PQC-AMX: Accelerating Saber and FrodoKEM on the Apple M1 and M3 SoCs

Décio Luiz Gazzoni Filho (decio.gazzoni@ic.unicamp.br) Guilherme Brandão Gora Adj Arwa Alblooshi Isaac A. Canales-Martínez Jorge Chávez-Saab Julio López

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

Agenda

Introduction

- The AMX matrix multiplication accelerator
- Arithmetic for Saber and FrodoKEM
- Saber on AMX
 - Baseline polynomial multiplication and reduction
 - Polynomial multiplication from linear algebra
 - Higher-level improvements
- FrodoKEM on AMX
- Gaussian sampling
- Conclusions

3 ×

Introduction

- Modern cryptography is under threat by quantum computers
- Post-quantum cryptography: cryptosystems resistant to both classical and quantum adversaries
- Key/ciphertext size and performance a concern
- Efficient and secure implementations are a must
- Turing award lecture by Hennessy and Patterson: "Innovations like domain-specific hardware (...) will lead the way."
- AI/deep learning requires TOPS-level processing power for linear algebra (matrix multiplication)
- Many accelerators and instruction set extensions available: TPUs, GPUs, Intel AMX, ARM SME, Apple AMX, etc.

くぼ ト く ヨ ト く ヨ ト

AMX

- Apple SoCs: multiple computational resources such as the CPU, Neural Engine, GPU and AMX, the latter of which is an undocumented coprocessor mainly for matrix multiplication
- Reverse engineering by D. Johnson, P. Cawley, M. Handley
- First application to crypto: NTRU (Gazzoni Filho et al, 2024)
- Programmer's model: X,Y vector registers (8 × 64 bytes each) and Z matrix registers (64 × 64 bytes)

	X [0]	X[1]		$\mathtt{X}[n]$
Y[0]	Z[0][0] += Y[0]X[0]	Z[0][1] += Y[0]X[1]		$\mathtt{Z}[0][n] \mathrel{+}= \mathtt{Y}[0]\mathtt{X}[n]$
Y [1]	$\mathtt{Z}[1][0] \mathrel{+}= \mathtt{Y}[1]\mathtt{X}[0]$	Z[1][1] += Y[1]X[1]		$\mathbf{Z}[1][n] \mathrel{+}= \mathbf{Y}[1]\mathbf{X}[n]$
÷	÷	:	·	÷
$\mathtt{Y}[n]$	$\mathbf{Z}[n][0] \mathrel{+}= \mathbf{Y}[n]\mathbf{X}[0]$	$\mathbf{Z}[n][1] \mathrel{+}= \mathbf{Y}[n]\mathbf{X}[1]$		$\mathbf{Z}[n][n] \mathrel{+}= \mathbf{Y}[n]\mathbf{X}[n]$

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

PQC-AMX: Accelerating Saber and FrodoKEM on the Apple M1 and M3 SoCs

< 🗇 🕨 < 🖃 🕨 <

AMX instructions

Instruction Type	Mnemonics(Opcode)	Description
Loads and stores	ldx(0), ldy(1), stx(2), sty(3), ldz(4), stz(5), ldzi(6), stzi(7)	Data movement between memory and AMX registers
Extract	extrh(8), extrx(8), extrv(9), extry(9)	Data movement within the AMX register file
First generation matrix/vector	fma64(10), fms64(11), fma32(12), fms32(13), mac16(14), fma16(15), fms16(16)	Outer or pointwise products with accumulation/subtraction
Second generation matrix and vector	<pre>vecint(18), vecfp(19), matint(20), matfp(21)</pre>	Outer or pointwise products with accumulation/subtraction
Miscellaneous	<pre>set(17), clr(17), genlut(22)</pre>	Context switching, lookup tables

Matrix-mode mac16, matint: outer products

- Vector-mode mac16, vecint: pointwise vector operations
- Bytewise register addressing \Rightarrow free bytewise shifts

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

PQC-AMX: Accelerating Saber and FrodoKEM on the Apple M1 and M3 SoCs

э

AMX performance (M1)

- Note: operation counts consider multiply-accumulate (MACs)
 multiplies or additions only halves throughput
- Outer product: building block for matrix multiplication
 - 16-bit FP or 8-bit integer: 3.05 TOPS/s
 - 16-bit integer: 1.53 TOPS/s
- Vector (pointwise) operations
 - FP or 8-bit integer: 381 GOPS/s
 - 16-bit integer: 191 GOPS/s
- Superscalar (2/cycle) execution of some vector instructions on M3, doubling throughput
- NEON 16-bit MAC throughput for comparison: 204.8 GOPS/s
- Matrix operations much faster than vector operations

글 🖌 🖌 글 🛌

Image: A mathematical states and a mathem

Saber

- Based on the module-Learning with Rounding (LWR) problem
- Small matrix-vector products in $R_q = \mathbb{Z}_q[x]/(x^{256}+1)$
- Arithmetic in R_q : polynomial arithmetic modulo $q = 2^{13} < 2^{16}$, no explicit modular reduction needed
- NTT-unfriendly ring
- Different security levels by increasing module dimension

Parameter set	Sec. level	ℓ	n	q
LightSaber	1	2	256	2 ¹³
Saber	3	3	256	2 ¹³
FireSaber	5	4	256	2 ¹³

▲□▶ ▲□▶ ▲臣▶ ▲臣▶ ―臣 - のへで

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

FrodoKEM

- Unstructured lattices multiplication of "large" square $(n \times n)$ by "thin" $(n \times \overline{n})$ matrices
- Coefficient arithmetic modulo $q = 2^b \le 2^{16}$

Parameter set	Sec. level	n	q	$\overline{m} = \overline{n}$
Frodo-640	1	640	2 ¹⁵	8
Frodo-976	3	976	2 ¹⁶	8
Frodo-1344	5	1344	2 ¹⁶	8

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

PQC-AMX: Accelerating Saber and FrodoKEM on the Apple M1 and M3 SoCs

(日) (四) (三) (三)

AMX polynomial multiplication

- Basic idea applied to NTRU by Gazzoni Filho et al (2024)
- Outer products vs. polynomial multiplication parallelogram:

	a_3	a_2	a_1	a_0	Left shift							
b_0	$a_{3}b_{0}$	a_2b_0	a_1b_0	a_0b_0	<i>i</i> -th row by				a_3b_0	a_2b_0	a_1b_0	a_0b_0
b_1	a_3b_1	a_2b_1	a_1b_1	a_0b_1	$\xrightarrow{i \text{ positions}}$			a_3b_1	a_2b_1	a_1b_1	a_0b_1	
b_2	a_3b_2	a_2b_2	a_1b_2	a_0b_2			$a_{3}b_{2}$	a_2b_2	a_1b_2	a_0b_2		
b_3	a_3b_3	a_2b_3	a_1b_3	a_0b_3		a_3b_3	a_2b_3	a_1b_3	a_0b_3			

- Sum-reduction of columns to finish polynomial multiplication
- Natural AMX dimension: 64 bytes i.e. 32, 16-bit words
- Combine these basic blocks using a product scanning (columnwise) approach, delaying shifting and sum-reduction to the end of the computation

AMX polynomial multiplication

	$a_7b_0 \ a_6b_0 \ a_5b_0 \ a_4b_0 \ a_3b_0 \ a_2b_0 \ a_1b_0$	$a_{0}b_{0}$ $a_{5}b_{0}$ $a_{5}b_{0}$ $a_{4}b_{0}$ $a_{3}b_{0}$ $a_{2}b_{0}$ $a_{1}b_{0}$ $a_{0}b_{0}$
	$a_7b_1 \ a_6b_1 \ a_5b_1 \ a_4b_1 \ a_3b_1 \ a_2b_1 \ a_1b_1 \ a_0b_1$	$a_7b_1 \ a_6b_1 \ a_5b_1 \ a_4b_1 \ a_3b_1 \ a_2b_1 \ a_1b_1 \ a_0b_1$
	$a_7b_2 \ a_6b_2 \ a_5b_2 \ a_4b_2 \ a_3b_2 \ a_2b_2 \ a_1b_2 \ a_0b_2$	$a_7b_2 \ a_6b_2 \ a_5b_2 \ a_4b_2 \ a_3b_2 \ a_2b_2 \ a_1b_2 \ a_0b_2$
(n)	$a_7b_3 \ a_6b_3 \ a_5b_3 \ a_4b_3 \ a_3b_3 \ a_2b_3 \ a_1b_3 \ a_0b_3$	$a_7b_3 \ a_6b_3 \ a_5b_3 \ a_4b_3 \ a_3b_3 \ a_2b_3 \ a_1b_3 \ a_0b_3$
<i>(a)</i>	$a_7b_4 \ a_6b_4 \ a_5b_4 \ a_4b_4 \ a_3b_4 \ a_2b_4 \ a_1b_4 \ a_0b_4$	$a_7b_4 \ a_6b_4 \ a_5b_4 \ a_4b_4 \ a_3b_4 \ a_2b_4 \ a_1b_4 \ a_0b_4$
6	$a_7b_5 \ a_6b_5 \ a_5b_5 \ a_4b_5 \ a_3b_5 \ a_2b_5 \ a_1b_5 \ a_0b_5$	$a_7b_5 \ a_6b_5 \ a_5b_5 \ a_4b_5 \ a_3b_5 \ a_2b_5 \ a_1b_5 \ a_0b_5$
a_7b_6 a	$a_6b_6 \ a_5b_6 \ a_4b_6 \ a_3b_6 \ a_2b_6 \ a_1b_6 \ a_0b_6$	$a_7b_6 \ a_6b_6 \ a_5b_6 \ a_4b_6 \ a_3b_6 \ a_2b_6 \ a_1b_6 \ a_0b_6$
$a_7b_7 \ a_6b_7 \ a_6$	$a_5b_7 \ a_4b_7 \ a_3b_7 \ a_2b_7 \ a_1b_7 \ a_0b_7$	$a_7b_7 \ a_6b_7 \ a_5b_7 \ a_4b_7 \ a_3b_7 \ a_2b_7 \ a_1b_7 \ a_0b_7$
\rightarrow	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{l} +a_{1}b_{4} \ a_{4}b_{0} + a_{0}b_{4} \ a_{3}b_{0} \ a_{2}b_{0} \ a_{1}b_{0} \ a_{0}b_{0} \\ +a_{1}b_{3} \ a_{4}b_{1} + a_{0}b_{5} \ a_{3}b_{1} \ a_{2}b_{1} \ a_{1}b_{1} \ a_{0}b_{1} \\ +a_{1}b_{6} \ a_{4}b_{2} + a_{0}b_{6} \ a_{3}b_{2} \ a_{2}b_{2} \ a_{1}b_{2} \ a_{0}b_{2} \\ +a_{1}b_{7} \ a_{4}b_{3} + a_{0}b_{7} \ a_{3}b_{3} \ a_{2}b_{3} \ a_{1}b_{3} \ a_{0}b_{3} \end{array} \tag{c}$
$\xrightarrow{a_7b_6} a_7b_7 a_6b_7$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$

- イロ・ イヨ・ イヨ・ トヨー シック

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

AMX polynomial multiplication, reduction modulo $x^n + 1$

- O(n) shifts and sum-reductions vs. $O(n^2)$ outer products \Rightarrow high ratio of matrix to vector operations
- Must perform polynomial reduction modulo $x^n + 1$
- An integrated procedure decreases (!) the vector operation count, by postponing shifts and sum-reductions after the polynomial reduction
- Next slide: toy example for reduction modulo $x^{10} + 1$

イロト イボト イヨト イヨト

AMX polynomial multiplication, reduction modulo $x^n + 1$

	0	$a_{11}b_{0}$	$a_{10}b_{0}$	a_9b_0	a_8b_0	a_7b_0	a_6b_0	a_5b_0	a_4b_0	a_3b_0	a_2b_0	a_1b_0	a_0b_0
	1	$a_{11}b_1$	$a_{10}b_1$	a_9b_1	a_8b_1	a_7b_1	a_6b_1	a_5b_1	a_4b_1	a_3b_1	a_2b_1	a_1b_1	a_0b_1
	2	$a_{11}b_{2}$	$a_{10}b_2$	a_9b_2	a_8b_2	a_7b_2	a_6b_2	a_5b_2	a_4b_2	a_3b_2	a_2b_2	a_1b_2	a_0b_2
	3	$a_{11}b_{3}$	$a_{10}b_3$	a_9b_3	a_8b_3	a_7b_3	a_6b_3	a_5b_3	a_4b_3	a_3b_3	a_2b_3	a_1b_3	a_0b_3
	0	a_7b_4	$a_{6}b_{4}$	a_5b_4	a_4b_4	a_3b_4	a_2b_4	a_1b_4	a_0b_4	$-a_{9}b_{4}$	$-a_{8}b_{4}$	$-a_{7}b_{4}$	$-a_6b_4$
(n)	1	a_7b_5	a_6b_5	a_5b_5	a_4b_5	a_3b_5	a_2b_5	a_1b_5	a_0b_5	$-a_{9}b_{5}$	$-a_{8}b_{5}$	$-a_{7}b_{5}$	$-a_6b_5$
(a)	2	a_7b_6	a_6b_6	a_5b_6	a_4b_6	a_3b_6	a_2b_6	a_1b_6	a_0b_6	$-a_{9}b_{6}$	$-a_{8}b_{6}$	$-a_{7}b_{6}$	$-a_6b_6$
	3	a_7b_7	a_6b_7	a_5b_7	a_4b_7	a_3b_7	a_2b_7	a_1b_7	a_0b_7	$-a_{9}b_{7}$	$-a_{8}b_{7}$	$-a_{7}b_{7}$	$-a_{6}b_{7}$
	0	$a_{3}b_{8}$	$\frac{a_2b_8}{a_2b_8}$	a_1b_8	a_0b_8	$-a_{9}b_{8}$	$-a_{8}b_{8}$	$-a_{7}b_{8}$	$-a_{6}b_{8}$	$-a_{5}b_{8}$	$-a_{4}b_{8}$	$-a_3b_8$	$-a_{2}b_{8}$
	1	a3b9	a_2b_9	a_1b_9	a_0b_9	$-a_{9}b_{9}$	$-a_{8}b_{9}$	$-a_{7}b_{9}$	$-a_{6}b_{9}$	$-a_{5}b_{9}$	$-a_{4}b_{9}$	$-a_{3}b_{9}$	$-a_{2}b_{9}$
	2	a3b10	a2b10	a1b10	$\frac{a_0b_{10}}{a_0b_{10}}$	agbin	$\frac{a_8b_{10}}{a_8b_{10}}$	$\frac{a7b_{10}}{a7b_{10}}$	$a_{6}b_{10}$	asbin	a_4b_{10}	a3b10	a2bin
													,
	3	$a_{3}b_{11}$	a_2b_{11}	a_1b_{11}	a_0b_{11}	a_9b_{11}	a_8b_{11}	a_7b_{11}	a_6b_{11}	a_5b_{11}	a_4b_{11}	a_3b_{11}	a_2b_{11}
	3	a3b11 11	a2b11 10	a1b11 9	a0b11 8	a9b11 7	asb _{IT}	a7b11 5	а_бb₁₁ 4	a5b11 3	$\frac{a_4b_{11}}{2}$	$\frac{a_3b_{1T}}{1}$	$\frac{a_2b_{11}}{0}$
(b)	3	азb₁₁ 11	a2b11 10	$\frac{a_{T}b_{TT}}{9}$ $M_{1,0}^{(2)}$	$\frac{a_0 b_{11}}{8}$ $M_{0,0}^{(2)}$	$\frac{a_9b_{11}}{7}$ $M_{3,0}^{(1)}$	$\frac{a_8 b_{TT}}{6}$ $M_{2,0}^{(1)}$	$\frac{a_7 b_{11}}{5}$ $M_{1,0}^{(1)}$	$\frac{a_6 b_{TT}}{4}$ $M_{0,0}^{(1)}$	$\frac{a_5b_{11}}{3}$ $M_{3,0}^{(0)}$	$\frac{a_4b_{11}}{2}$ $M_{2,0}^{(0)}$	$\frac{a_3b_{11}}{1}$ $M_{1,0}^{(0)}$	$a_2 b_{11}$ 0 $M_{0,0}^{(0)}$
(b)	3	a3b11 11	$\frac{a_2b_{11}}{10}$ $M_{1,1}^{(2)}$	$a_{1}b_{11}$ 9 $M_{1,0}^{(2)}$ $M_{0,1}^{(2)}$	$a_0 b_{11}$ 8 $M_{0,0}^{(2)}$ $M_{3,1}^{(1)}$	$\frac{a_{9}b_{11}}{7}$ $M_{3,0}^{(1)}$ $M_{2,1}^{(1)}$	a_8b_{11} 6 $M^{(1)}_{2,0}$ $M^{(1)}_{1,1}$	$\frac{a_7 b_{11}}{5}$ $M_{1,0}^{(1)}$ $M_{0,1}^{(1)}$	$a_{6}b_{11}$ 4 $M_{0,0}^{(1)}$ $M_{3,1}^{(0)}$	a_5b_{11} 3 $M^{(0)}_{3,0}$ $M^{(0)}_{2,1}$	${a_4 b_{11} \over 2} \ M^{(0)}_{2,0} \ M^{(0)}_{1,1}$	a_3b_{11} 1 $M_{1,0}^{(0)}$ $M_{0,1}^{(0)}$	a_2b_{11} 0 $M_{0,0}^{(0)}$
(b)	3	$\frac{a_3b_{11}}{11}$ $M_{1,2}^{(2)}$	$a_2 b_{11}$ 10 $M_{1,1}^{(2)}$ $M_{0,2}^{(2)}$	$a_{1}b_{11}$ 9 $M_{1,0}^{(2)}$ $M_{0,1}^{(2)}$ $M_{3,2}^{(1)}$	$a_0 b_{11}$ 8 $M_{0,0}^{(2)}$ $M_{3,1}^{(1)}$ $M_{2,2}^{(1)}$	$a_{9}b_{11}$ 7 $M^{(1)}_{3,0}$ $M^{(1)}_{2,1}$ $M^{(1)}_{1,2}$	a_8b_{11} 6 $M^{(1)}_{2,0}$ $M^{(1)}_{1,1}$ $M^{(1)}_{0,2}$	$a_7 b_{11}$ 5 $M_{1,0}^{(1)}$ $M_{0,1}^{(1)}$ $M_{3,2}^{(0)}$	$a_{6}b_{11}$ 4 $M_{0,0}^{(1)}$ $M_{3,1}^{(0)}$ $M_{2,2}^{(0)}$	a_5b_{11} 3 $M_{3,0}^{(0)}$ $M_{2,1}^{(0)}$ $M_{1,2}^{(0)}$	${a_4 b_{11} \over 2} \ {M^{(0)}_{2,0} \over M^{(0)}_{1,1} \over M^{(0)}_{0,2}}$	a_3b_{11} 1 $M_{1,0}^{(0)}$ $M_{0,1}^{(0)}$	$a_2 b_{11}$ 0 $M_{0,0}^{(0)}$
(b)	(2) .,3	$a_{3}b_{11}$ 11 $M_{1,2}^{(2)}$ $M_{0,3}^{(2)}$	a_2b_{11} 10 $M_{1,1}^{(2)}$ $M_{0,2}^{(2)}$ $M_{3,3}^{(1)}$	$a_{1}b_{11}$ 9 $M_{1,0}^{(2)}$ $M_{0,1}^{(2)}$ $M_{3,2}^{(1)}$ $M_{2,3}^{(1)}$	$a_0 b_{11}$ 8 $M_{0,0}^{(2)}$ $M_{3,1}^{(1)}$ $M_{2,2}^{(1)}$ $M_{1,3}^{(1)}$	$a_{9}b_{11}$ 7 $M_{3,0}^{(1)}$ $M_{2,1}^{(1)}$ $M_{1,2}^{(1)}$ $M_{0,3}^{(1)}$	a_8b_{11} 6 $M^{(1)}_{2,0}$ $M^{(1)}_{1,1}$ $M^{(1)}_{0,2}$ $M^{(0)}_{3,3}$	$a_7 b_{11}$ 5 $M_{1,0}^{(1)}$ $M_{0,1}^{(1)}$ $M_{3,2}^{(0)}$ $M_{2,3}^{(0)}$	$a_{6}b_{11}$ 4 $M_{0,0}^{(1)}$ $M_{3,1}^{(0)}$ $M_{2,2}^{(0)}$ $M_{1,3}^{(0)}$	a_5b_{11} 3 $M^{(0)}_{3,0}$ $M^{(0)}_{2,1}$ $M^{(0)}_{1,2}$ $M^{(0)}_{0,3}$	$\begin{array}{c} \frac{a_4 b_{11}}{2} \\ M^{(0)}_{2,0} \\ M^{(0)}_{1,1} \\ M^{(0)}_{0,2} \end{array}$	a_3b_{11} 1 $M_{1,0}^{(0)}$ $M_{0,1}^{(0)}$	a_2b_{11} 0 $M_{0,0}^{(0)}$
(b)	(2) (3) (2) (2) (2)	a_3b_{11} 11 $M_{1,2}^{(2)}$ $M_{0,3}^{(2)}$ 11	$a_2 b_{11}$ 10 $M_{1,1}^{(2)}$ $M_{0,2}^{(2)}$ $M_{3,3}^{(1)}$ 10	$a_{1}b_{11}$ 9 $M_{1,0}^{(2)}$ $M_{0,1}^{(2)}$ $M_{3,2}^{(1)}$ $M_{2,3}^{(1)}$ 9	$\begin{array}{c} a_0 b_{11} \\ 8 \\ M_{0,0}^{(2)} \\ M_{3,1}^{(1)} \\ M_{2,2}^{(1)} \\ M_{1,3}^{(1)} \\ 8 \end{array}$	$\begin{array}{c} \frac{agb_{11}}{7} \\ M^{(1)}_{3,0} \\ M^{(1)}_{2,1} \\ M^{(1)}_{1,2} \\ M^{(1)}_{0,3} \\ 7 \end{array}$	a_8b_{11} 6 $M^{(1)}_{2,0}$ $M^{(1)}_{1,1}$ $M^{(1)}_{0,2}$ $M^{(0)}_{3,3}$ 6	$a_7 b_{11}$ 5 $M_{1,0}^{(1)}$ $M_{0,1}^{(1)}$ $M_{3,2}^{(0)}$ $M_{2,3}^{(0)}$ 5	$\begin{array}{c} a_{6}b_{11} \\ 4 \\ M^{(1)}_{0,0} \\ M^{(0)}_{3,1} \\ M^{(0)}_{2,2} \\ M^{(0)}_{1,3} \\ 4 \end{array}$	a_5b_{11} 3 $M_{3,0}^{(0)}$ $M_{2,1}^{(0)}$ $M_{1,2}^{(0)}$ $M_{0,3}^{(0)}$ 3	a_4b_{11} 2 $M_{2,0}^{(0)}$ $M_{1,1}^{(0)}$ $M_{0,2}^{(0)}$ 2	$\frac{a_3b_{11}}{1}$ $M_{1,0}^{(0)}$ $M_{0,1}^{(0)}$ 1	$a_{2}b_{11}$ 0 $M_{0,0}^{(0)}$
(b)	3 (2) ,3 2	$a_{3}b_{11}$ 11 $M_{1,2}^{(2)}$ $M_{0,3}^{(2)}$ 11	$\frac{a_2b_{TT}}{10}$ $M_{1,1}^{(2)}$ $M_{0,2}^{(2)}$ $M_{3,3}^{(1)}$ 10	$\begin{array}{c} \frac{a_{\mathrm{I}}b_{\mathrm{II}}}{9} \\ M^{(2)}_{1,0} \\ M^{(2)}_{0,1} \\ M^{(2)}_{3,2} \\ M^{(1)}_{2,3} \\ 9 \\ M^{(2)}_{1,0} \end{array}$	$\begin{array}{c} a_{0}b_{11}\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\$	$\begin{array}{c} agb_{11} \\ 7 \\ M^{(i)}_{3,0} \\ M^{(1)}_{2,1} \\ M^{(1)}_{1,2} \\ M^{(1)}_{0,3} \\ 7 \\ M^{(i)}_{3,0} \end{array}$	a_8b_{11} 6 $M_{2,0}^{(1)}$ $M_{1,1}^{(1)}$ $M_{0,2}^{(1)}$ $M_{3,3}^{(0)}$ 6 $M_{2,0}^{(1)}$	$\begin{array}{c} a_7 b_{11} \\ 5 \\ M_{1,0}^{(1)} \\ M_{0,1}^{(1)} \\ M_{3,2}^{(0)} \\ M_{2,3}^{(0)} \\ 5 \\ M_{1,0}^{(1)} \end{array}$	$\begin{array}{c} a_{6}b_{11} \\ 4 \\ M_{0,0}^{(1)} \\ M_{3,1}^{(0)} \\ M_{2,2}^{(0)} \\ M_{1,3}^{(0)} \\ 4 \\ M_{0,0}^{(1)} \end{array}$	$a_{5}b_{11}$ 3 $M_{3,0}^{(0)}$ $M_{2,1}^{(0)}$ $M_{1,2}^{(0)}$ $M_{0,3}^{(0)}$ 3 $M_{3,0}^{(0)}$	a_4b_{11} 2 $M_{2,0}^{(0)}$ $M_{1,1}^{(0)}$ $M_{0,2}^{(0)}$ 2 $M_{2,0}^{(0)}$	$\frac{a_3b_{11}}{1}$ $M_{1,0}^{(0)}$ $M_{0,1}^{(0)}$ 1 $M_{1,0}^{(0)}$	$a_{2}b_{11}$ 0 $M_{0,0}^{(0)}$ 0 $M_{0,0}^{(0)}$
(b) <u>M</u> 1 (c)	3 (2) (3) 2	$a_{3}b_{11}$ 11 $M_{1,2}^{(2)}$ $M_{0,3}^{(2)}$ 11	$\frac{a_2b_{11}}{10}$ $M_{1,1}^{(2)}$ $M_{0,2}^{(1)}$ $M_{3,3}^{(1)}$ 10	$\begin{array}{c} a_{\rm I} b_{\rm II} \\ g \\ M_{1,0}^{(2)} \\ M_{0,1}^{(2)} \\ M_{3,2}^{(1)} \\ M_{2,3}^{(1)} \\ g \\ M_{1,0}^{(2)} \\ M_{0,1}^{(2)} \end{array}$	$\begin{array}{c} a_{0}b_{\mathrm{TT}} \\ 8 \\ M_{0,0}^{(2)} \\ M_{3,1}^{(1)} \\ M_{1,3}^{(2)} \\ 8 \\ M_{0,0}^{(2)} \\ M_{3,1}^{(2)} \end{array}$	$\begin{array}{c} agb_{11} \\ 7 \\ M_{3,0}^{(1)} \\ M_{2,1}^{(1)} \\ M_{1,2}^{(1)} \\ M_{0,3}^{(1)} \\ 7 \\ M_{3,0}^{(1)} \\ M_{2,1}^{(1)} \end{array}$	$\begin{array}{c} a_8 b_{111} \\ \hline 6 \\ M_{2,0}^{(1)} \\ M_{1,1}^{(1)} \\ M_{0,2}^{(0)} \\ M_{3,3}^{(0)} \\ \hline 6 \\ M_{2,0}^{(1)} \\ M_{1,1}^{(1)} \end{array}$	$\begin{array}{c} a_7 b_{\rm TT} \\ 5 \\ M_{1,0}^{(1)} \\ M_{0,1}^{(1)} \\ M_{3,2}^{(0)} \\ M_{2,3}^{(0)} \\ 5 \\ M_{1,0}^{(1)} \\ M_{0,1}^{(1)} \end{array}$	$\begin{array}{c} a_{5}b_{11} \\ 4 \\ M_{0,0}^{(1)} \\ M_{3,1}^{(0)} \\ M_{2,2}^{(0)} \\ M_{1,3}^{(0)} \\ 4 \\ M_{0,0}^{(1)} \\ M_{3,1}^{(0)} \end{array}$	$a_{5}b_{11}$ 3 $M_{3,0}^{(0)}$ $M_{2,1}^{(0)}$ $M_{0,3}^{(0)}$ 3 $M_{3,0}^{(0)}$ $M_{2,1}^{(0)}$	$\begin{array}{c} a_4 b_{11} \\ 2 \\ M_{2,0}^{(0)} \\ M_{1,1}^{(0)} \\ M_{0,2}^{(0)} \end{array}$	$a_{3}b_{11}$ 1 $M_{1,0}^{(0)}$ $M_{0,1}^{(0)}$ 1 $M_{1,0}^{(0)}$ $M_{0,1}^{(0)}$	a_2b_{11} 0 $M_{0,0}^{(0)}$ 0 $M_{0,0}^{(0)}$ $-M_{1,1}^{(0)}$
(b) <i>M</i> 1 (c)	3 (2) (3) 2	$a_{3}b_{11}$ 11 $M_{1,2}^{(2)}$ $M_{0,3}^{(2)}$ 11	a_2b_{11} 10 $M_{1,1}^{(2)}$ $M_{0,2}^{(2)}$ $M_{3,3}^{(1)}$ 10	$\begin{array}{c} a_{\rm I} b_{\rm II} \\ 9 \\ M_{1,0}^{(2)} \\ M_{0,1}^{(2)} \\ M_{3,2}^{(1)} \\ M_{2,3}^{(1)} \\ 9 \\ M_{1,0}^{(2)} \\ M_{0,1}^{(2)} \\ M_{3,2}^{(2)} \end{array}$	$\begin{array}{c} a_{0}b_{11}\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\$	$\begin{array}{c} a_{3}b_{11}\\ \hline 7\\ M_{3,0}^{(1)}\\ M_{2,1}^{(1)}\\ M_{1,2}^{(1)}\\ M_{0,3}^{(1)}\\ \hline 7\\ M_{3,0}^{(1)}\\ M_{2,1}^{(1)}\\ M_{1,2}^{(1)} \end{array}$	$\begin{array}{c} \frac{a_8b_{11}}{6} \\ M^{(1)}_{2,0} \\ M^{(1)}_{1,1} \\ M^{(1)}_{0,2} \\ M^{(0)}_{3,3} \\ 6 \\ M^{(1)}_{2,0} \\ M^{(1)}_{1,1} \\ M^{(1)}_{0,2} \end{array}$	$\begin{array}{c} a_7 b_{11} \\ 5 \\ M_{1,0}^{(1)} \\ M_{0,1}^{(1)} \\ M_{3,2}^{(0)} \\ M_{2,3}^{(0)} \\ 5 \\ M_{1,0}^{(1)} \\ M_{0,1}^{(1)} \\ M_{3,2}^{(0)} \end{array}$	$\begin{array}{c} a_{6}b_{11} \\ 4 \\ M_{0,0}^{(1)} \\ M_{3,1}^{(0)} \\ M_{2,2}^{(0)} \\ M_{1,3}^{(0)} \\ 4 \\ M_{0,0}^{(0)} \\ M_{3,1}^{(0)} \\ M_{2,2}^{(0)} \end{array}$	$\begin{array}{c} \frac{a_5b_{11}}{3}\\ M^{(0)}_{3,0}\\ M^{(0)}_{2,1}\\ M^{(0)}_{1,2}\\ M^{(0)}_{0,3}\\ 3\\ M^{(0)}_{3,0}\\ M^{(0)}_{2,1}\\ M^{(0)}_{1,2} \end{array}$	$\begin{array}{c} \frac{a_4b_{11}}{2} \\ M^{(0)}_{2,0} \\ M^{(0)}_{1,1} \\ M^{(0)}_{0,2} \end{array}$	$\begin{array}{c} a_{3}b_{11}\\ 1\\ M_{1,0}^{(0)}\\ M_{0,1}^{(0)}\\ \end{array}$ $\begin{array}{c} 1\\ M_{1,0}^{(0)}\\ M_{0,1}^{(0)}\\ -M_{1,2}^{(2)}\\ \end{array}$	a_2b_{11} 0 $M_{0,0}^{(0)}$ 0 $M_{0,0}^{(0)}$ $-M_{1,1}^{(2)}$ $-M_{0,2}^{(2)}$

지나 지역에 지역 지역 지역 지역 가지 않는 것이다.

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

- The previous techniques were applied to NTRU by Gazzoni Filho et al (2024), which uses a similar (implementation-wise) ring, but modulo xⁿ - 1 rather than xⁿ + 1
- Alone it was competitive, but not a definite improvement, over the state-of-the-art (Becker et al, 2021)
- We propose an alternative technique, casting polynomial multiplication in the language of linear algebra

(E)

 Polynomial multiplication modulo x²⁵⁶ + 1 can be viewed as a multiplication of a skew-circulant matrix by a vector:

	a ₀	- <i>a</i> ₂₅₅	- <i>a</i> ₂₅₄	•••	$-a_2$	$-a_1$		b_0
Mv =	a_1	a_0	- <i>a</i> ₂₅₅	•••	$-a_3$	$-a_2$		b_1
	a ₂	a_1	<i>a</i> 0	•••	$-a_4$	$-a_3$		<i>b</i> ₂
	÷	÷	·	÷	÷		•	:
	a ₂₅₄	a ₂₅₃	a ₂₅₂	•••	<i>a</i> 0	- <i>a</i> 255		b ₂₅₄
	a ₂₅₅	a ₂₅₄	a ₂₅₃	•••	a_1	a ₀		b ₂₅₅

Can decompose M into 8 × 8 block Toeplitz matrix, and v into block 8 × 1 vector

We can write Mv as

- <ロ> <個> <ヨ> <ヨ> (日>)

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

If we "reshape" the 256 × 1 result into a 32 × 8 matrix, it can be computed as:

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

Outer level Saber M-V product improvements

- Saber performs matrix-vector products, with elements in Z_q/(x²⁵⁶ + 1): A^Ts + h and As' + h in PKE key generation and encryption, respectively (A is ℓ × ℓ for ℓ ∈ {2,3,4})
- We noticed that (example for PKE encryption and $\ell = 2$):

$$\mathbf{As}' = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} s'_0 \\ s'_1 \end{pmatrix} = s'_0 \begin{pmatrix} A_{00} \\ A_{10} \end{pmatrix} + s'_1 \begin{pmatrix} A_{01} \\ A_{11} \end{pmatrix},$$

- Allows batching multiplications by s_0' and s_1'
- We reuse the technique of the previous slides to write As' as a sum of 8 multiplications of 32 × 32 by 32 × 8ℓ matrices, increasing AMX utilization (full utilization for ℓ = 4)
- One of the reviewers pointed out an analogous trick for the baseline (polymul) implementation, which sped it up nicely

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

Saber performance results (kilocycles, M3 only)

Best-performing implementation and memory allocation in each case

Sec. level	Туре	Keygen	Encaps	Decaps	M-V mul
1	NEON	19.1	26.3	25.7	3.96
1	AMX-NTRU	18.5	25.6	24.5	3.28
	AMX-TMVP	17.5	24.3	23.2	2.32
NEON/AN	AX-TMVP (×)	1.09	1.08	1.11	1.71
2	NEON	31.4	40.2	40.3	7.52
5	AMX-NTRU	32.3	40.9	40.9	7.43
	AMX-TMVP	28.4	36.5	36.4	3.62
NEON/AN	AX-TMVP (×)	1.11	1.10	1.11	2.08
5	NEON	48.3	59.7	59.8	12.1
5	AMX-NTRU	51.3	62.2	62.4	13.3
	AMX-TMVP	42.9	53.2	53.5	4.83
NEON/AN	/X-TMVP (×)	1.12	1.12	1.12	2.51

- イロト (四) (日) (日) (日) (日) (の)

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

FrodoKEM on AMX

- FrodoKEM is based on matrix multiplication and should be a nice match to AMX, but we ran into some issues
- Data layout mismatches require transpositions (easy but not free with AMX); can't generate A^T directly
- AS in particular requires two transposes, SA only one
- AMX is underutilized due to $n \times 8$ dimension of **S** and **S**'
- Use batching by a factor of 4 in encapsulation/decapsulation
- Given there is no publicly available FrodoKEM implementation in NEON, we wrote our own to have a fair baseline

< 同 > < 三 > < 三 >

Gaussian sampling

- Random sampling hard to perform in constant-time for some distributions (e.g. FrodoKEM's rounded continuous Gaussian)
- Inversion sampling using a table, requires an inefficient full scan to implement in constant-time
- genlut: peculiar instruction with generate and lookup modes
- Two input (vector) registers, a source **S** and a table **T**
- For sorted tables, generate mode works like a search
- Lookup similar to Intel pshufb, NEON tbl instructions
- In either case, works in parallel on all 32 lanes (and is fast)
- Trivial changes to FrodoKEM sampling tables allowed us to repurpose genlut for table-based inversion sampling, with excellent performance; experiments suggest it is constant-time

FrodoKEM performance results (kilocycles, M3 only)

Best-performing implementation and memory allocation in each case

Sec. level	Туре	Keygen	Encaps	Decaps	E 4 \times	D 4 \times	Sampling
1	Opt	558	669	641	1755	1755	4.59
1	NEON	468	561	532	1395	1387	4.35
	AMX	414	494	447	907	905	0.84
NEON/AI	MX (×)	1.13	1.14	1.19	1.54	1.53	5.18
2	Opt	1220	1310	1255	3264	3098	5.99
5	NEON	940	1070	1005	2594	2419	5.65
	AMX	839	930	845	1555	1381	1.28
NEON/AI	MX (×)	1.12	1.15	1.19	1.67	1.75	4.41
F	Opt	1931	2156	2061	5807	5569	5.48
5	NEON	1573	1766	1681	4249	4004	5.09
	AMX	1396	1500	1388	2352	2101	1.76
NEON/AI	MX (×)	1.13	1.18	1.21	1.81	1.91	2.88

- イロ・ イヨ・ イヨ・ イロ・

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

FrodoKEM performance results (kilocycles, M3 only)

Best-performing implementation and memory allocation in each case

Sec. level Type		Α	S + E	S′	A + E	$\mathbf{S}'\mathbf{A} + \mathbf{E} \; 4 imes$	
		Full	Mat mul	Full	Mat mul	Full	Mat mul
1	Opt	354	192	345	189	924	745
T	NEON	263	118	254	108	614	452
	AMX	226	77.7	197	52.4	211	64.1
NEON/AI	MX (×)	1.16	1.52	1.29	2.05	2.91	7.05
2	Opt	880	418	824	384	2001	1533
3	NEON	600	247	586	247	1423	1001
	AMX	530	181	461	125	485	148
NEON/AI	(×) XN	1.13	1.37	1.27	1.97	2.93	6.78
F	Opt	1479	808	1474	766	4000	3111
5	NEON	1116	465	1113	471	2690	1942
	AMX	970	328	855	229	887	252
NEON/AI	(×) XN	1.15	1.42	1.30	2.06	3.03	7.70
					• • • • • • • • • • • • • • • • • • •	P → < E →	< 差→ 一差

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

Conclusions and future work

- AMX suitable for polynomial multiplication, but must ensure high matrix/vector operation ratio
- Current cryptosystems are best suited to CPU implementations; designers may wish to revisit parameter choices to favor matrix multiplication accelerators
- Symmetric primitives becoming the bottleneck of PQCFuture work:
 - Application to NTT-based schemes (e.g. Kyber, Dilithium)
 - Application of genlut to other schemes

< ロ > < 同 > < 三 > < 三 >

Thank you!

Questions?

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

Memory performance issues

- Performance anomalies observed in NTRU (and Saber), especially in the M1
- Observation by M. Handley that concurrent AMX/CPU memory accesses block; not only for the same addresses or even the same cache line, but rather a full memory page
- Arrays of polynomial coefficients in reference/optimized implementation are allocated on the stack, right next to other variables used by the CPU for other routines
- Solution: allocate arrays using mmap (one page per array)

Saber PKE

Algorithm

II.1

Saber.PKE.KeyGen()

Input: None Output: Key pair (pk, sk)1: $seed_{\mathbf{A}} \leftarrow \mathcal{U}(\{0, 1\}^{256})$ 2: $\mathbf{A} \leftarrow \operatorname{gen}(seed_{\mathbf{A}}) \in \mathbb{R}_q^{l \times l}$ 3: $r \leftarrow \mathcal{U}(\{0, 1\}^{256})$ 4: $\mathbf{s} \leftarrow \beta_{\mu}(\mathbb{R}_q^{l \times 1}; r)$ 5: $\mathbf{b} \leftarrow ((\mathbf{A}^{\mathsf{Ts}} + \mathbf{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in \mathbb{R}_p^{l \times 1}$ 6: return $(pk := (seed_{\mathbf{A}}, \mathbf{b}), sk := \mathbf{s})$

Algorithm

II.2

Saber.PKE.Dec(sk, c)

Input: Secret key sk, ciphertext cOutput: Message m'1: $v \leftarrow \mathbf{b'}^{\mathsf{T}}(\mathbf{s} \mod p) \in R_p$ 2: $m' \leftarrow ((v - 2^{e_p - e_T}c_m + h_2) \mod p) \gg (e_p - 1) \in R_2$ 3: return m' Algorithm

П.3

イロト イヨト イヨト イヨト

Saber.PKE.Enc(pk, m; r)

Input: Public key pk, message $m \in R_2$, optional randomness rOutput: Cipherext c1: $\mathbf{A} \leftarrow \text{gen}(seed_{\mathbf{A}}) \in R_q^{l \times l}$ 2: if r is not specified then 3: $r \leftarrow \mathcal{U}(\{0,1\}^{266})$ 4: $\mathbf{s}' \leftarrow \beta_{\mu}(R_q^{l \times 1}; r)$ 5: $\mathbf{b}' \leftarrow ((\mathbf{As}' + \mathbf{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in R_p^{l \times 1}$ 6: $v' \leftarrow \mathbf{b}^{\mathsf{T}}(\mathbf{s}' \mod p) \in R_p$ 7: $c_m \leftarrow (v' + h_1 - 2^{\epsilon_p - 1}m \mod p) \gg (\epsilon_p - \epsilon_T) \in R_T$ 8: return $c := (c_m, \mathbf{b}')$

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

FrodoKEM PKE

Algorithm

II.4

FrodoPKE.KeyGen()

Input: None Output: Key pair (pk, sk)1: $seed_{\mathbf{A}} \leftarrow \mathcal{U}(\{0, 1\}^{l_{\mathbf{A}}})$ 2: $\mathbf{A} \leftarrow Gen(seed_{\mathbf{A}})$ 3: $seed_{\mathbf{SE}} \leftarrow \mathcal{U}(\{0, 1\}^{l_{\mathbf{SE}}})$ 4: $\mathbf{r} \leftarrow SK(0x5F||seed_{\mathbf{SE}}, 2n\overline{n} \cdot 16)$ 5: $\mathbf{S}^{\mathsf{T}} \leftarrow SM(\mathbf{r}[0: n\overline{n} - 1], \overline{n}, n)$ 6: $\mathbf{E} \leftarrow SM(\mathbf{r}[n\overline{n} : 2n\overline{n} - 1], n, \overline{n})$ 7: $\mathbf{B} = \mathbf{AS} + \mathbf{E}$ 8: return $(pk := (seed_{\mathbf{A}}, \mathbf{B}), sk := \mathbf{S}^{\mathsf{T}})$

Algorithm

II.5

FrodoPKE.Dec(sk, c)

Input: Secret key sk, ciphertext cOutput: Message m'1: $\mathbf{M} = \mathbf{C}_2 - \mathbf{C}_1 \mathbf{S}$ 2: return $m' := Decode(\mathbf{M})$ Algorithm FrodoPKE.Enc(pk, m, r)

II.6

Input: Public key pk, message mOutput: Ciphetext c1: $\mathbf{A} \leftarrow Gen(seed_{\mathbf{A}})$ 2: $seed_{\mathbf{SE}} \leftarrow \mathcal{U}(\{0,1\}^{1}\mathbf{SE})$ 3: $\mathbf{r} \leftarrow SK(0\times96||seed_{\mathbf{SE}},(2\overline{m}n + \overline{mn}) \cdot 16)$ 4: $\mathbf{S}' \leftarrow SM(\mathbf{r}[0:\overline{m}n - 1],\overline{m},n)$ 5: $\mathbf{E}' \leftarrow SM(\mathbf{r}[\overline{m}n : 2\overline{m}n - 1],\overline{m},n)$ 6: $\mathbf{E}'' \leftarrow SM(\mathbf{r}[2\overline{m}n : 2\overline{m}n + \overline{mn} - 1],\overline{m},\overline{n})$ 7: $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'; \mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$ 8: return $c := (\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + Encode(m))$

Image: A math a math

Gazzoni Filho, Brandão, Adj, Alblooshi, Canales-Martínez, Chávez-Saab, López

PQC-AMX: Accelerating Saber and FrodoKEM on the Apple M1 and M3 SoCs

э