Time-Efficient Finite Field Microarchitecture Design for Curve448 and Ed448 on Cortex-M4

Mila Anastasova[1], Reza Azarderakhsh[1], Mehran Mozaffari Kermani[2] and Lubjana Beshaj[3]

[1]CEECS Department and I-SENSE at Florida Atlantic University, Boca Raton, FL, USA
(manastasova2017, razarderakhsh)@fau.edu

[2]CES Department at University of South Florida, Tampa, FL, USA,
mehran2@usf.edu

[3]United States Military Academy West Point, West Point, NY, USA,
lubjana.beshaj@westpoint.edu

December 2022

Content

1. Introduction
2. X448 & Ed448
3. Target Platform
4. Preliminaries - Multi-precision arithmetic
5. Proposed Finite Field Design
6. Proposed Finite Field Design
7. Performance Results
8. Conclusions

Introduction

- The elliptic curve family of schemes has the lowest computational latency, memory use, energy consumption, and bandwidth requirements, making it the most preferred public key method for adoption into network protocols.
- The attractive properties of the relatively new curve Curve448 contribute to its inclusion in the TLS1.3 protocol and show high interest of academics and engineers in studying and optimizing the schemes.
- Despite the later focus in post-quantum robust primitives, an efficient implementation is needed for the transition to PQ cryptography.

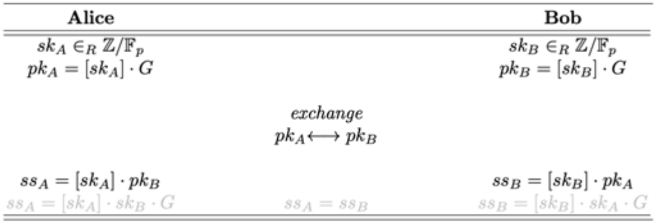
Related Work

- Several X448 and Ed448 optimizations have been presented targeting high-end devices.
  - *Oliveira et al.* present an optimal fixed-point multiplication strategy targeting Haswell and the Skylake [7].
  - *Faz-Hernandez et al.* present speed optimizations targeting the Intel AVX2 vector instruction set [4].
- X448 and Ed448 Low-end devices have not been so well studied.
  - *Seo* presents an optimized implementation of Curve448 arithmetic targeting 8-bit AVR and 16-bit MSP processors [8].
  - *Seo et al.* [11] and *Anastasova et al.* [3] present the first implementation of X448/Ed448 targeting Cortex-M4.
- Multi-precision arithmetic optimizations were also presented in the literature [6, 12, 13, 9].

Our Contributions

- We present a novel design for the underlying finite field operations multi- precision multiplication and squaring targeting the ARM Cortex-M4 plat- form.
  - We present the first handcrafted assembly implementation of multi-precision squaring procedure with the goal of improving Curve448 and Ed448 for the ARMv7-M architecture.
  - We present a speedup of around 48% and 11% for the X448 and Ed448 DSA protocols, compared to the best previously reported results in [11] and [3].
- We evaluate the proposed design's performance at 24MHz and at 168MHz on the NIST recommended STM32F407VG.

Elliptic Curve Diffie Hellman based on Curve448

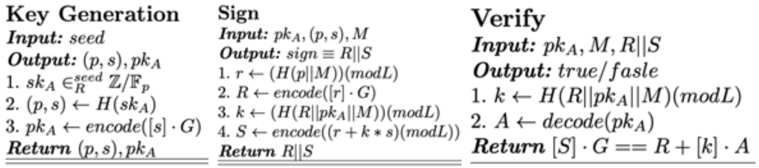


Alice		**Bob**
$sk_A \in_R \mathbb{Z}/\mathbb{F}_p$		$sk_B \in_R \mathbb{Z}/\mathbb{F}_p$
$pk_A = [sk_A] \cdot G$		$pk_B = [sk_B] \cdot G$
	exchange $pk_A \longleftrightarrow pk_B$	
$ss_A = [sk_A] \cdot pk_B$		$ss_B = [sk_B] \cdot pk_A$
$ss_A = [sk_A] \cdot sk_B \cdot G$	$ss_A = ss_B$	$ss_B = [sk_B] \cdot sk_A \cdot G$

Figure – X448 algorithm. G represents the value of the base point

- Each party must apply the scalar-point multiplication function X448 depending on the scalar value of their secret key and a public base point G, for instance using the Montgomery ladder.

Edwards Curve Digital Signature Algorithm (EdDSA) based on Ed448



```
Key Generation
Input: seed
Output: $(p, s), pk_A$
1. $sk_A \in_R^{seed} \mathbb{Z}/\mathbb{F}_p$
2. $(p, s) \leftarrow H(sk_A)$
3. $pk_A \leftarrow encode([s] \cdot G)$
Return $(p, s), pk_A$
```

```
Sign
Input: $pk_A, (p, s), M$
Output: $sign \equiv R||S$
1. $r \leftarrow (H(p||M))(modL)$
2. $R \leftarrow encode([r] \cdot G)$
3. $k \leftarrow (H(R||pk_A||M))(modL)$
4. $S \leftarrow encode((r + k * s)(modL))$
Return $R||S$
```

```
Verify
Input: $pk_A, M, R||S$
Output: $true/fasle$
1. $k \leftarrow H(R||pk_A||M)(modL)$
2. $A \leftarrow decode(pk_A)$
Return $[S] \cdot G == R + [k] \cdot A$
```

Figure – Ed448 algorithm [31]. H denotes SHAKE256. L represents the order of Ed448 curve. G represents the value of the base point

- The Edwards Curve Digital Signature Algorithm (EdDSA) is defined in three phases - Key Generation, Sign and Verify.

ARM Cortex-M4

The STM32F407VG microcontroller is recommended by NIST for benchmarking cryptographic algorithms on low-end devices.

Features

- ARMv7-M architecture
- 16 32-bit core registers
- 32 32-bit FP registers
- 1 CC per instruction except memory accesses

Implementation Design

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

Operand Scanning

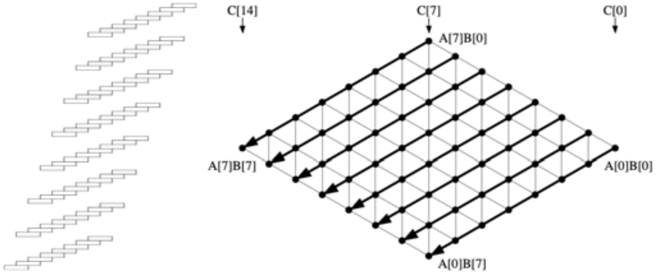


Fig. 1: Operand-scanning multiplication of 8-word large Integers a and b.

Figure – Operand Scanning (schoolbook or row-wise multiplication) - figure taken from [6]

Product Scanning

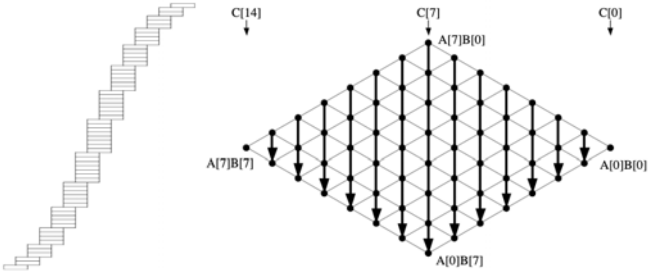


Fig. 2: Product-scanning multiplication of 8-word large Integers a and b.

Figure – figure taken from [6]

Hybrid Multiplication

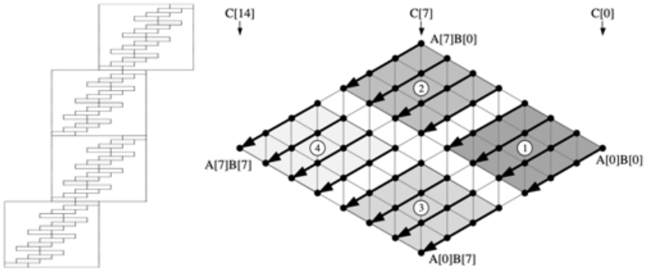


Fig. 3: Hybrid multiplication of 8-word large Integers a and b ($d = 4$).

Figure – figure taken from [6]

Operand Caching

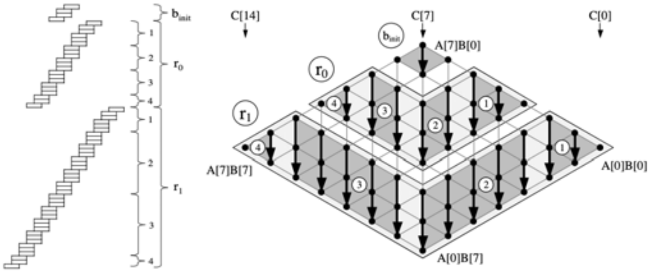


Fig. 4: Operand-caching multiplication of 8-word large Integers a and b ($e = 3$).

Figure – Operand Caching with row size of 3 - figure taken from [6]

Consecutive Operand Caching

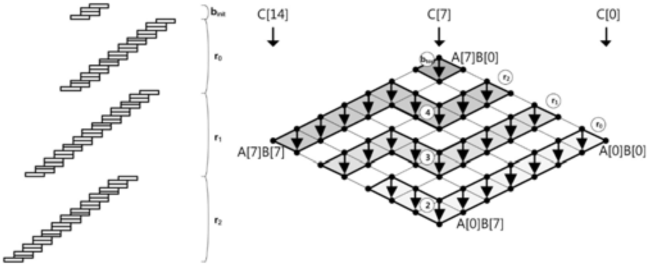


Fig. 2. Consecutive Operand-Caching Method

Figure – Consecutive Scanning where last loaded words per row are reused at the beginning of the next row - figure taken from [13]

Refined Operand Caching

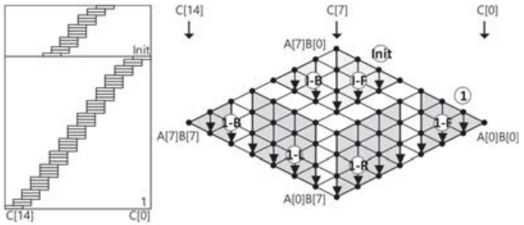


Fig. 4. Proposed 256-bit Refined Operand Caching multiplication at the word-level where e is 4 on ARM Cortex-M4, Ⓘnit: initial block; ①: order of rows; Ⓕ : front part; Ⓡ : middle right part; Ⓛ : middle left part; Ⓑ : back part.

Figure – Refined Operand Caching with increase row size to 4 accumulative 32$\times$32-bit multiplications - figure taken from [10]

Extended Operand Caching

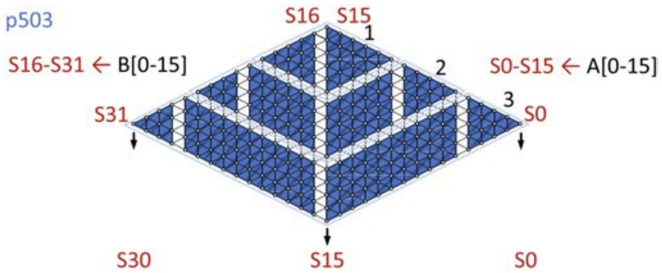


Figure – Extended Operand Caching with floating point register bank utilization for fast reload - figure taken from [1]

Hybrid^2 Multiplication

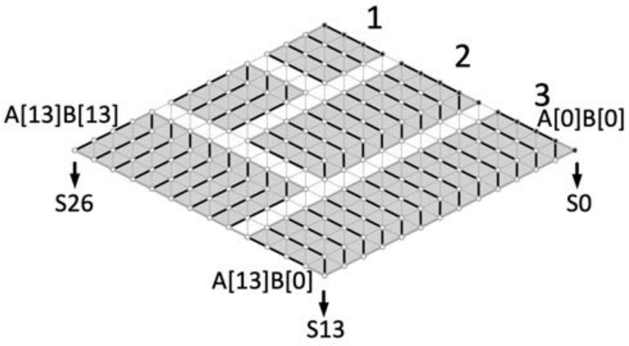


Fig. 2. Proposed architecture for 448-bit multi-precision multiplication. Black lines denote inner loop execution flow.

Figure – Hybrid multi-precision multiplication in the outer and the inner multiplication loop - figure taken from [2]

Proposed Design - Notation

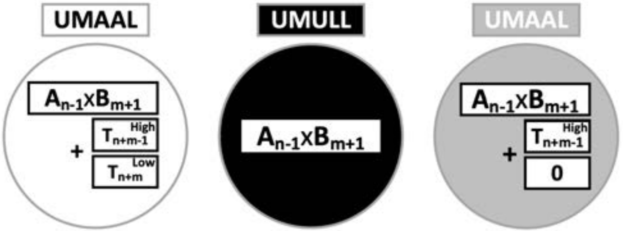


Figure – Instruction set notation in dot format for the rhombus representation of the arithmetic operations multiplication and squaring.

Multi-precision Multiplication

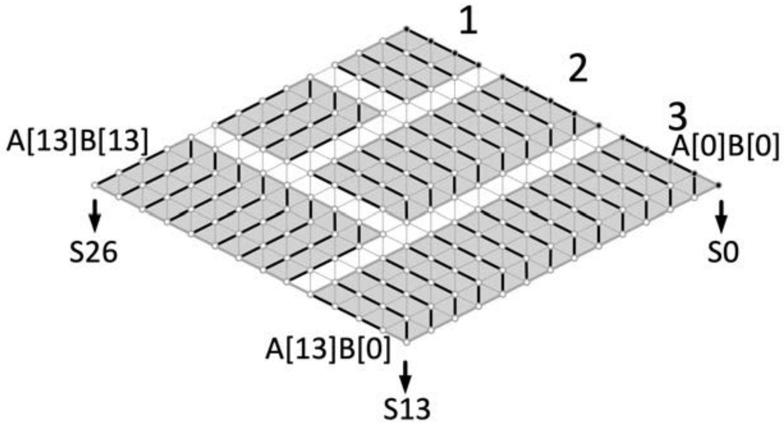


Figure – Proposed architecture for 448-bit multi-precision multiplication. Black lines denote inner loop execution flow.

Multi-precision Multiplication

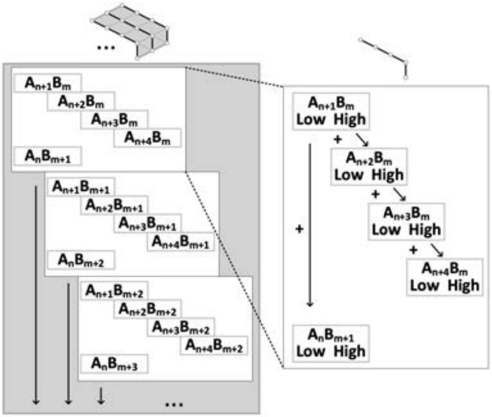
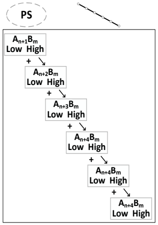
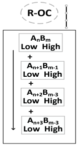


Figure – Proposed design, register utilization and carry propagation for the multiprecision multiplication outer (left) and inner (right) loop execution flow.

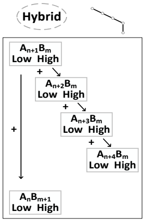
Proposed Design - Notation



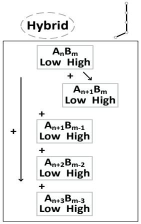
(a) Product Scanning (PS)



(b) R-OC, identical to OS



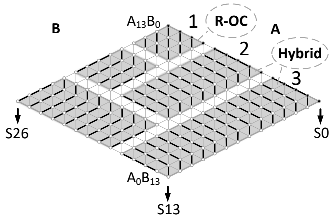
(c) Hybrid (4 PS&1 OS).



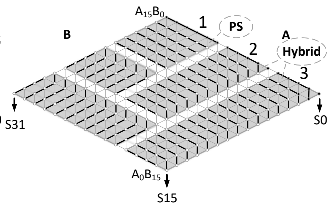
(d) Hybrid (1 PS&4 OS).

Figure – Deployed list of inner multi-precision loop execution flows along with the associated assembly instruction set.

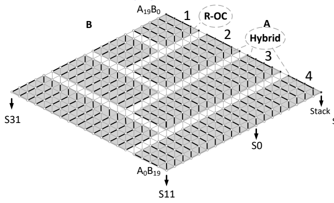
Multi-precision Multiplication



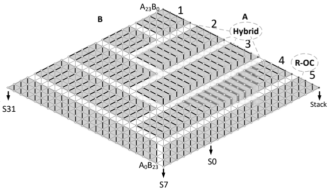
(a) P434.



(b) P503.



(c) P610.



(d) P751.

Figure – Proposed architecture for multi-precision multiplication.

Multi-precision Squaring

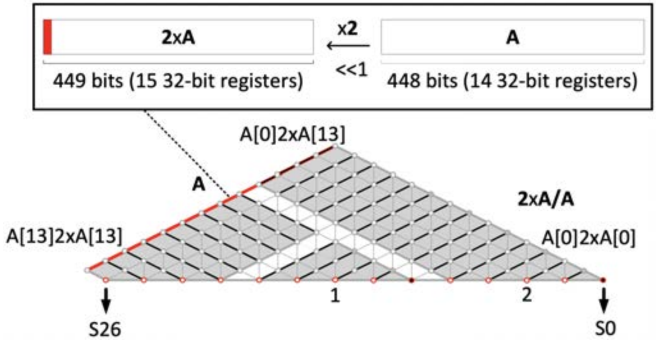

2xA
x2
<<1
A
449 bits (15 32-bit registers)
448 bits (14 32-bit registers)
A[0]2xA[13]
A
2xA/A
A[13]2xA[13]
A[0]2xA[0]
1
2
S26
S0

Figure – Proposed architecture for 448-bit multi-precision multiplication with 14th index word for the doubled operand value (red line).

Multi-precision Squaring

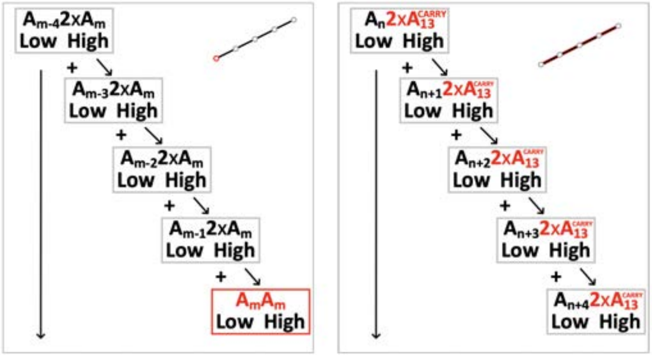
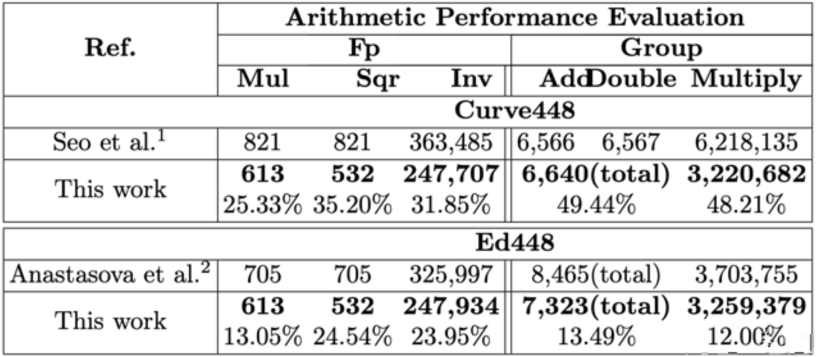


Figure – Proposed design, register utilization and carry propagation for the multi-precision squaring with coinciding indexes of the operand (left) and with the carry word produced after doubling the operand (right).

Results Subroutine Latency

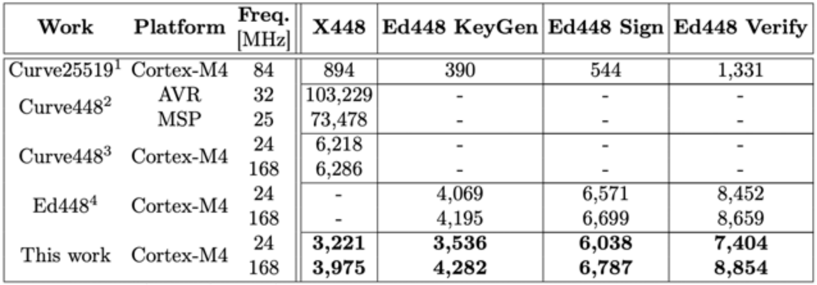


Ref.	Arithmetic Performance Evaluation					
	Fp			Group		
	Mul	Sqr	Inv	Add	Double	Multiply
Curve448						
Seo et al.[1]	821	821	363,485	6,566	6,567	6,218,135
This work	**613** 25.33%	**532** 35.20%	**247,707** 31.85%	**6,640(total)** 49.44%		**3,220,682** 48.21%
Ed448						
Anastasova et al.[2]	705	705	325,997	8,465(total)		3,703,755
This work	**613** 13.05%	**532** 24.54%	**247,934** 23.95%	**7,323(total)** 13.49%		**3,259,379** 12.00%

Figure – Finite field operations for Curve448/Ed448 targeting ARMv7-M.

Refer to : [1] [11], [2] [3]

Performance Results @24MHz



Work	Platform	Freq. [MHz]	X448	Ed448 KeyGen	Ed448 Sign	Ed448 Verify
Curve25519[1]	Cortex-M4	84	894	390	544	1,331
Curve448[2]	AVR	32	103,229	-	-	-
	MSP	25	73,478	-	-	-
Curve448[3]	Cortex-M4	24	6,218	-	-	-
		168	6,286	-	-	-
Ed448[4]	Cortex-M4	24	-	4,069	6,571	8,452
		168	-	4,195	6,699	8,659
This work	Cortex-M4	24	**3,221**	**3,536**	**6,038**	**7,404**
		168	**3,975**	**4,282**	**6,787**	**8,854**

Figure – Curve 25519 and Curve448 key exchange and digital signature computation latency performance on IoT platforms.

Refer to : [1] [5], [2] [8], [3] [11], [4] [3]

Conclusions

- We present a novel design for time-efficient finite field arithmetic over Curve448 and its birationally equivalent Ed448.
- We present the first hybrid implementation of operand and product scanning techniques in the multiplication routine's inner loop.
- We observe around 48% speedup for X448 and around 11% of speedup for Ed448.

References

[1] M. Anastasova, R. Azarderakhsh, and M. M. Kermani. Fast strategies for the implementation of SIKE round 3 on ARM Cortex-M4. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 68(10) :4129–4141, 2021.

[2] M. Anastasova, R. Azarderakhsh, M. M. Kermani, and L. Beshaj. Time-efficient finite field microarchitecture design for curve448 and ed448 on cortex-m4. *Cryptology ePrint Archive*, 2023.

[3] 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 *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 93–96. IEEE, 2022.

[4] A. Faz-Hernández, J. López, and R. Dahab. High-performance implementation of elliptic curve cryptography using vector instructions. *ACM Transactions on Mathematical Software (TOMS)*, 45(3) :1–35, 2019.

[5] H. Fujii and D. F. Aranha. Curve25519 for the Cortex-M4 and beyond. In *International Conference on Cryptology and Information Security in Latin America*, pages 109–127. Springer, 2017.

[6] M. Hutter and E. Wenger. Fast multi-precision multiplication for public-key cryptography on embedded microprocessors. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 459–474. Springer, 2011.

[7] T. Oliveira, J. López, H. Hışıl, A. Faz-Hernández, and F. Rodríguez-Henríquez. How to (pre-) compute a ladder. In *International Conference on Selected Areas in Cryptography*, pages 172–191. Springer, 2017.

[8] H. Seo. Compact implementations of Curve Ed448 on low-end IoT platforms. *ETRI Journal*, 41(6) :863–872, 2019.

[9] H. Seo. Memory efficient implementation of modular multiplication for 32-bit ARM Cortex-M4. *Applied Sciences*, 10(4) :1539, 2020.

[10] H. Seo, M. Anastasova, A. Jalali, and R. Azarderakhsh. Supersingular isogeny key encapsulation (SIKE) round 2 on ARM Cortex-M4. *IEEE Transactions on Computers*, 70(10) :1705–1718, 2020.

[11] H. Seo and R. Azarderakhsh. Curve448 on 32-bit ARM Cortex-M4. In *International Conference on Information Security and Cryptology*, pages 125–139. Springer, 2020.

[12] H. Seo and H. Kim. Multi-precision multiplication for public-key cryptography on embedded microprocessors. In *International Workshop on Information Security Applications*, pages 55–67. Springer, 2012.

[13] H. Seo and H. Kim. Consecutive operand-caching method for multiprecision multiplication. *Journal of information and communication convergence engineering*, 13(1) :27–35, 2015.