The 23rd Annual International Conference on Information Security and Cryptology
ICISC 2020
December 2 (Wed) ~ December 4 (Fri), 2020 | Virtual Conference

Hosted by
Korea Institute of Information Security and Cryptology (KIISC)
National Security Research Institute (NSR)

Korea Institute of Information Security & Cryptology

Curve448 on 32-bit ARM Cortex-M4

Hwajeong Seo, Reza Azarderakhsh
Hansung University,
Florida Atlantic University,
PQSecure Technologies, LLC

Contents

Introduction

- **Public key cryptography**
  - Key exchange and digital signature protocols

- **Challenge: implementation of PKC on low-end microcontrollers**
  - Low-end microcontrollers: low energy, performance, and memory

  - Efficiency of ECC:
    compact implementation of finite field arithmetic & group operations

Motivation

- **Importance of Curve448**
  - 224-bit security for ECDH.
  - Favored by IRTF CFRG for TLS standards along with Curve25519.
  - Confirmed in FIPS 186-5 (US federal government).
  - However, the implementation of Curve448 has not been actively conducted.

- **Few Curve448 works on 32-bit ARM Cortex-M4**
  - Widely used in practice
    (Relatively powerful computational ability: ALU, frequency, RAM, and ROM)
  - Recommended by NIST post-quantum cryptography

Contribution

- **First implementation of Curve448 on 32-bit ARM Cortex-M4**
- **Secure and efficient implementation of primitive operations**
- **In-depth comparison of pre-quantum and post-quantum cryptography**
- **First Curve448 on ARM Cortex-M4 as an open source**
  https://github.com/solowal/DEVELOP/tree/master/Source%20Code/ICISC'20

Related Work

- **Target curve: Curve448**
  - Edwards curve provides complete addition formulas.
  - Faster and simpler than traditional NIST curves.
  - Curve448 satisfies the requirement of SafeCurves.
  - ECC standanrds of TLS 1.3

Related Work

- **Target microcontroller: 32-bit ARM Cortex-M4**
  - Small and energy-efficient ARM processor
  - ARMv7E-M instruction set (Thumb-2 and DSP extensions)
  - 3-stage pipeline with branch speculation
  - 16 32-bit registers (R0~R15)
  - Powerful single-cycle multiply and multiply-and-accumulate instructions
  - UMAAL D, C, A, B : {D|C} ← A X B + C + D

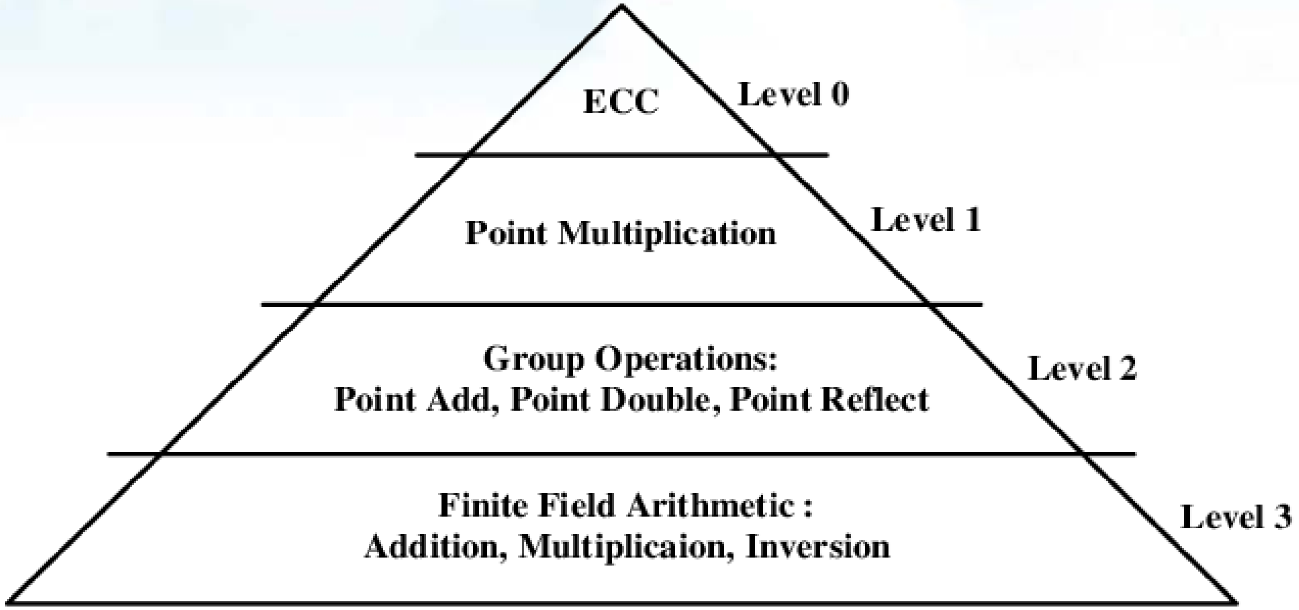
Previous Implementations

- **Curve448 (224-bit security) implementation**
  - 8-bit AVR (103,228,541 cc) and 16-bit MSP430 (73,477,660 cc)
  - No works on ARM Cortex-M4 (only Curve25519; 128-bit security)
  - For long term security, Curve448 should be considered.
- **First Curve448 implementation on 32-bit ARM Cortex-M4**

Optimization Techniques for Curve448 on 32-bit ARM Cortex-M4

• Hierarchy of ECC implementation

- **Finite field operation**
  - Finite field addition/subtraction
  - Finite field multiplication
  - Finite field inversion
- **Group operation**
  - Point addition
  - Point doubling
- **Point multiplication**
  - Montgomery ladder

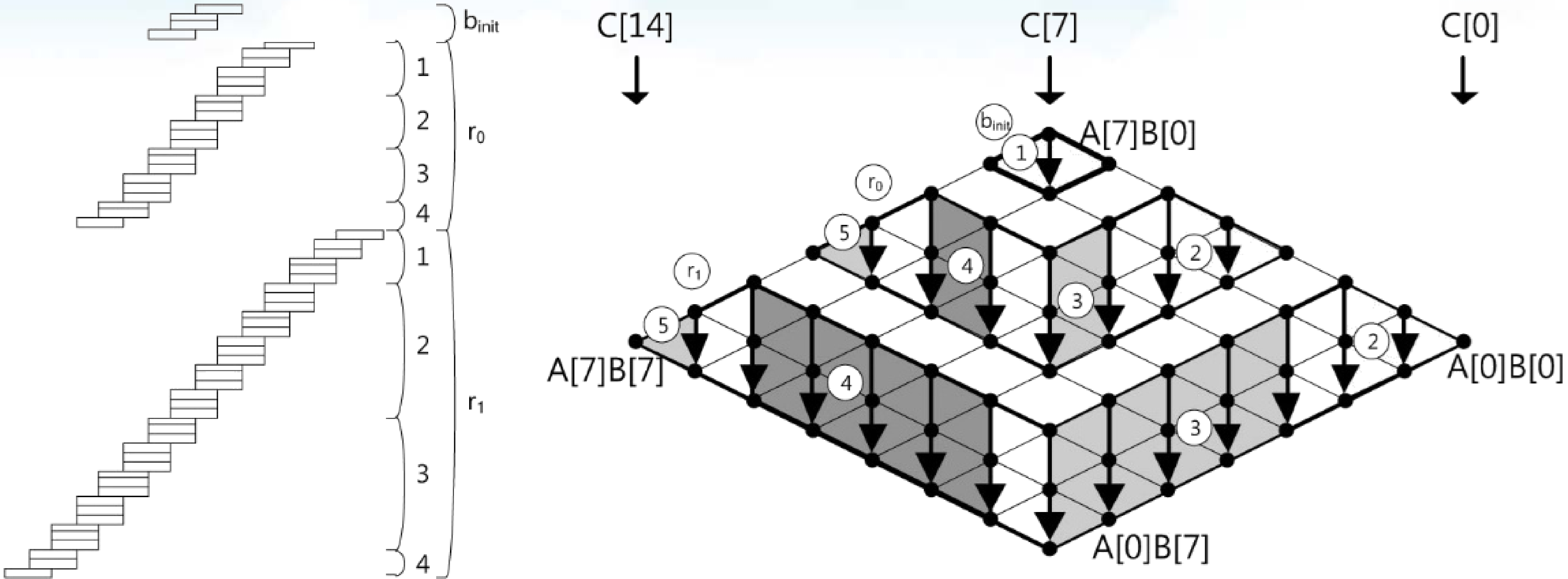


Optimization Techniques for Curve448 on 32-bit ARM Cortex-M4

- **Finite field addition**
  - **Integer addition → modular reduction**
  - **Process of (masked) finite field addition**
    1. Carry | C ← A + B (Integer addition)
    2. Mask ← 0 – carry (Mask is 0 or 0xFFFFFFFF)
    3. Masked modulus ← mask & modulus
    4. Result ← C – masked modulus

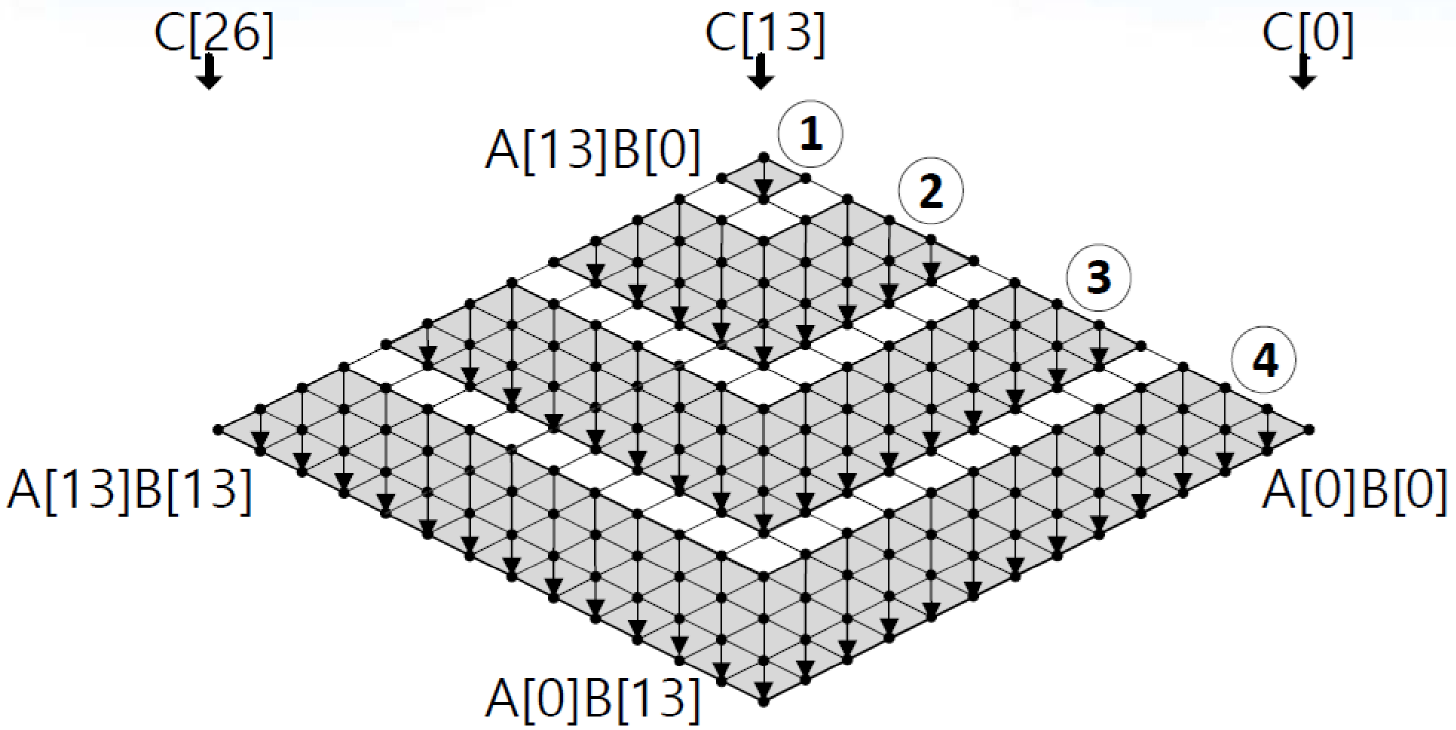
Optimization Techniques for Curve448 on 32-bit ARM Cortex-M4

- **Finite field multiplication**
  - Integer multiplication (operand caching) → modular reduction



Optimization Techniques for Curve448 on 32-bit ARM Cortex-M4

- **Finite field multiplication**
  - Integer multiplication (operand caching w/ width 4; CANS’19) → modular reduction




```
  ⋮
LDR     R6, [R0, #4 * 4]    //Loading operand B[4] from memory
LDR     R1, [SP, #4 * 4]    //Loading result C[4] from memory
UMAAL   R14, R10, R5, R7    //Partial product (B[1]*A[3])
UMAAL   R14, R11, R4, R8    //Partial product (B[2]*A[2])
UMAAL   R14, R12, R3, R9    //Partial product (B[3]*A[1])
UMAAL   R1, R14, R2, R6     //Partial product (B[4]*A[0])
  ⋮
```

1. Intermediate result is stored in STACK.
2. The result in STACK is directly reduced.

Optimization Techniques for Curve448 on 32-bit ARM Cortex-M4

- **Finite field multiplication**
  - Integer multiplication → **modular reduction**
  - **Fast reduction on Curve448 (ETRI Journal'19): 8 x 224-bit ADD**

Require: 896-bit intermediate result A (A[3]∼A[0] in 224-bit)
Ensure: 448-bit result C (C[1]||C[0] in 224-bit)

```
ε0||T ← A[2]+A[3]

ε1||C[0] ← A[0]+ε0||T
ε2||C[1] ← A[1]+A[3]+ε0||T

ε3||C[0] ← C[0]+ε2
ε4||C[1] ← C[1]+(ε1+ε2+ε3)

ε5||C[0] ← C[0]+ε4
C[1] ← C[1]+(ε4+ε5)

return C
```

Optimization Techniques for Curve448 on 32-bit ARM Cortex-M4

- **Finite field inversion**
  - **Prime of Curve448:** $p = 2^{448} - 2^{224} - 1$

  - **Fermat's theorem**
    (computation of inversion; $a = z^{-1} \equiv z^{2^{448}-2^{224}-3} \; mod \; p$)

  - **Inversion**
    - 447 squaring + 13 multiplication

Require: Integer z satisfying $1 \leq z \leq p-1$.
Ensure: Inverse $t_7 = z^{p-2} \bmod p = z^{-1} \bmod p$.

$z_3 \leftarrow z^{2^1} \cdot z$	{cost: 1S+1M}
$t_0 \leftarrow z_3^{2^2} \cdot z_3$	{cost: 2S+1M}
$t_1 \leftarrow t_0^{2^1} \cdot z$	{cost: 1S+1M}
$t_2 \leftarrow t_1^{2^4} \cdot t_0$	{cost: 4S+1M}
$t_3 \leftarrow t_2^{2^9} \cdot t_2$	{cost: 9S+1M}
$t_4 \leftarrow (t_3^{2^{18}} \cdot t_3)^2 \cdot z$	{cost: 19S+2M}
$t_5 \leftarrow (t_4^{2^{37}} \cdot t_4)^{2^{37}} \cdot t_4$	{cost: 74S+2M}
$t_6 \leftarrow t_5^{2^{111}} \cdot t_5$	{cost: 111S+1M}
$t_7 \leftarrow (t_6^{2^1} \cdot z^{2^{223}} \cdot t_6)^{2^2} \cdot z$	{cost: 226S+3M}
return t_7	

Optimization Techniques for Curve448 on 32-bit ARM Cortex-M4

- **Group operation**
  - Point addition and doubling in extended projective coordinates

- **Point multiplication**
  - Montgomery ladder algorithm

Require: Point $P1 = (x1, y1, z1, e1, h1)$ in extended projective coordinates, Point $P2 = (u2, v2, w2)$ in extended affine coordinates
Ensure: $P3 = (x3, y3, z3, e3, h3)$ in extended projective coordinates

$t1 \leftarrow e1 \cdot h1$
$e3 \leftarrow y1 - x1$
$h3 \leftarrow y1 + x1$
$x3 \leftarrow e3 \cdot v2$ $\{A = (y1 - x1) \cdot (y2 - x2)\}$
$y3 \leftarrow h3 \cdot u2$ $\{B = (y1 + x1) \cdot (y2 + x2)\}$
$e3 \leftarrow y3 - x3$ $\{E = B - A\}$
$h3 \leftarrow y3 + x3$ $\{H = B + A\}$
$x3 \leftarrow t1 \cdot w2$ $\{C = t1 \cdot w2\}$
$t1 \leftarrow z1 - x3$ $\{F = z1 - C\}$
$x3 \leftarrow z1 + x3$ $\{G = z1 + C\}$
$z3 \leftarrow t1 \cdot x3$ $\{Z3 = F \cdot G\}$
$y3 \leftarrow x3 \cdot h3$ $\{Y3 = G \cdot H\}$
$x3 \leftarrow e3 \cdot t1$ $\{X3 = E \cdot F\}$

return $P3(x3, y3, z3, e3, h3)$

Require: Point $P1 = (x1, y1, z1, e1, h1)$ in extended projective coordinates
Ensure: $P3 = (x3, y3, z3, e3, h3)$ in extended projective coordinates

$e3 \leftarrow x1 \cdot x1$ $\{A = x1 \cdot x1\}$
$h3 \leftarrow y1 \cdot y1$ $\{B = y1 \cdot y1\}$
$t1 \leftarrow e3 - h3$ $\{G = A - B\}$
$h3 \leftarrow e3 + h3$ $\{H = A + B\}$
$x3 \leftarrow x1 + y1$
$e3 \leftarrow x3 \cdot x3$
$e3 \leftarrow h3 - e3$ $\{E = H - (x1 + y1) \cdot (x1 + y1)\}$
$y3 \leftarrow z1 \cdot z1$
$y3 \leftarrow 2 \cdot y3$ $\{C := 2 \cdot z1 \cdot z1\}$
$y3 \leftarrow t1 + y3$ $\{F := G + C\}$
$x3 \leftarrow e3 \cdot y3$ $\{X3 := E \cdot F\}$
$z3 \leftarrow y3 \cdot t1$ $\{Z3 := F \cdot G\}$
$y3 \leftarrow t1 \cdot h3$ $\{Y3 := G \cdot H\}$

return $P3(x3, y3, z3, e3, h3)$

Evaluation

Frequency	Finite field operation				Group operation		
	Addition	Subtraction	Multiplication	Inversion	Addition	Doubling	Point multiplication
24 MHz	164	161	821	363,485	6,566	6,567	6,218,135
168 MHz	181	172	838	363,626	6,686	6,674	6,285,904

- **Experiment setting**
  - STM32F4 discovery board
  - GCC –O3
  - Two frequencies (24MHz and 168MHz)

- **Evaluation on low frequency (24MHz)**
  - Avoid wait cycles due to the speed of the memory controller
  - Ensure the correct clock cycles

Evaluation

Method	128-bit security		224-bit security
	Curve25519	FourQ	Curve448
Groot	1,816,351	-	-
Santis and Sigl	1,563,852	-	-
Fujii and Aranha	907,240	-	-
Haase and Labrique	847,048	-	-
Liu et al.	-	542,900	-
This work	-	-	6,218,135

- Curve448 is 86% and 91% slower than Curve25519 and FourQ.

Masked implementation	Early termination prevention	Montgomery ladder	w/o look-up table
O	O	O	O

- ECC implementations in constant timing.
  - Avoiding conditional branch and cache access.

Hybrid Post-Quantum TLS

Classical TLS 1.2	Hybrid Post-Quantum TLS 1.2
premaster_secret = ECDHE_KEY seed = “master secret” || ClientHello.random || ServerHello.random master_secret = HMAC (premaster_secret, seed)	premaster_secret = ECDHE_KEY || PQ_KEY seed = “hybrid master secret” || ClientHello.random || ServerHello.random master_secret = HMAC (premaster_secret, seed)

- **Advantage of hybrid PQ TLS**
  - Two independent key exchanges (classical and post-quantum).
  - Both keys are combined into a single TLS master secret.
  - Hybrid PQ TLS is still secure when one of key exchanges is compromised.
  - e.g. Amozon AWS evaluated ECDH w/ BIKE, SIKE.

Hybrid Post-Quantum TLS

SIKEp434	Timings [second]			
	KeyGen	Encaps	Decaps	Total
IEEE TC'20	0.32	0.53	0.56	1.09

- **Performance comparison of Hybrid PQ TLS**
  - Curve25519 (0.01 sec) + SIKE434 (1.09 sec) → 1.1 sec
    109x slower
  - FourQ (0.006 sec) + SIKE434 (1.09 sec) → 1.096 sec
    181.6x slower
  - Curve448 (0.074 sec) + SIKE434 (1.09 sec) → 1.164 sec
    14.7x slower

Conclusion

- **First implementation of Curve448 on ARM Cortex-M4**
  - Point multiplication in 6,218,135 clock cycles
    → practically fast enough for M4@168 MHz
  - Secure implementation against timing attacks
- **Future work**
  - Practical implementation of hybrid post-quantum TLS on low-end IoT

Q & A