Optimized SIKE Round 2 on 64-bit ARM

Hwajeong Seo[1], Amir Jalali[2], and Reza Azarderakhsh[2]

[1]Hansung University, [2]Florida Atlantic University

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

Motivation

- **High performance** is important to achieve **the service availability**
- The cryptography imposes the **high overheads on computers**
- Furthermore, **some PQC standard show low performance**
- We target **SIKE Round 2 curves**, such as SIKEp434 and SIKEp610

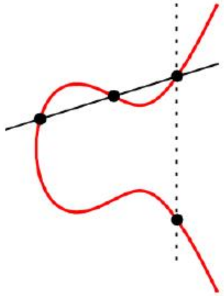
Diffie-Hellman Instantiations

$\mathbb{Z}_q$

$g^a \bmod q$

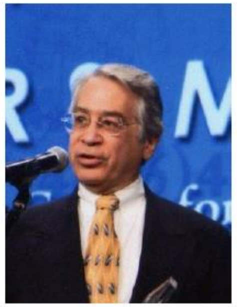
$g^b \bmod q$

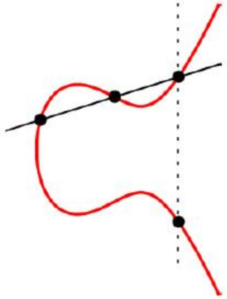
$\mathbb{Z}_q$

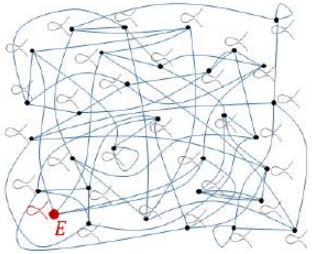


$[a]P$

$[b]P$

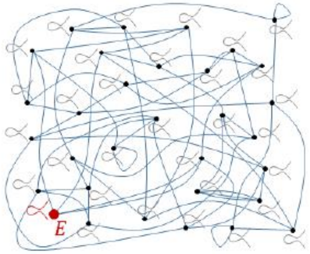




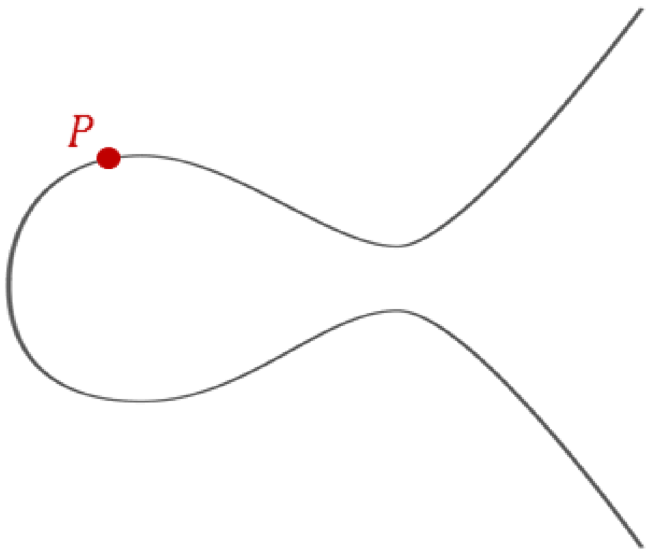


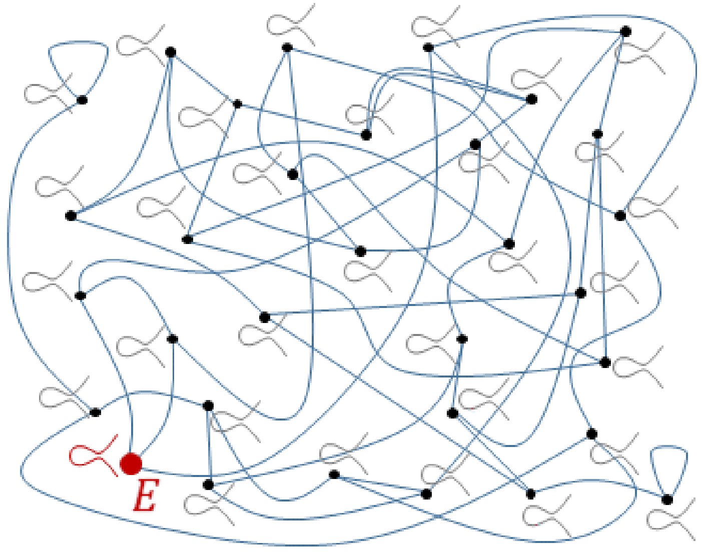
$\phi_A(E)$

$\phi_B(E)$



Pre-quantum ECC vs Post-quantum ECC





Post-quantum key exchange algorithm

- **Supersingular Isogeny Diffie-Hellman (SIDH)**
  - Shared key generation between two parties over an insecure communication channel.
  - SIDH works with the set of supersingular elliptic curves over $\mathbb{F}_{p^2}$ and their isogenies.

$$E_{AB} = \Phi'_B\left(\Phi_A(E_0)\right) \cong E_0/\langle P_A + [s_A]Q_A, P_B + [s_B]Q_B\rangle \cong E_{BA} = \Phi'_A\left(\Phi_B(E_0)\right)$$

SIKE Round 2

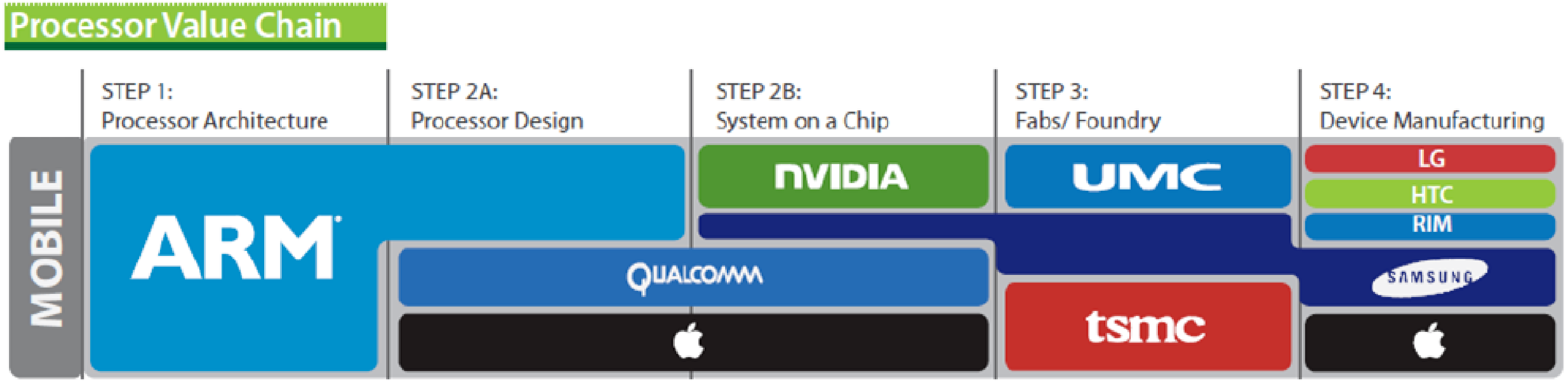
- **SIKE**
  - One of the PQC candidates
  - Hardness of solving isogeny maps between supersingular elliptic curves

- **SIKE Round 1**
  - SIKEp503 (NIST 1), SIKEp751 (NIST 3), and SIKEp964 (NIST 5)

- **SIKE Round 2**
  - New curves: SIKEp434, SIKEp610
  - Security level changes: SIKEp503 (NIST 1→2), SIKEp751 (NIST 3→5)
  - Excluding SIKEp964

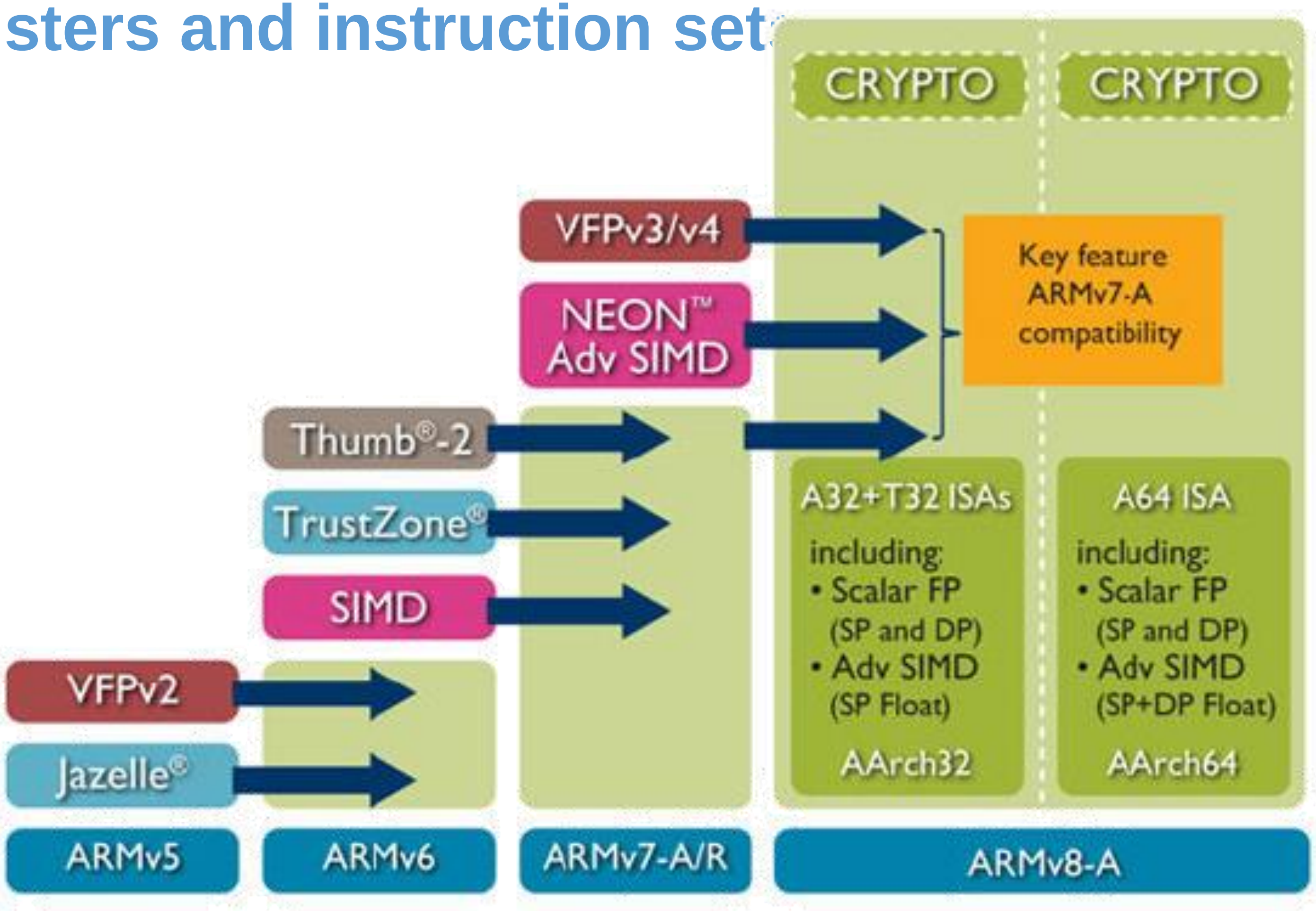
Target Platform – 64-bit ARMv8-A

- **95%** of smartphones based on ARM architecture
- Modern smartphone supports **64-bit ARMv8**
  - **e.g. iPhone and Galaxy**

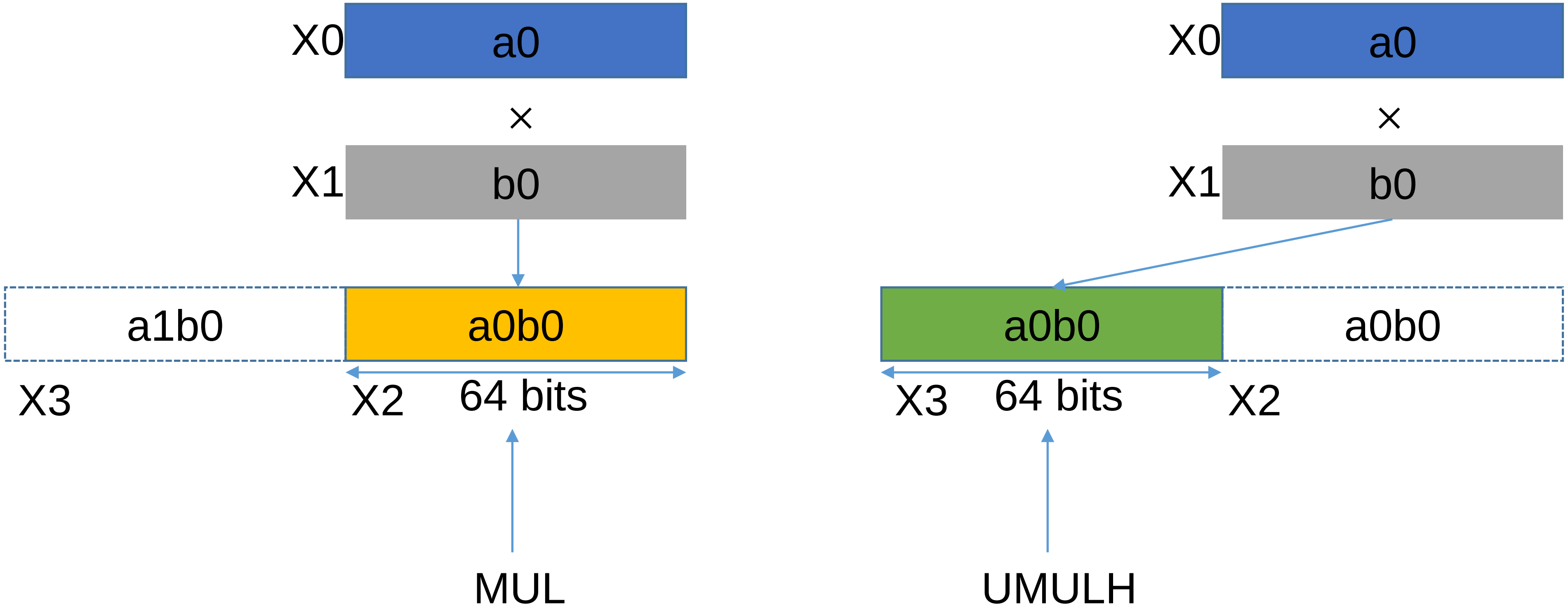


Target Platform – 64-bit ARMv8-A

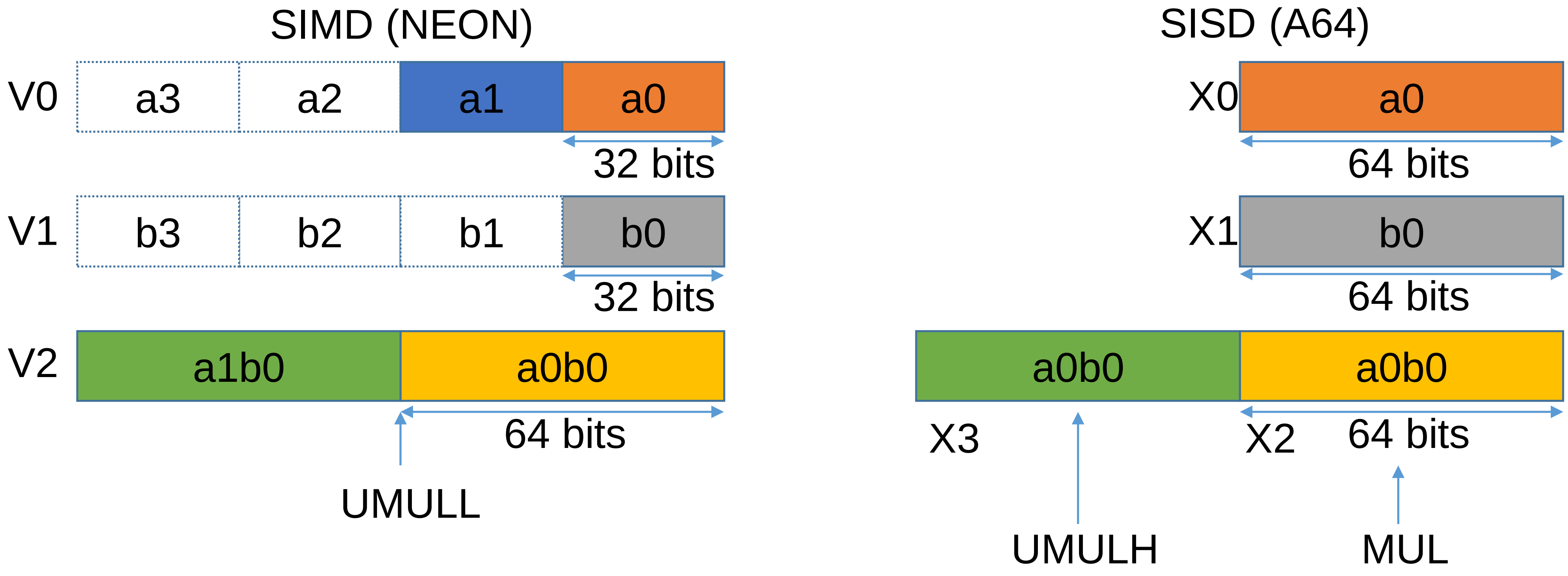
- 32-bit mode (AArch32) & 64-bit mode (AArch64)
- 64-bit ARM & **128-bit NEON registers and instruction set**
- Crypto (AES and SHA) operation



Multiplication on ARMv8



Multiplication on ARMv8



For 64-bit multiplication on **ARMv8**,
NEON requires **4 UMULL** routines but **A64** only needs **1 MUL and 1 UMULH**.
A64 is more efficient than NEON for big integer multiplication.

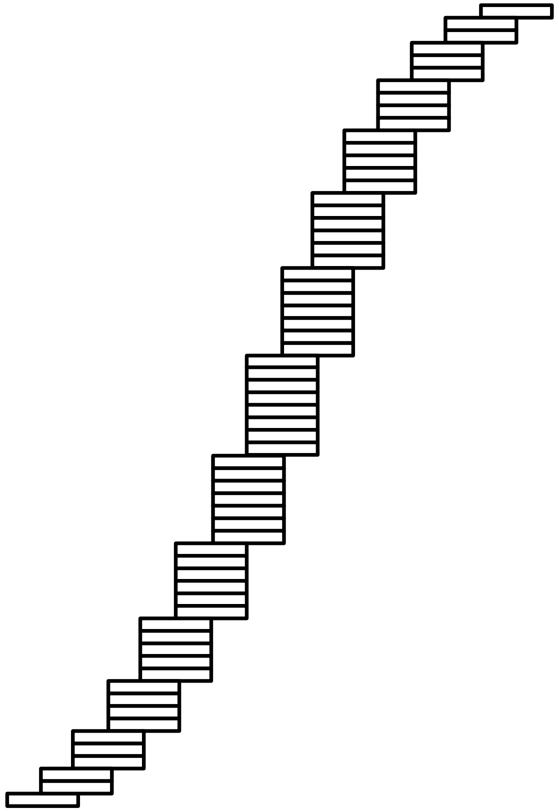
Multi-precision Multiplication

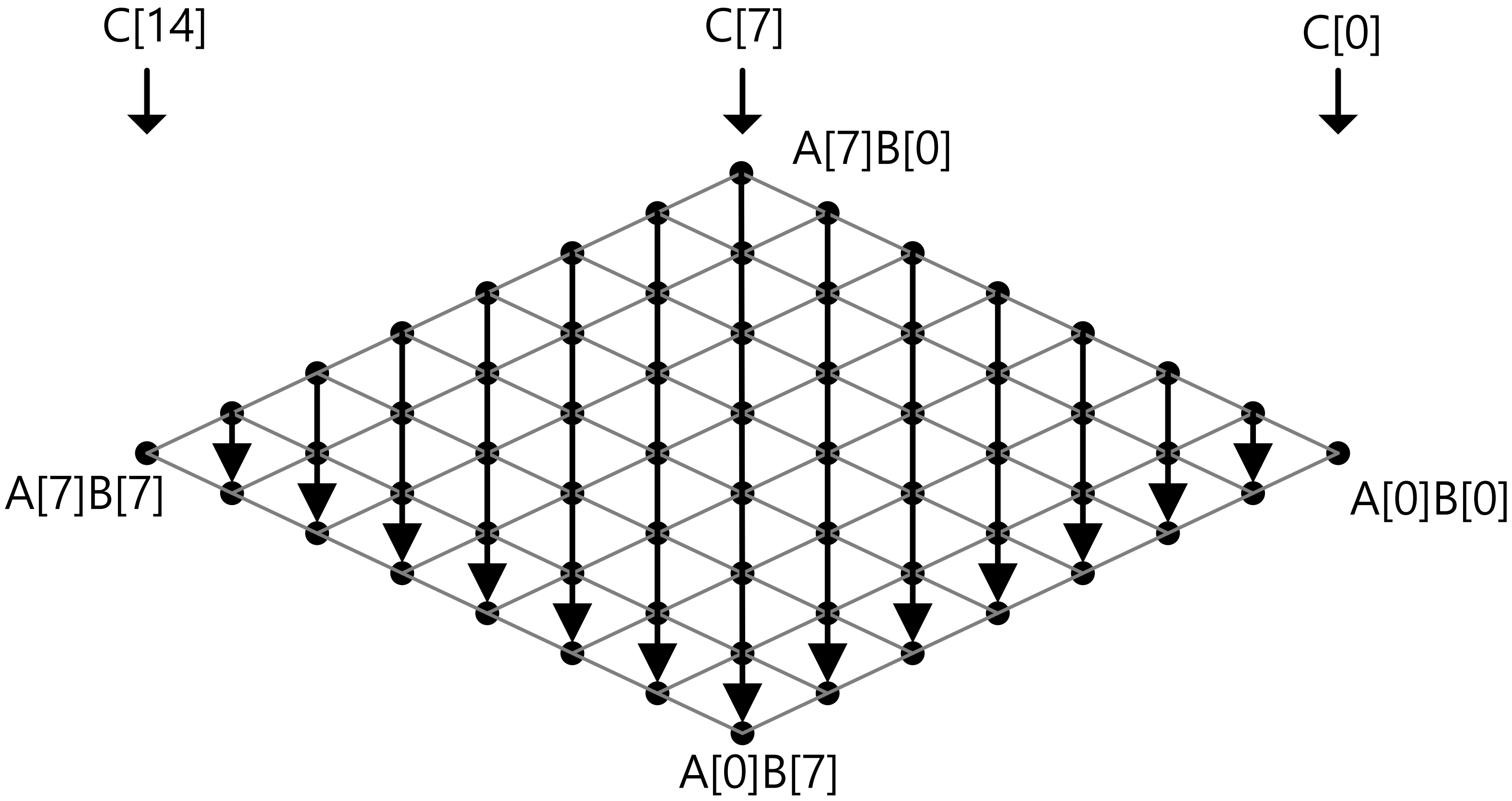
- **256~2048-bit multiplication** on 64-bit architecture
- Divide big integer (256~2048-bit) into small integer (64-bit)

Method	Operand-scanning	Product-scanning	Hybrid-scanning
Computation order	Row-wise	Column-wise	Mixture of row/column
Requirement	Many registers	Efficient MAC routine	General processor

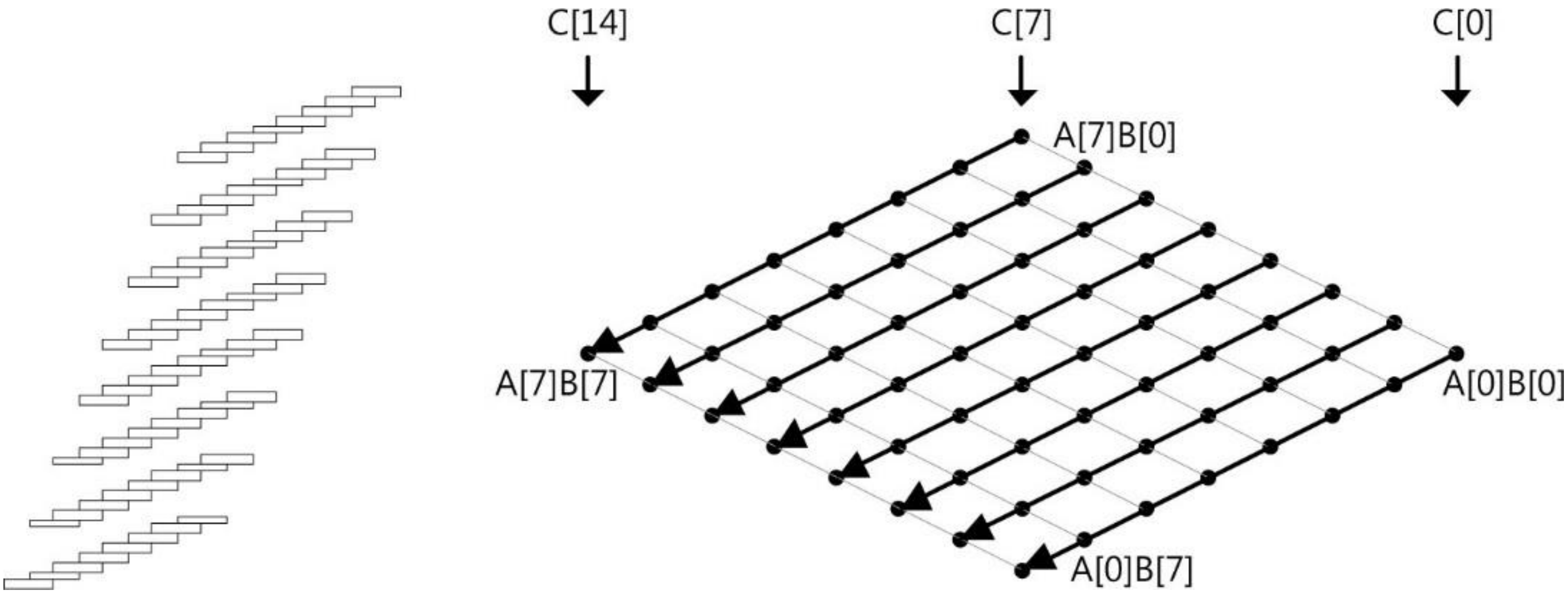
- ARMv8 supports **31x64-bit registers**

Multi-precision Multiplication (Product Scanning)



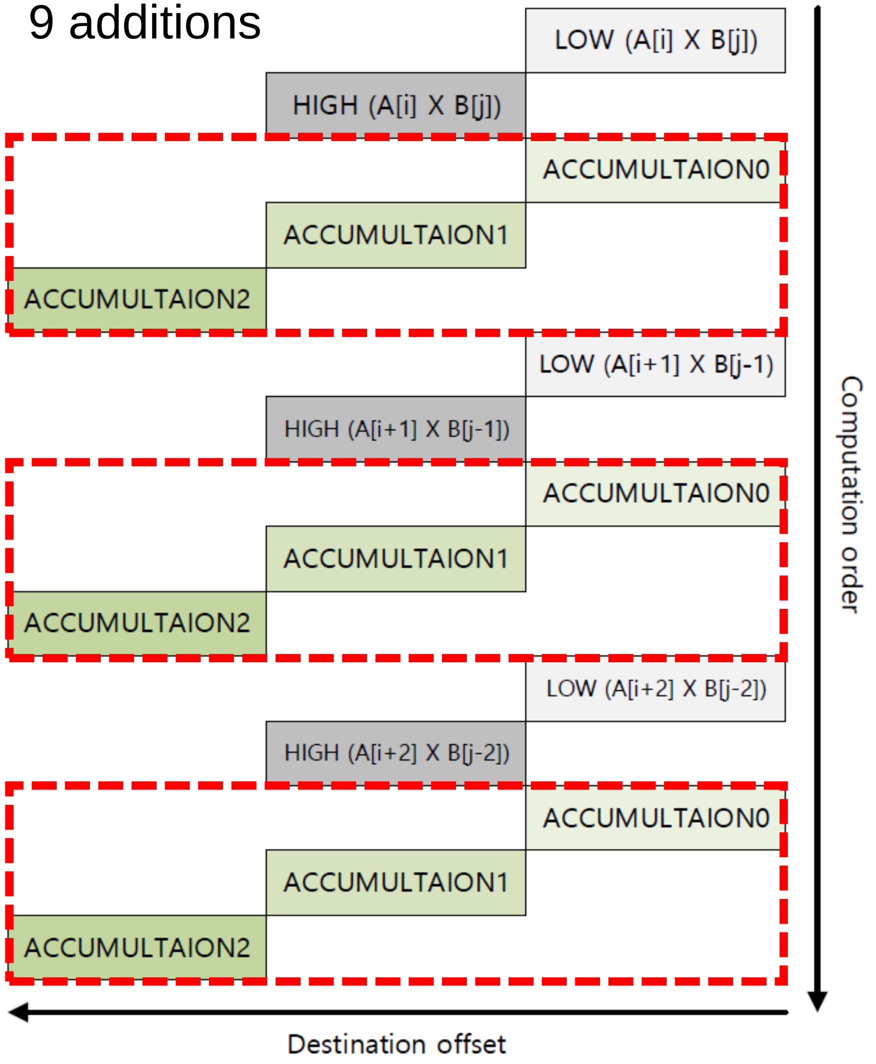


Multi-precision Multiplication (Operand Scanning)



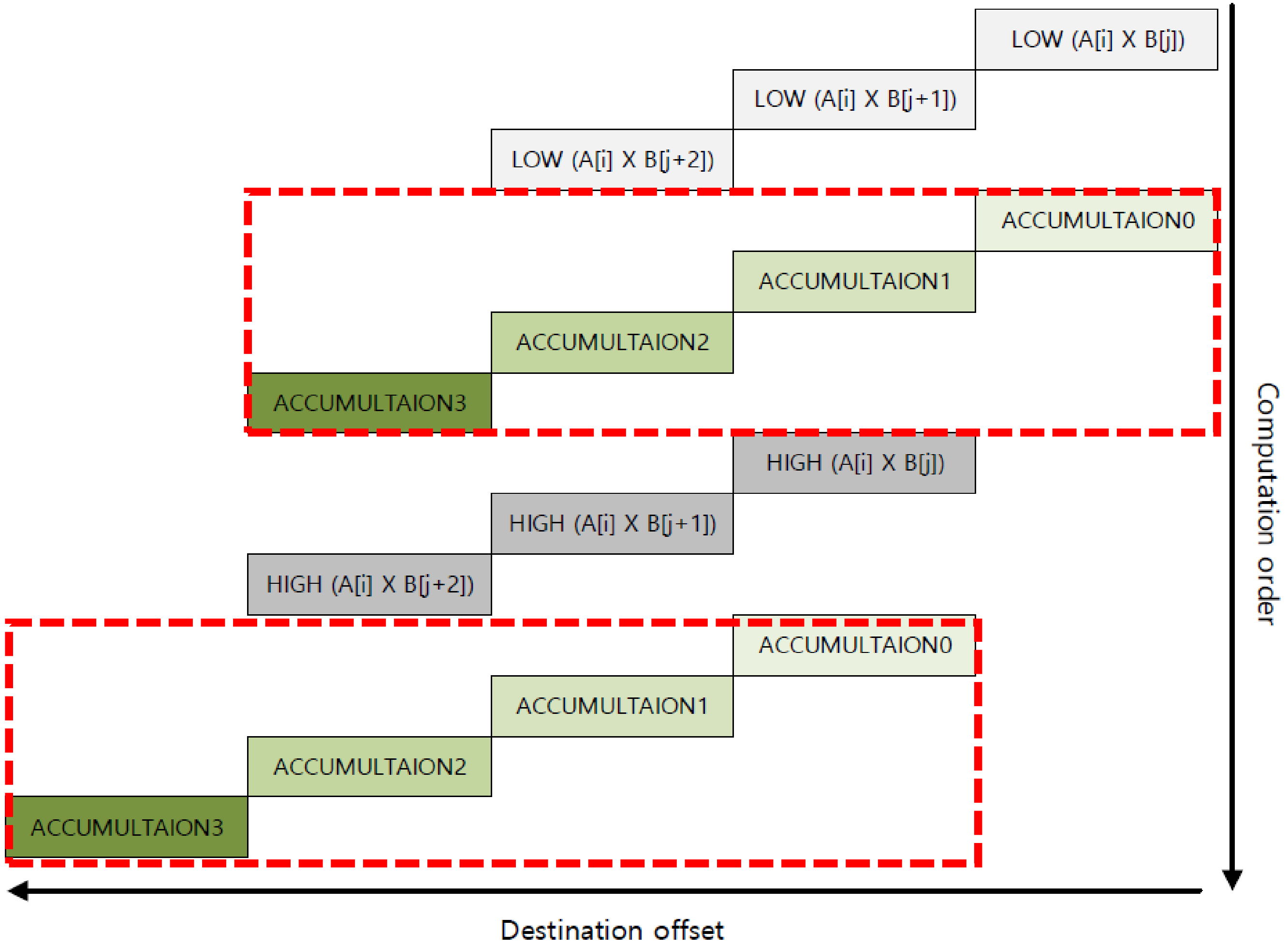
Comparison between Product-scanning & Operand-scanning

9 additions



Product-scanning

8 additions



Operand-scanning

Comparison between Product-scanning & Operand-scanning

- **Number of addition operations depending on the number of limb**
  - Operand-scanning shows lower number of addition than product-scanning

Method	Number of limb				
	3	4	5	6	7
Operand-scanning	**24**	**40**	**60**	**84**	**112**
Product-scanning	27	48	75	108	147

Karatsuba-Ofman Algorithm

- Number of partial product

School-book	Karatsuba-Ofman
N^2	$N^{\log_2 3}$

- The product $C = A \cdot B$ of two n-bit integers $A = A_L + A_H 2^{n/2}$ and $B = B_L + B_H 2^{n/2}$

- $C = A_H \cdot B_H 2^n + \left((A_L + A_H) \cdot (B_L + B_H) - A_L \cdot B_L - A_H \cdot B_H\right) 2^{n/2} + A_L \cdot B_L$

- **610-bit multiplication uses Karatsuba method**

Subtractive Karatsuba Algorithm

$$C = A_H \cdot B_H 2^n + \left((A_L + A_H) \cdot (B_L + B_H) - A_L \cdot B_L - A_H \cdot B_H\right) 2^{n/2} + A_L \cdot B_L$$

$$(A_L + A_H) \cdot (B_L + B_H) - A_L \cdot B_L - A_H \cdot B_H = A_L \cdot B_L + A_H \cdot B_H - |A_H - A_L| \cdot |B_H - B_L|$$

- **Advantage:**
  - constant size of operands (*n/2*) → fast constant-time multiplication

- **Requirement:**
  - Absolute value in two’s complement representation

Evaluation

- Multiplication implementations for **SIKEp434 and SIKEp610** are improved by **4.5x and 4.9x**, respectively.

Implementation	Language	Protocol	Timing [cc]			
			Addition	Subtraction	Multiplication	Inversion
SIKE Round 2	C	**SIKEp434**	172	129	3,110	1,648,372
This work	**ASM**		**71**	**63**	**691**	**380,711**
SIKE Round 2	C	**SIKEp610**	257	187	6,599	4,800,694
This work	**ASM**		**100**	**91**	**1,329**	**963,064**

Evaluation

- Total protocols for **SIKEp434 and SIKEp610** are also improved by **3.8x and 3.4x**, respectively.
- SIKEp434 shows the most optimal performance among SIKE curves.

Implementation	Language	Protocol	Timing [cc]	Timing [cc * 10^6]			
			Multiplication	KeyGen	Encaps	Decaps	Total
SIKE Round 2	C	SIKEp434	3,110	114	186	199	499
This work	**ASM**		**691**	**30**	**49**	**52**	**130**
Seo et al.	ASM	SIKEp503	849	38	63	67	168
SIKE Round 2	C	SIKEp610	6,599	344	634	615	1,593
This work	**ASM**		**1,329**	**99**	**183**	**183**	**465**
Seo et al.	ASM	SIKEp751	2,450	164	265	284	713

Conclusion

- **Achievements**
  - Efficient implementations of **multi-precision multiplication** on ARMv8
  - First implementation of **SIKE Round 2 curves** on ARMv8

- **Future works**
  - Further cryptography implementations (**other SIKE and PQC**)

Q & A