Side-Channel Analysis and Countermeasure Design for Implementation of Curve448 on Cortex-M4

HASP 2022

Mojtaba Bisheh-Niasar[1], Mila Anastasova[1], Abubakr Abdulgadir[2], Hwajeong Seo[3], and Reza Azarderakhsh[1,2]

[1]CEECS Department, Florida Atlantic University, FL, USA

[2]PQSecure Technologies, LLC, Boca Raton, FL, USA

[3]Hansung University, South Korea

October 1, 2022

Outline

Introduction

- Motivation
  - Complexities of Curve448 with extended field size on IoT devices
  - Secure implementation challenges due to SCA leakages
  - Importance of hybrid cryptosystems to transition to PQC
  - Lack of Curve448 implementation
- Curve448 Protected Implementation
  - Addressing backdoor issues in other ECC constructions [1]
  - Considering Safe-Curve policies [2]
  - Offering 224-bit security for applications at a higher security level

[1] D. J. Bernstein and T. Lange. 2011. Security dangers of the NIST curves. https://www.hyperelliptic.org/tanja/vortraege/20130531.pdf
[2] D. J. Bernstein and T. Lange. 2016. SafeCurves: choosing safe curves for elliptic-curve cryptography.

Background

- Curve448 architecture on Cortex-M4 by Seo et. al. [1]
  - Performed 26 scalar multiplications per second at 168 MHz
  - Utilized extended affine and projective coordinates
  - Investigated constant-time algorithms
- Curve448 implementations
  - FPGA implementations and SCA evaluation by Sasdrich et. al. and Bisheh-Niasar et. al.
  - Ed448 implementation on AVR, MSP by Seo et. al.
  - Ed448 implementation on FPGA by Bisheh-Niasar et. al.

- Improve field arithmetic with careful memory management
- Employ efficient restricted-X coordinates
- Embed advanced security mechanisms to avoid DPA attacks

[1] Hwajeong Seo and Reza Azarderakhsh. Curve448 on 32-Bit ARM Cortex-M4. In Information Security and Cryptology - ICISC 2020

Our Contributions

- Optimize low-level field arithmetics:
  - Carry/borrow catcher technique
  - Refined-Operand Caching method
  - Interleaved reduction technique
- Exploit the special form of Curve448 prime
- Implement and evaluate side-channel and fault injection (FI) countermeasures

Our Contributions

- Optimize low-level field arithmetics:
  - Carry/borrow catcher technique
  - Refined-Operand Caching method
  - Interleaved reduction technique
- Exploit the special form of Curve448 prime
- Implement and evaluate side-channel and fault injection (FI) countermeasures

- Reduce memory access
- 18% low-level speedup
- 40% total speedup

Our Contributions

- Optimize low-level field arithmetics:
  - Carry/borrow catcher technique
  - Refined-Operand Caching method
  - Interleaved reduction technique

  - Reduce memory access
  - 18% low-level speedup
  - 40% total speedup

- Exploit the special form of Curve448 prime

- Implement and evaluate side-channel and fault injection (FI) countermeasures

  - Prevent leakage at the cost of 8%-22% overhead
  - A trade-off between performance and required protection

Cortex-M4 Platform

- NIST recommended STM32F407-VG platform:
  - ARMv7-M architecture
  - 16 32-bit core registers
  - 32 32-bit FP registers
  - 1 cycle per instruction except memory access

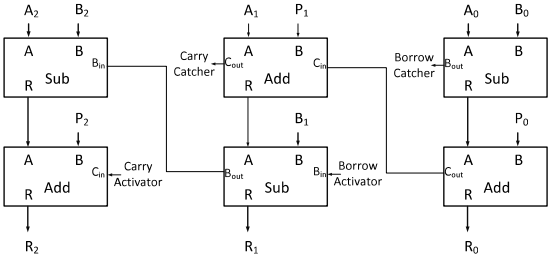
Implementation strategies:

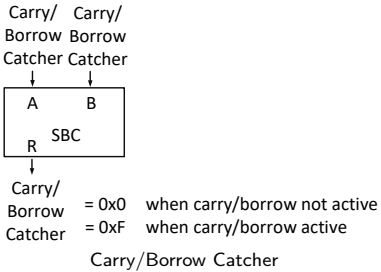
- Use the entire register set.
- Operate on larger operand sets.
- Re-organize the instruction flow for efficient design.

STM32F407-VG Discovery board

Modular Addition/Subtraction

- Implement a carry/borrow catcher technique
- Use reduced instruction set
- Two arithmetic operations on long integers in parallel
- Alternate the add/sub blocks
- Reduce the number of memory accessing instructions



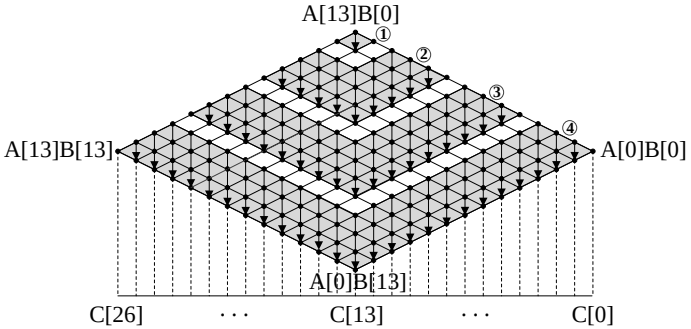


Alternating the add/sub blocks

Modular Multiplication

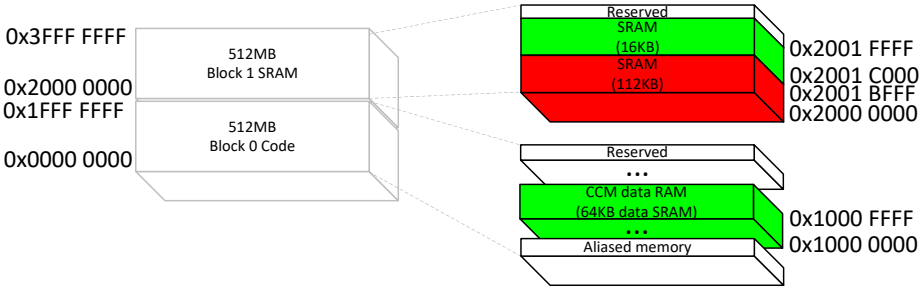
- Refined-Operand Caching method
- Multi-precision multiplication achieved by UMAAL instruction
- Reduce the number of pipeline stalls by removing the interdependency
- Caching width = 4 words (128-bit)

- $A = (A[13], ..., A[1], A[0])$
- $B = (B[13], ..., B[1], B[0])$
- $C = (C[27], ..., C[1], C[0])$



448-bit wise multi-precision multiplication.

Memory Managment



Cortex-M4 memory map

- Features 1MB of flash and 192KB of RAM - 128KB SRAM and 64KB CCM RAM.
- The 128KB of SRAM - not enough to run Curve448 scalar multiplication.
- Reserve a region inside the CCM RAM to place part of the large data structures, residing into the stack.

Implementation Results

- Implementation results for Curve448 scalar multiplication on STM32F4 platform:

Work	Freq [MHz]	Memory [B]	Latency [CC$\times 10^6$]	Time [ms]	Throughput [op/sec]	Improvement [%]
Seo et. al. [1]	24	-	6.218	259	3.9	-
Double-and-always-Add		564	5.269	219	4.6	15.2
Montgomery ladder		788	3.740	155	6.4	39.8
Seo et. al. [1]	168	-	6.286	37.4	26.7	-
Double-and-always-Add		564	5.532	32.9	30.4	12.0
Montgomery ladder		788	3.917	23.3	42.9	37.6

- Montgomery ladder:
  - 29% performance improvement over the restricted X-coordinate
  - 40% memory utilization penalty
- 1.6$\times$ speedup (43 scalar multiplications per second) at 168 MHz

[1] Hwajeong Seo and Reza Azarderakhsh. Curve448 on 32-Bit ARM Cortex-M4. In Information Security and Cryptology - ICISC 2020

Implementation Results

- Implementation results on embedded processors:

Algorithm	pre/post quantum	Cortex	Freq [MHz]	Latency [CC$\times 10^3$]	Time [ms]	Throughput [op/sec]
Curve25519 [Fujii et. al.]	pre	M4	48	907	18.9	52.9
Secp256r1 [Lenngren]	pre	M4	64	994	15.5	64.3
Four$\mathbb{Q}$ [Liu et. al.]	pre	M4	168	511	3.0	328.8
Secp384r1 [Tschofenig et. al.]	pre	M3	100	20,200	202	4.9
Secp521r1 [Tschofenig et. al.]	pre	M3	100	35,100	351	2.8
SIKEp434 [Anastasova et. al.]	post	M4	24	68,260	2,844	0.3
Curve448 [Ours]	pre	M4	168	3,917	23.3	42.9

- Higher security levels come with a performance penalty
- Possibility of algorithmic improvements to reduce the required computation to break ECC.
- Moving to a higher security level to keep a margin against unknown attack improvements

SCA-Protected Performance Results

❑ Protected Cortex-M4 implementation results:

Countermeasure	Freq [MHz]	Latency [CC$\times 10^6$]	Time [ms]	Throughput [op/sec]	Cost [%]
Unprotected	168	3.917	23.3	42.9	-
Point Randomization		4.222	25.1	39.8	7.8
Scalar Blinding		4.417	26.3	38.0	12.7
Both countermeasures		4.789	28.5	35.1	22.2

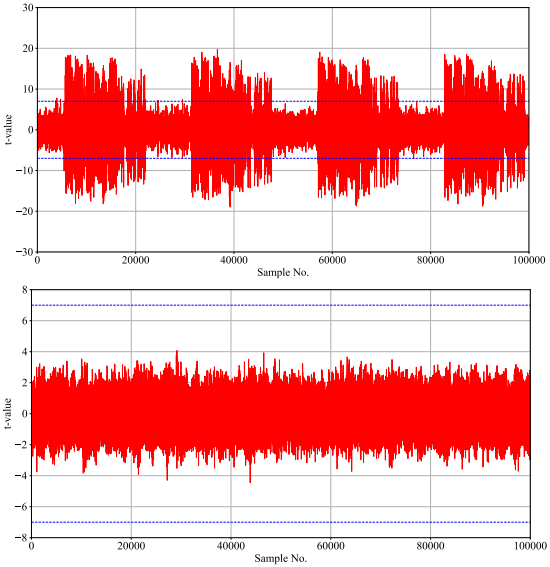
- Montgomery ladder: countermeasure against timing, SPA, and sign change fault attacks
- Base point randomization: add one multiplication per ladder step
- Scalar blinding:
  - Countermeasure to avoid DPA, cross-correlation, safe-error, and differential fault analysis attacks
  - Extend the number of ladder step iterations
- Flow-counter countermeasure: protection against FI loop-abort attacks with negligible latency overhead

SCA Evaluation

- NewAE CW308T-STM32F
- Cortex-M4 at 25 MHz
- Capturing power traces via Picoscope 3000
- Sampling rate of 125 MS/s
- TVLA with pool of 10,000 traces

$$\alpha = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_1^2}{n_2}}}$$

Leakage detection test on Curve448 after applying TVLA with a pool of 10,000 measurements:
(Up) t-test values for an unprotected implementation,
(Down) t-test values by enabling both countermeasures



Conclusion and future works

- Conclusion:
  - Implementing a secure design of Curve448 targeting a 224-bit security level for the 32-bit ARM Cortex-M4 architecture
  - Reducing the latency of scalar multiplication to 23 milliseconds at 168 MHz
  - Embedding different effective countermeasures at the cost of 8%-22% overhead
  - Evaluating our SCA protection with TVLA over 10,000 power measurements
- Future work:
  - Extending the design by fault attack countermeasures
  - Applying the method on Ed448

Thanks for your attention.