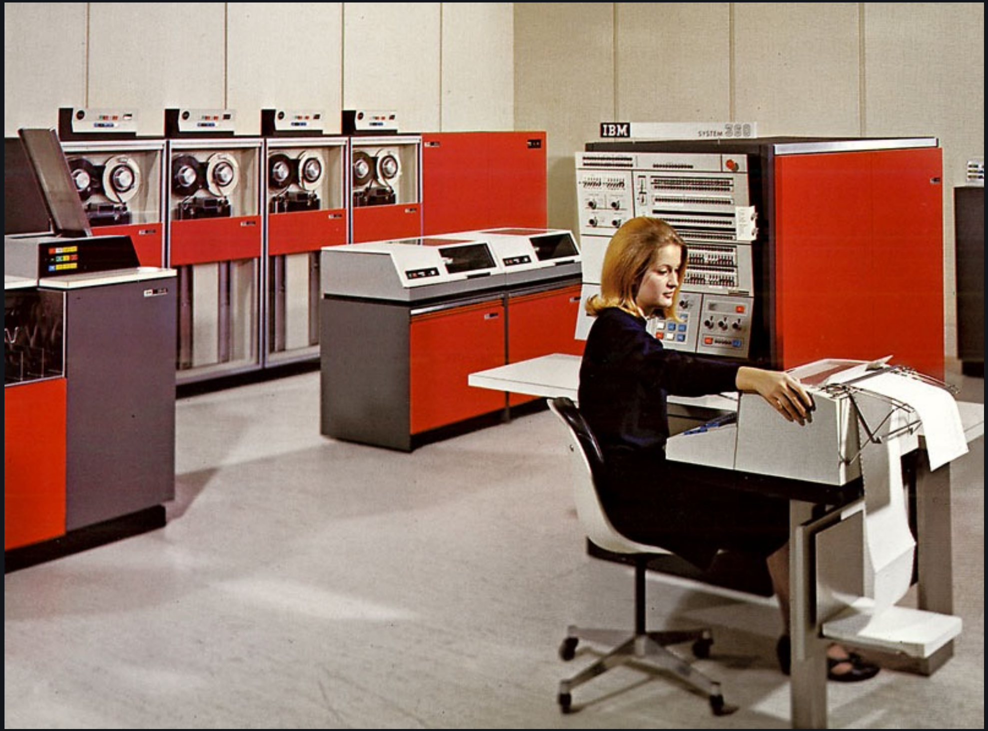
Fast Quantum-Safe Cryptography on IBM Z

Basil Hess, Jonathan Bradbury
IBM Research Europe
IBM Systems

NIST Third PQC Conference, 9 June 2021

What most people think a mainframe is



What a mainframe is today: IBM z15

92

of the top 100
worldwide banks

10

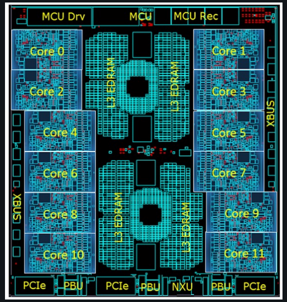
out of 10 of the world's
largest insurers

23

of the top 25
US retailers

23

out of 25 of the world's
largest airlines



BASIC FEATURES

64-bit CPUs

Big-Endian

CISC architecture

Application compatible back to IBM 360

12-core CPU chip

5.2 GHz clock frequency

Large Caches

- 128K L1
- 8M L2
- 256M Shared L3

IBM Z - Cryptographic Acceleration

Symmetric Ciphers

- Algorithms
  - DES
  - TDES (2 and 3 key)
  - **AES128**
  - **AES192**
  - **AES256**
- Block Modes
  - ECB
  - CBC
  - CFB
  - CTR
  - OFB
  - XTS
  - GCM (AEAD)

Hashing/XOF

- SHA1
- SHA256
- SHA512
- **SHA3-256**
- **SHA3-384**
- **SHA3-512**
- **SHAKE128**
- **SHAKE256**
- GHASH

Random Number Generation

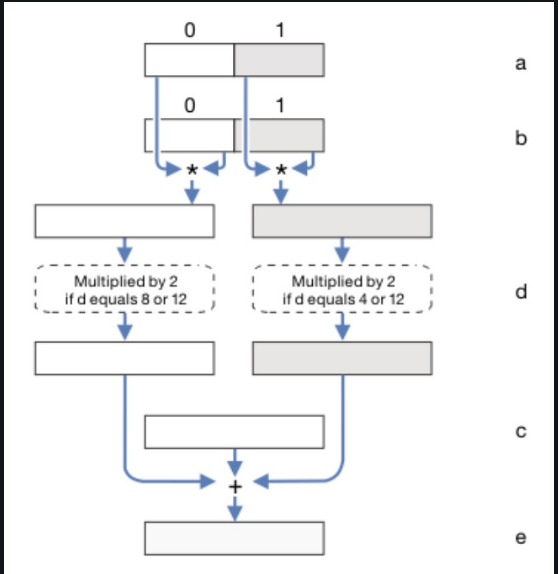
- SP800-90A Hash DRBG using SHA512
- SP800-90B True Random Number

IBM Z - SIMD Vector Instructions

32 128-bit vector registers

VMSL (Vector Multiply Sum Logical)

- Two 56-bit multiplications
- Optional multiplication by 2
- Full 128-bit addition



VA (Vector Add), VS (Vector Subtract)

- Full 128-bit unsigned addition / subtraction
- With and without carry/borrow in/out

VML (Vector Multiply Low)

VMH (Vector Multiply High)

VPERM (Vector Permute)

VSLDB (Vector Shift Left)

VMRL (Vector Merge)

ANDC (Vector And with Complement)

Optimizing SIKE and Dilithium on Z

SIKE

ISOGENY-BASED KEM

- Finite field arithmetic (F_p)
  - Multiplication
  - Montgomery reduction using special form of p
  - Addition, Subtraction
- Quadratic extension field arithmetic (F_{p^2})
- Elliptic curve arithmetic
- Isogeny computation
- SIDH/SIKE

Primes from 434-bit to 751-bit length

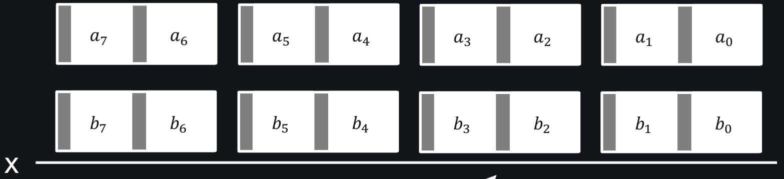
DILITHIUM

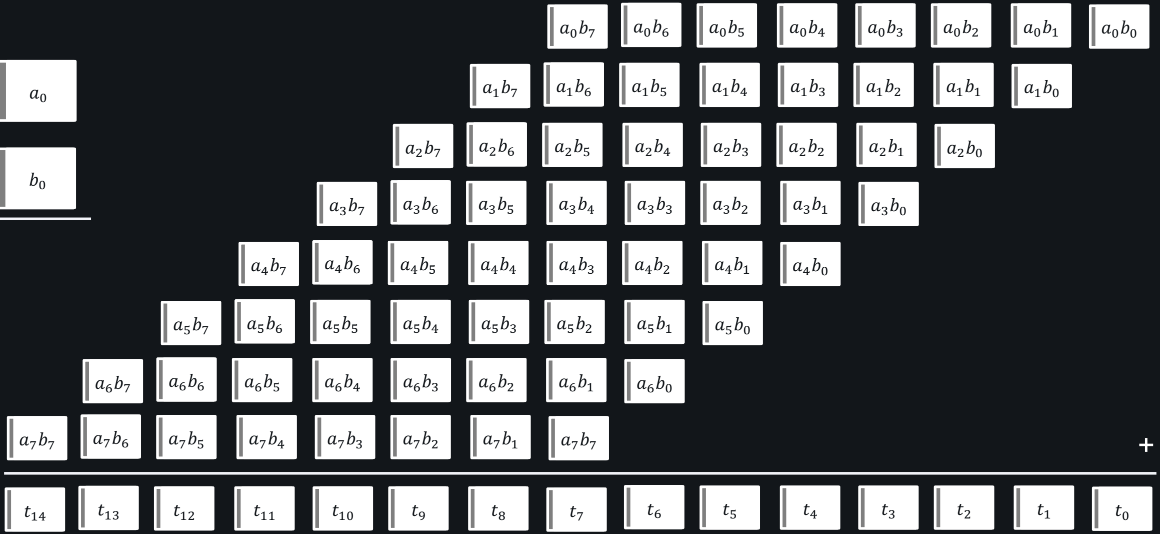
LATTICE-BASED SIGNATURE SCHEME

- Modular Multiplication
- NTT for arithmetic in polynomial ring
- Generating long random sequences from seeds: using SHAKE or AES256-CTR
- Sampling

Polynomial ring $Z_q[X]/(X^{256} + 1)$
$q = 2^{23} - 2^{13} + 1$

SIKE – F_p optimizations





Multiplication

1. Step
Limbification: Split element to 2x56-bit limbs stored in 128-bit vectors (Load, VPERM and VSLD)

2. Step
Multiplication using Comba's method: VMSL, VADD
$t_0, \ldots, t_{14}$ each at least 112 bit (for SIKEp434)

3. Step
Normalization
$t_0, \ldots, t_{14}$ (> 112-bit to 56-bit limbs)
(VADD and VSLD)

4. Step
Delimbification
56-bit limbs to 64-bit digits
(Store)

Squaring benefits from many multiplications appearing twice, which is free with VMSL

Montgomery ***reduction*** uses multiplication by $p+1$, which is optimized because λ least significant 56 bit limbs in SIKE are zero (e.g. $\lambda_{p434} = 3$)

Addition / Subtraction using 128-bit addition / subtraction chain

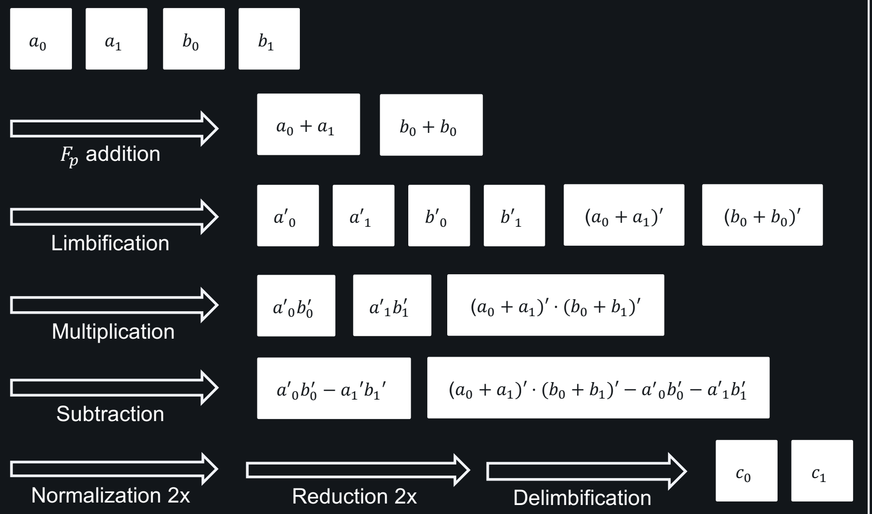
Speedup compared to unrolled plain C version:
3.0 x – 3.8 x (multiplication), 3.0 x – 4.2 x (reduction)

SIKE - F_{p^2} optimizations

Quadratic extension field as $F_{p^2} = F_p(i)$ with $i^2 + 1 = 0$

MULTIPLICATION

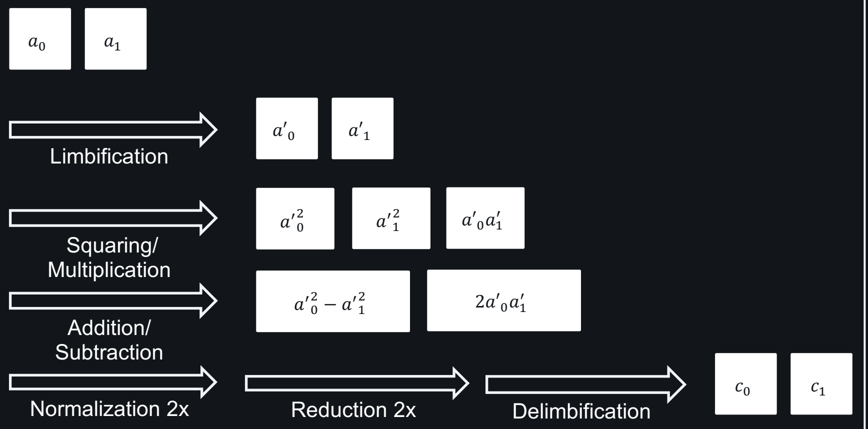
$$a \cdot b = c = (\mathrm{a}_0\mathrm{b}_0 - \mathrm{a}_1\mathrm{b}_1) + \big((\mathrm{a}_0 + \mathrm{a}_1)(\mathrm{b}_0 + \mathrm{b}_1) - \mathrm{a}_0\mathrm{b}_0 - \mathrm{a}_1\mathrm{b}_1\big) \cdot i$$



Speedup to simple version: 1.37 x (p434) to 1.59 x (p751)

SQUARING

$$a^2 = (a_0^2 - a_1^2) + (2a_0a_1) \cdot i$$



Speedup to simple version: 1.15 x (p434) to 1.22 x (p751)

SIKE results

Implementation based on SIKE optimized library (version 3.3)

Overall speedup compared to baseline implementation: factor 4 to 5.

Fastest reported performance metrics compared to SIKE 3rd round optimized/additional implementations (except SIKEp503).

Speedup increases with larger parameter sizes, and better 56-bit limb utilization (best in SIKEp434 and SIKEp610).

Scheme	KeyGen	Encaps	Decaps	total (Encaps + Decaps)
SIKEp434				
Portable C	22'771	36'807	39'089	75'897
This work	5'233 (1.01 ms)	8'676 (1.67 ms)	9'141 (1.76 ms)	17'818 (3.43 ms)
Speedup	4.4 x	4.2 x	4.3 x	4.3 x
SIKEp503				
Portable C	34'442	57'364	60'663	118'028
This work	8'200 (1.58 ms)	13'915 (2.68 ms)	14'763 (2.84 ms)	28'667 (5.51 ms)
Speedup	4.2 x	4.1 x	4.1 x	4.1 x
SIKEp610				
Portable C	61'783	113'745	114'270	228'015
This work	12'428 (2.39 ms)	23'338 (4.49 ms)	23'400 (4.50 ms)	46'738 (8.99 ms)
Speedup	5.0 x	4.9 x	4.9 x	4.9 x
SIKEp751				
Portable C	110'838	179'540	193'048	372'589
This work	21'908 (4.21 ms)	37'700 (7.25 ms)	37'560 (7.22 ms)	75'260 (14.47 ms)
Speedup	5.1 x	4.8 x	5.1 x	5.0 x

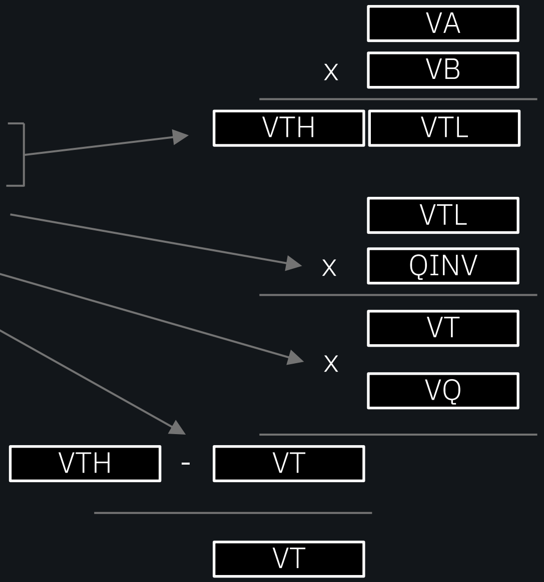
Performance in 10^3 cycles, on IBM z15 LPAR, 5.2 GHz. Linux on Z

Dilithium: Modular Multiplication and NTT

Modular multiplication $VA \cdot VB$ with centered reduction to range $-\frac{q-1}{2} \leq r' \leq \frac{q-1}{2}$.

Vectors VA, VB, and VT contain four 32-bit elements each.

```
VMLF  VTL, VA, VB
VMHF  VTH, VA, VB
VMLF  VT,  VTL, VQINV
VMHF  VT,  VT,  VQ
VSF   VT,  VT,  VTH
```



Modular multiplication used in NTT and inverse-NTT, possible to perform 4 levels of NTT without reloading registers.

- 14x speedup for NTT
- 32x speedup for inverse-NTT

(compared to C reference implementation)

Dilithium:
Keccak, AES256 and Sampling

SHA3/SHAKE

- Supported since z14 (2017).
- High single digit GB/s for long hashing.
- Most hashing generates 840 bytes for SHAKE128.
- The Keccak state is 400 bytes to load and store, so the overhead to start and stop the accelerator is high.

AES256-CTR

- Supported since z196 (2010), further improved in z14 (2017) with hardware IV+counter generation.
- Encrypt/decrypt performance at ~12GB/s for long enough inputs.
- Increased initial hash to output 64 more bytes decreasing the number of calls.

SAMPLING

- Vectorized sampling is similar to the AVX2 optimization, sampling 4 values at the time: Approx. 6.5x speedup compared to the reference implementation.

Dilithium results

Implementation based on PQCRYSTALS code base (round 3 submission)

Overall speedup compared to baseline implementation: factor 6 to 20

Performance of Keccak-based Dilithium comes close to AES-based version (on platforms without Keccak acceleration, the gap is bigger)

Further Keccak speed improvements expected in future generations of Z

	Dilithium		
	Keygen (median us)	Sign (median us)	Verify (median us)
Dilithium2-ref	131.70 us	596.70 us	146.90 us
Dilithium2 (this work)	20.00 us	48.70 us	17.90 us
Dilithium2 Speedup	6.59 x	12.25 x	8.21 x
Dilithium2aes-ref	238.70 us	757.50 us	236.90 us
Dilithium2aes (this work)	16.30 us	42.80 us	14.70 us
Dilithium2aes Speedup	14.64 x	17.70 x	16.12 x
Dilithium3-ref	233.30 us	1005.90 us	234.20 us
Dilithium3 (this work)	46.00 us	80.60 us	27.50 us
Dilithium3 Speedup	5.07 x	12.48 x	8.52 x
Dilithium3aes-ref	454.40 us	1323.50 us	395.00 us
Dilithium3aes (this work)	38.70 us	70.70 us	21.60 us
Dilithium3aes Speedup	11.74 x	18.72 x	18.29 x
Dilithium5-ref	336.30 us	1123.40 us	358.10 us
Dilithium5 (this work)	51.30 us	103.50 us	45.10 us
Dilithium5 Speedup	6.56 x	10.85 x	7.94 x
Dilithium5aes-ref	693.90 us	1569.40 us	666.50 us
Dilithium5aes (this work)	39.30 us	88.10 us	34.10 us
Dilithium5aes Speedup	17.66 x	17.81 x	19.55 x

Performance in microseconds, on IBM z15 LPAR, 5.2 GHz, Linux on Z

Resources

- IBM Z Principles of Operation (ISA reference)
- http://publibfp.dhe.ibm.com/epubs/pdf/a227832c.pdf

- IBM LinuxOne Community Cloud
- https://developer.ibm.com/linuxone/

Thank You!