enable the handling of big integers. The clock overhead of the proposed error detection is about 5% in software.

TABLE I
ARMv8 Implementation Results

	Montgomery Ladder	
Scheme	Our Scheme	Baseline work
Clock cycles	13,427	12,764
Clock overhead[1]	5.2%	-

[1]Clock overhead $= \frac{\text{Approach's clock cycles} - \text{Baseline work's clock cycle}}{\text{Baseline work's clock cycle}} \times 100$

B. Hardware Implementation

In this section, we discuss hardware design of Montgomery Ladder on Curve448. Hardware implementation code written in SystemVerilog is available in our GitHub account.[1] A crucial requirement in elliptic curve systems is the effective execution of finite field arithmetic. Curve448 over $GF(p)$ is defined by $y^2 + x^2 \equiv 1 - 39081x^2y^2 \bmod p$ where $p = 2^{448} - 2^{224} - 1$. The proposed Baseline Montgomery Ladder over Curve448 is shown in Fig. 4. The design includes modular multiplication, data bus, modular addition/subtraction, controller, and RAM in the top-level architecture. As operands in Curve448 are 448 bits, due to IO buffer limitation on our chosen FPGA boards we chose the data bus of width 56 bits which needs eight clock cycles to load each operand.

1) Modular Multiplication: Based on the fact that $2^{448} \equiv 2^{224} + 1 \bmod p$, we used 224-bit multiplication for modular multiplication as follows:

$$\begin{aligned} c = a \cdot b &= (a_1\phi + a_0) \cdot (b_1\phi + b_0) \\ &= (a_1b_1 + a_0b_0) + (a_1b_0 + a_0b_1 + a_0b_0)\,\phi \pmod p \end{aligned}$$

where $\phi = 2^{224}$ and $a, b, c \in GF(p)$. We converted the 448×448-bit multiplication to three 224×224-bit multiplications respecting the carry. The multiplication postprocess module in Fig. 4 uses interleaved fast reduction for p based on [29].

[1]github.com/KasraAhmadi/Montgomery_Curve448_error_detection

TABLE II
AMD/XILINX ZYNQ ULTRASCALE+ AND AMD/XILINX ARTIX ULTRASCALE+ FPGA IMPLEMENTATION RESULTS

Platform		AMD/Xilinx Zynq Ultrascale+ xczu4ev-sfvc784-2-i		AMD/Xilinx Artix Ultrascale+ xcau10p-ubva368-2-e	
Scheme		Our scheme	Baseline Montgomery Ladder	Our scheme	Baseline Montgomery Ladder
Area	LUTs	19,014	17,247	19,349	17,549
	FFs	18,412	17,962	18,411	17,961
	DSPs	169	169	169	169
	CLBs	3,814	3,434	3,766	3,324
	Overall Converted Area[1]	20,714 (+380)	20,334	20,666 (+442)	20,224
Power (W) @ 50 MHZ		0.44 (+0.01)	0.43	0.33 (+0.02)	0.31
Timing	Latency [Clock Cycles]	13,120 (+56)	13,064	13,120 (+56)	13,064
	Total time [ns]	262,400 (+1,120)	261,280	262,400 (+1,120)	261,280
Energy (nJ)		115,456 (+3,106)	112,350	86,592 (+5,596)	80,996

[1]Overall Converted Area = CLBs + DSPs × 100

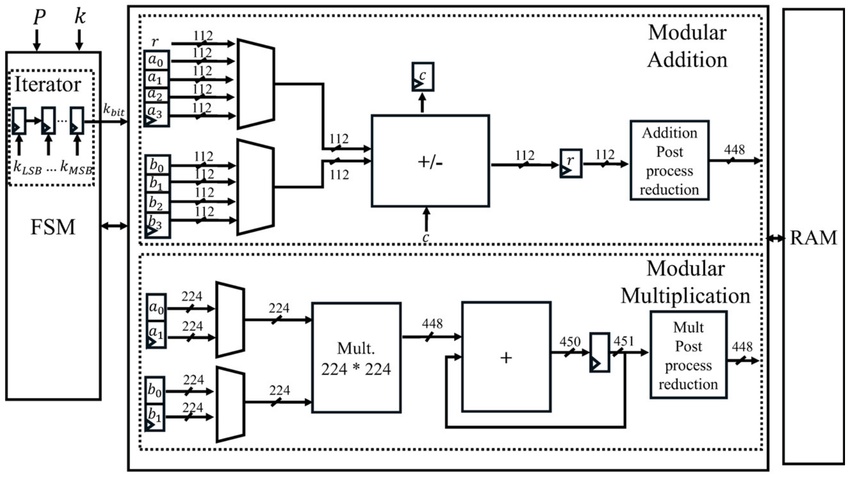

Fig. 4. Baseline Montgomery Ladder over Curve448.

2) Modular Addition/Subtraction: Addition/subtraction module is implemented by two 112-bit data using pipeline addition/subtraction. This module calculates $c = a \pm b \pm p$. The sddition postprocess module in Fig. 4 calculates $c = r \pm p$ based on the comparison between r and p.

3) Error Detection Scheme Overhead: In the existence of scalar blinding that is discussed in Section II-A, the constructed binary string (k') is of 672-bit length. As shown in Fig. 1, we used a Barret reduction module to reduce k' before comparing it to k. The Barret reduction module operation is based on two multiplications, one shifting that is free in hardware, and two subtractions. One multiplication performs on 672-bit by 448-bit with 34 clock cycles, one multiplication on 224-bit by 224-bit with 10 clock cycles, and two subtractions that each needs six clock cycles. The error detection scheme requires a total of 56 clock cycles to complete. To save the area and reuse the existing modules, we convert all the operands into 224-bit segments to use the DSP modules already used in modular multiplication as shown in Fig. 4. Furthermore, for subtraction we used the same 112-bit by 112-bit addition/subtraction component used in modular addition in Fig. 4. The extra CLB utilization detailed in Table II refers to additional registers and logic elements, e.g., XOR gates, which are used in building the error detection scheme depicted in Fig. 1.

V. COMPARISON WITH PREVIOUS STUDIES

Our error detection scheme encompasses not previously mentioned fault models and incurs much lower computational overhead in both software and hardware compared with previous methods. We reference our work against the prior study [24]. In terms of fault coverage, our research provides an advantage by detecting faults that cause premature termination or loop extension, a capability that was missing in earlier investigations. In terms of safeguarding against fault attacks, our study concentrated on fault attacks that aimed to alter scalars rather than altering the base point [30]. Finally, in terms of overhead, the prior research analyzed the Montgomery Ladder algorithm using projective coordinates. The overhead is quantified based on the number of finite field operations rather than hardware or software utilization. This analysis yielded a cost of $1I + (6t + 4)M + (5t - 2)S$, where I denotes inversion, M denotes modular multiplication, S denotes squaring, and t (the security parameter) is 192. Moreover, the error detection scheme in the previous study used $1I + (18t - 18)M + (9t - 9)S$ leading to a 27.4% overhead. As indicated in Tables I and II, our error detection scheme incurs a roughly 2% overhead in hardware and about 5% in software.

VI. Conclusion

In this brief, we introduced a scheme for detecting errors at the algorithm level within the Montgomery Ladder ECSM operation, a crucial component in cryptographic systems based on ECC. By generating two binary trees, one prior to the Montgomery Ladder operation and another concurrent with it, we implement our error detection scheme efficiently with an $O(n)$ complexity. We have shown that our proposed error detection scheme can fully detect errors that influence the Montgomery Ladder loop or arise from alterations in the private key bits. We put our proposed error detection scheme into practice by deploying it in software on the ARMv8 architecture and in hardware on two different FPGAs, namely, the Zynq Ultrascale+ and Artix Ultrascale+. In terms of hardware and software, the implementation resulted in less than 2% additional area in hardware and around 5% additional clock cycle in software.

References

[1] V. S. Miller, “Use of elliptic curves in cryptography,” in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, vol. 218. Berlin, Germany: Springer, 1986, pp. 417–426, doi: 10.1007/3-540-39799-X_31.

[2] N. Koblitz, “Elliptic curve cryptosystems,” *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.

[3] C. F. Kerry and P. D. Gallagher, *Digital Signature Standard (DSS)*, FIPS Standard 186-4, Jul. 2013.

[4] *IEEE Standard Specifications for Public-Key Cryptography*, IEEE Standard 1363-2000, Aug. 2000, pp. 1–228.

[5] *Information Technology, Security Techniques, Cryptographic Techniques Based on Elliptic Curves, Parts 1–4*, 15946 Standard 15946, 2002.

[6] *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI Standard X9.62, 1999.

[7] M. Hamburg, “Ed448-Goldilocks, a new elliptic curve,” International Association for Cryptologic Research (IACR), Tech. Rep. 625, 2015.

[8] D. Moody. (Feb. 2016). *Post-Quantum Cryptography: NIST’s Plan for the Future*. [Online]. Available: https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf

[9] (May 2024). *Safecurves: Introduction*. [Online]. Available: https://safecurves.cr.yp.to/

[10] P. Kocher, “Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems,” in *Proc. CRYPTO*, 1109, pp. 104–113.

[11] P. C. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Proc. Annu. Int. Cryptol. Conf.*, 1999, pp. 388–397.

[12] J. Daemen, C. Dobraunig, M. Eichlseder, H. Gross, F. Mendel, and R. Primas, “Protecting against statistical ineffective fault attacks,” Cryptol. ePrint Arch., Tech. Rep. 2019/536, 2019.

[13] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, “The EM side–channel(s),” in *Proc. Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2002, pp. 29–45.

[14] D. Boneh, R. A. DeMillo, and R. J. Lipton, “On the importance of checking cryptographic protocols for faults,” in *Proc. Eurocrypt*, 1997, pp. 37–51.

[15] I. Biehl, B. Meyer, and V. Müller, “Differential fault attacks on elliptic curve cryptosystems,” in *Proc. Annu. Int. Cryptol. Conf.*, 2000, pp. 131–146.

[16] P.-A. Fouque, R. Lercier, D. Réal, and F. Valette, “Fault attack on elliptic curve Montgomery ladder implementation,” in *Proc. 5th Workshop Fault Diagnosis Tolerance Cryptography*, Aug. 2008, pp. 92–98.

[17] S. Bayat-Sarmadi and M. A. Hasan, “Concurrent error detection in finite-field arithmetic operations using pipelined and systolic architectures,” *IEEE Trans. Comput.*, vol. 58, no. 11, pp. 1553–1567, Nov. 2009.

[18] K. Ahmadi, S. Aghapour, M. M. Kermani, and R. Azarderakhsh, “Efficient error detection schemes for ECSM window method benchmarked on FPGAs,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 3, pp. 592–596, Mar. 2024.

[19] K. Ahmadi, S. Aghapour, M. M. Kermani, and R. Azarderakhsh, “Error detection schemes for τ-NAF conversion within Koblitz curves benchmarked on various ARM processors,” *Authorea Preprints*, pp. 1–10, Sep. 2023.

[20] A. Cintas-Canto, M. Mozaffari-Kermani, R. Azarderakhsh, and K. Gaj, “CRC-oriented error detection architectures of post-quantum cryptography niederreiter key generator on FPGA,” in *Proc. IEEE Nordic Circuits Syst. Conf. (NorCAS)*, Oct. 2022, pp. 1–7.

[21] A. Aghaie, M. M. Kermani, and R. Azarderakhsh, “Fault diagnosis schemes for secure lightweight cryptographic block cipher RECTANGLE benchmarked on FPGA,” in *Proc. IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2016, pp. 768–771.

[22] S. Saha, D. Jap, D. B. Roy, A. Chakraborty, S. Bhasin, and D. Mukhopadhyay, “A framework to counter statistical ineffective fault analysis of block ciphers using domain transformation and error correction,” *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1905–1919, 2020.

[23] B. Lac, A. Canteaut, J. J. A. Fournier, and R. Sirdey, “Thwarting fault attacks against lightweight cryptography using SIMD instructions,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[24] A. Dominguez-Oviedo and M. A. Hasan, “Algorithm-level error detection for ECSM,” Centre Appl. Crypto. Res., Waterloo, ON, Canada, Tech. Rep. TR-2009-05, 2009.

[25] S.-M. Yen and M. Joye, “Checking before output may not be enough against fault-based cryptanalysis,” *IEEE Trans. Comput.*, vol. 49, no. 9, pp. 967–970, Feb. 2000.

[26] S.-M. Yen, S. Kim, S. Lim, and S. Moon, “A countermeasure against one physical cryptanalysis may benefit another attack,” in *Proc. 4th Int. Conf. Inf. Secur. Cryptol.* London, U.K.: Springer, 2002, pp. 414–427.

[27] J. Blömer, M. Otto, and J. P. Seifert, “Sign change fault attacks on elliptic curve cryptosystems,” in *Proc. Workshop Fault Diagnosis Tolerance Cryptography*, 2006, pp. 36–52.

[28] S. Akter, K. Khalil, and M. Bayoumi, “A survey on hardware security: Current trends and challenges,” *IEEE Access*, vol. 11, pp. 77543–77565, 2023.

[29] A. M. Awaludin, J. Park, R. W. Wardhani, and H. Kim, “A high-performance ECC processor over Curve448 based on a novel variant of the Karatsuba formula for asymmetric digit multiplier,” *IEEE Access*, vol. 10, pp. 67470–67481, 2022.

[30] A. Russon, “Differential fault attack on Montgomery Ladder and in the presence of scalar randomization,” in *Proc. INDPCRYPT*, 2021, pp. 287–310.