
Cliff of Rsa
Forest of public key
River of ECC
Sea of classical cryptography
Curve 25519
N
The Realm of post-quantum


Sword of
Academia
Sword of
Industry
pgc
State
of
the
Art
pgc

Fraunhofer AISEC

zeroRISC

CHELPIS

Rivos

Migrating a Silicon Root of Trust to Post-Quantum Crypto

Amin Abdulrahman, Evan Apinis, Andrew 'bunnie' Huang, Matthias J. Kannwischer, Ruben Niederhagen, Felix Oberhansl,
Hoang Nguyen Hien Pham, Jade Philipoom, Dominic Rizzo, Robert Schilling, Peter Schwabe, Tobias Stelzer, Augustine Tang, Andreas Zankl

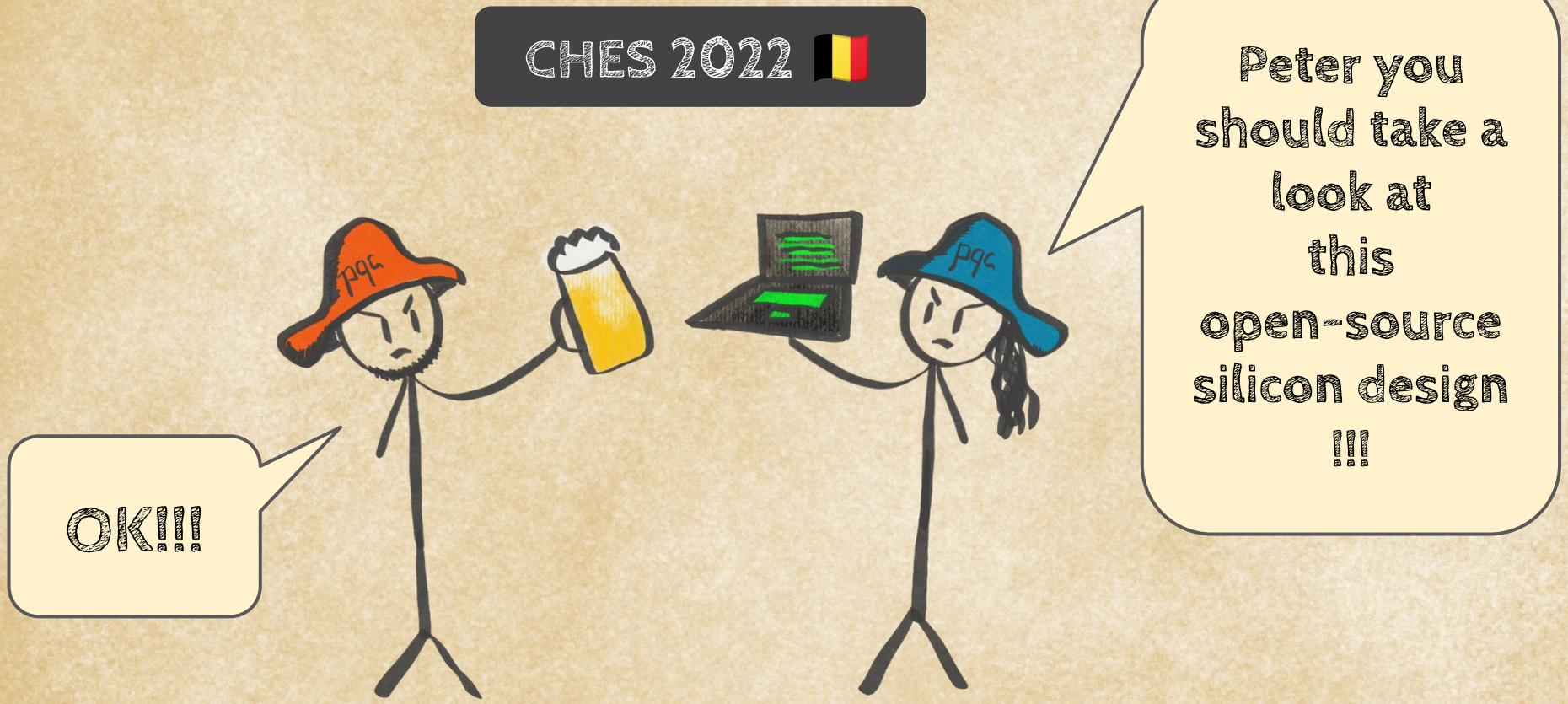
1. SPHINCS+ Secure Boot

1. SPHINCS+ Secure Boot
2. HW/SW Codesign for ML-KEM & ML-DSA

1. SPHINCS+ Secure Boot
2. HW/SW Codesign for ML-KEM & ML-DSA
3. Ongoing Work & Lessons Learned

SPHINCS+

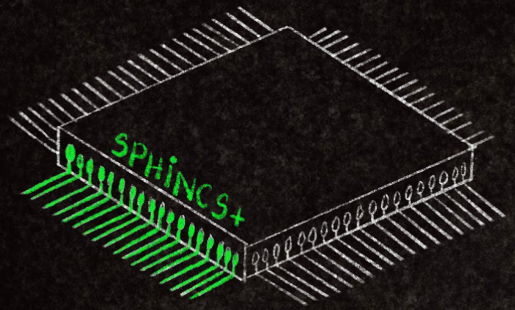
(aka SLH-DSA)


CHES 2022 🇧🇪
pqc
pqc
Peter you should take a look at this open-source silicon design !!!
OK!!!

29 hours later...

The first signature is VERIFIED

6 months later... SPHINCS+ secure boot!

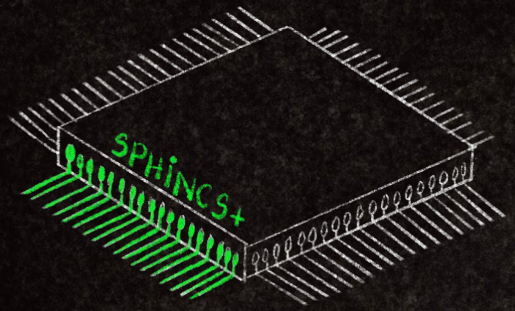


3x faster than unmodified reference implementation

- shake-128s-simple: **< 13 ms @100MHz**
- sha3-128s-simple: **~ 9.3 ms @100MHz**

← Blog post: Landing SPHINCS+ on OpenTitan

6 months later... SPHINCS+ secure boot!



3x faster than unmodified reference implementation

- shake-128s-simple: **< 13 ms @100MHz**
- sha3-128s-simple: **~ 9.3 ms @100MHz**

6x slower than RSA-3072

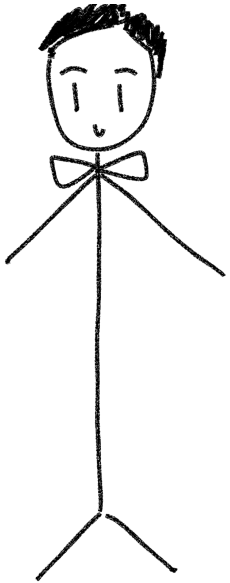
3x slower than ECDSA-P256

- one-time programmable (OTP) memory configuration to enable

← Blog post: Landing SPHINCS+ on OpenTitan

HW/SW Codesign for ML-KEM & ML-DSA

Amin's Master Thesis: Dilithium on OpenTitan (2023)

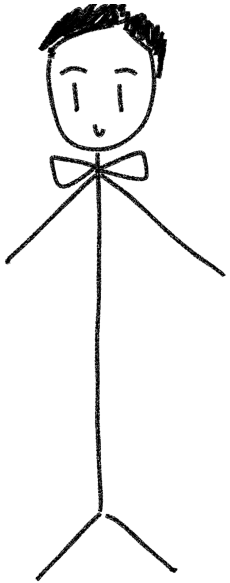


8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

Thesis

Amin's Master Thesis: Dilithium on OpenTitan (2023)



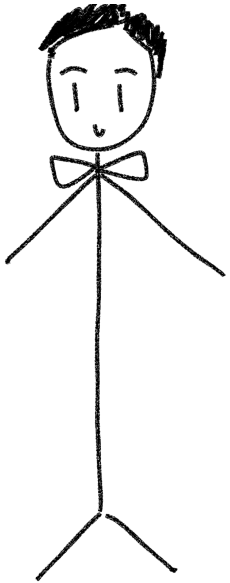
8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

- bn.addv(m): vector (modular) addition

Thesis

Amin's Master Thesis: Dilithium on OpenTitan (2023)



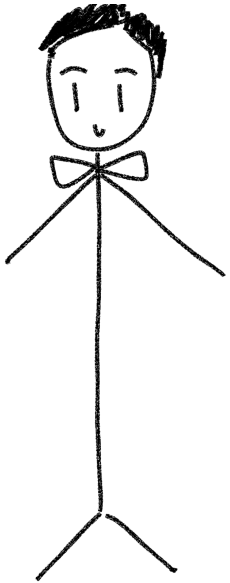
8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

- bn.addv(m): vector (modular) addition
- bn.subv(m): vector (modular) subtraction

Thesis

Amin's Master Thesis: Dilithium on OpenTitan (2023)



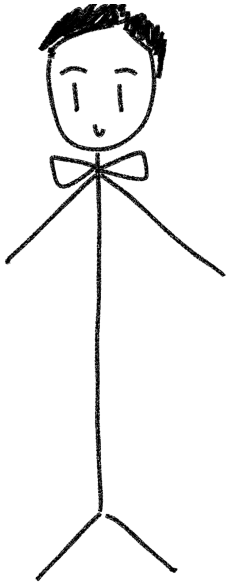
8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

- bn.addv(m): vector (modular) addition
- bn.subv(m): vector (modular) subtraction
- bn.mulv(m): vector (modular) multiplication
  -> hardware multiplier

Thesis

Amin's Master Thesis: Dilithium on OpenTitan (2023)



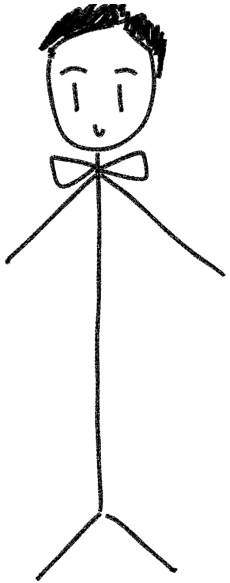
8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

- bn.addv(m): vector (modular) addition
- bn.subv(m): vector (modular) subtraction
- bn.mulv(m): vector (modular) multiplication
  -> hardware multiplier
- bn.shv: vector shift
- bn.trans: vector transpose

Thesis

Amin's Master Thesis: Dilithium on OpenTitan (2023)



8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

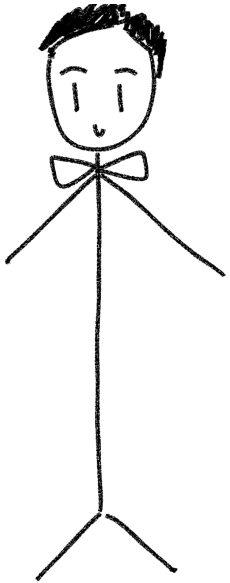
Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

- bn.addv(m): vector (modular) addition
- bn.subv(m): vector (modular) subtraction
- bn.mulv(m): vector (modular) multiplication
  -> hardware multiplier
- bn.shv: vector shift
- bn.trans: vector transpose

Idea for SHA3/SHAKE connection to OTBN

Thesis

Amin's Master Thesis: Dilithium on OpenTitan (2023)



8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

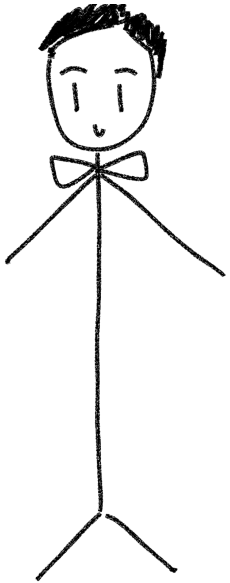
- bn.addv(m): vector (modular) addition
- bn.subv(m): vector (modular) subtraction
- bn.mulv(m): vector (modular) multiplication
  -> hardware multiplier
- bn.shv: vector shift
- bn.trans: vector transpose

Idea for SHA3/SHAKE connection to OTBN

Software implementation of Round-3 Dilithium

Thesis

Amin's Master Thesis: Dilithium on OpenTitan (2023)



8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Proposed vector instructions for OpenTitan Big-Number Accelerator (OTBN):

- bn.addv(m): vector (modular) addition
- bn.subv(m): vector (modular) subtraction
- bn.mulv(m): vector (modular) multiplication
  -> hardware multiplier
- bn.shv: vector shift
- bn.trans: vector transpose

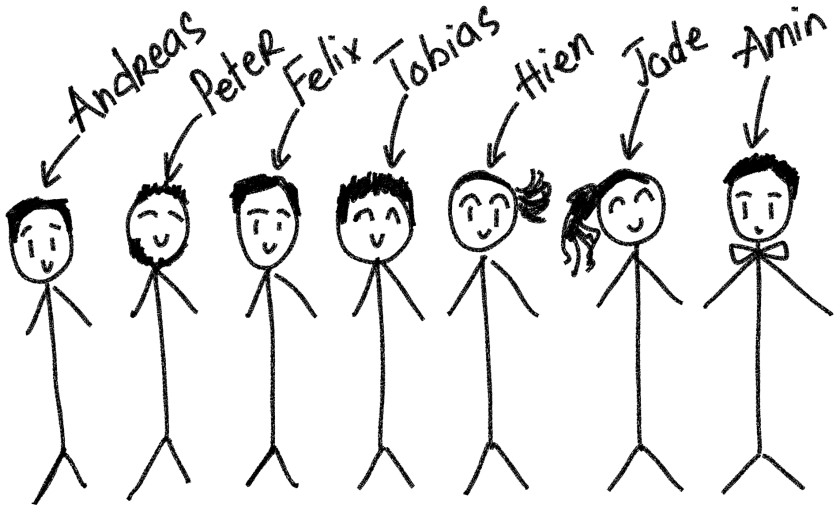
Idea for SHA3/SHAKE connection to OTBN

Software implementation of Round-3 Dilithium

Increased IMEM from 8 to 32KB and DMEM from 4 to 128KB

Thesis

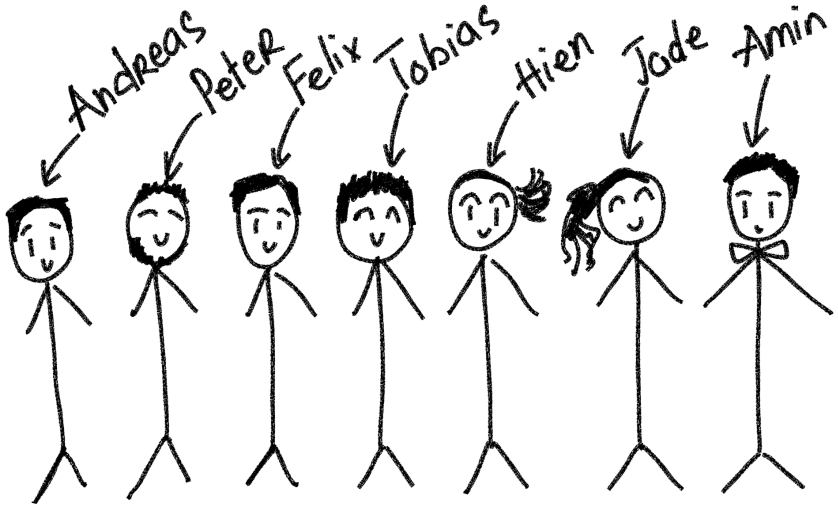

Let's collaborate again



Vector instructions for OTBN:

- bn.{add,sub,mul}v(m)
- bn.shv

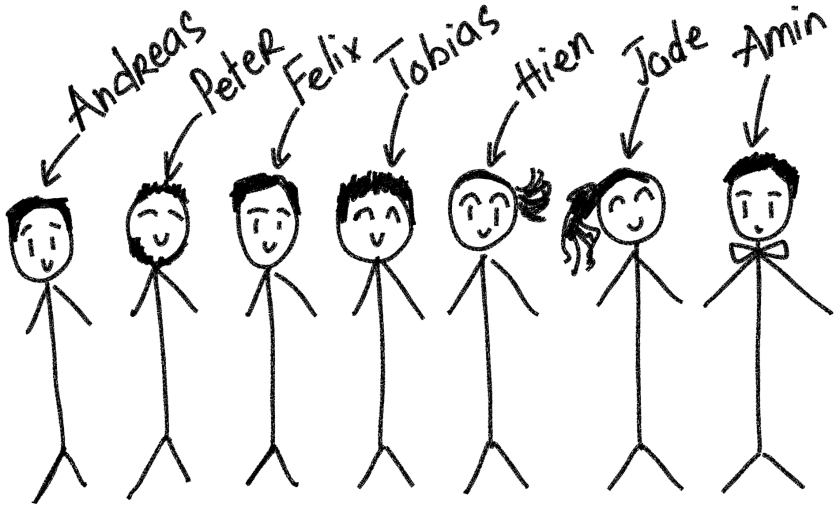
Paper Code



Vector instructions for OTBN:

- bn.{add,sub,mul}v(m)
- bn.shv
- ~~bn.trans~~ bn.trn: vector transpose

Paper Code

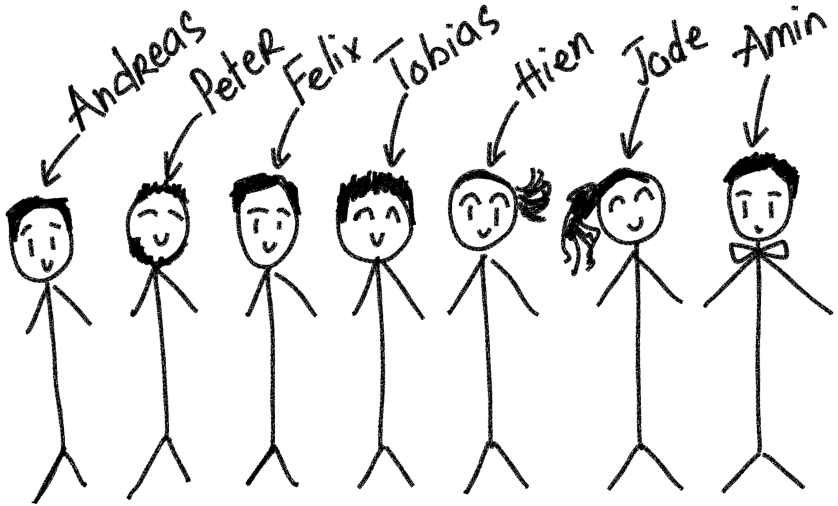


Vector instructions for OTBN:

- bn.{add,sub,mul}v(m)
- bn.shv
- ~~bn.trans~~ bn.trn: vector transpose

Software implementation of ~~Round 3 Dilithium~~ ML-DSA and ML-KEM

Paper Code



Vector instructions for OTBN:

- bn.{add,sub,mul}v(m)
- bn.shv
- ~~bn.trans~~ bn.trn: vector transpose

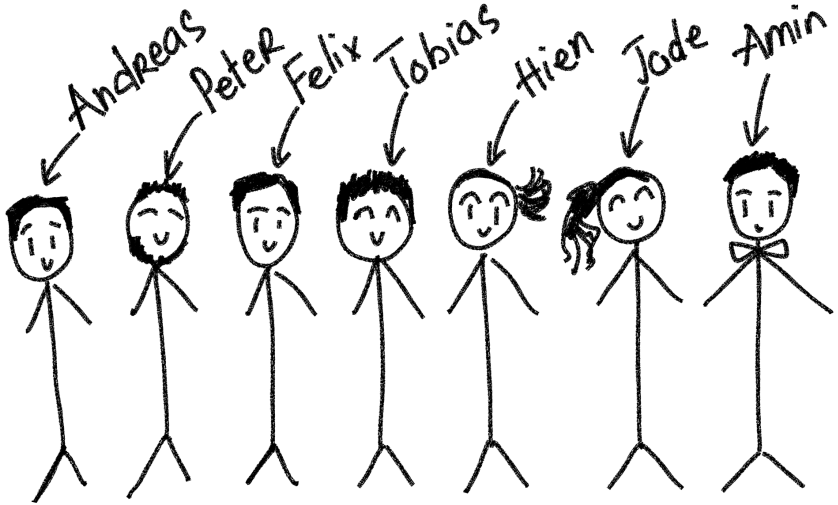
Software implementation of ~~Round 3 Dilithium~~ ML-DSA and ML-KEM

Hardware implementations of the proposed instructions

New SHA3/SHAKE hardware connection

Paper Code

Towards ML-KEM and ML-DSA on OpenTitan (IEEE S&P 2025)



Vector instructions for OTBN:

- bn.{add,sub,mul}v(m)
- bn.shv
- ~~bn.trans~~ bn.trn: vector transpose

Software implementation of ~~Round 3 Dilithium~~ ML-DSA and ML-KEM

Hardware implementations of the proposed instructions

New SHA3/SHAKE hardware connection

Paper Code

Result at target frequency

← Blog post:
Accelerating Post-Quantum Cryptography on OpenTitan-based Designs: Flexible Hardware for a Secure Future

Result at target frequency

- **6-9x times faster** than ML-KEM and ML-DSA on OTBN without vector instructions

← Blog post:
Accelerating Post-Quantum Cryptography on OpenTitan-based Designs: Flexible Hardware for a Secure Future

Result at target frequency

- **6-9x times faster** than ML-KEM and ML-DSA on OTBN without vector instructions
    * ML-KEM-768 decapsulation: **0.8 ms** at 100MHz

← Blog post:
Accelerating Post-Quantum Cryptography on OpenTitan-based Designs: Flexible Hardware for a Secure Future

Result at target frequency

- **6-9x times faster** than ML-KEM and ML-DSA on OTBN without vector instructions
    * ML-KEM-768 decapsulation: **0.8 ms** at 100MHz
    * ML-DSA-65 signing (median): **7.0 ms** at 100 MHz

← Blog post:
Accelerating Post-Quantum Cryptography on OpenTitan-based Designs: Flexible Hardware for a Secure Future

Result at target frequency

- **6-9x times faster** than ML-KEM and ML-DSA on OTBN without vector instructions
    * ML-KEM-768 decapsulation: **0.8 ms** at 100MHz
    * ML-DSA-65 signing (median): **7.0 ms** at 100 MHz

=> Faster than non-PQC alternatives for most operations!

← Blog post:
Accelerating Post-Quantum Cryptography on OpenTitan-based Designs: Flexible Hardware for a Secure Future

Result at target frequency

- **6-9x times faster** than ML-KEM and ML-DSA on OTBN without vector instructions
    * ML-KEM-768 decapsulation: **0.8 ms** at 100MHz
    * ML-DSA-65 signing (median): **7.0 ms** at 100 MHz

=> Faster than non-PQC alternatives for most operations!

- Cell-count increase of **less than 17% for OTBN** and **less than 3% for OpenTitan's Top Earlgrey (without memory)**

← Blog post:
Accelerating Post-Quantum Cryptography on OpenTitan-based Designs: Flexible Hardware for a Secure Future

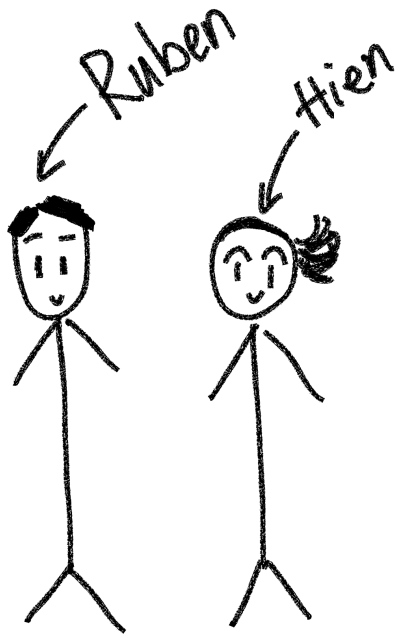
But...We can still do better

But...We can still do better

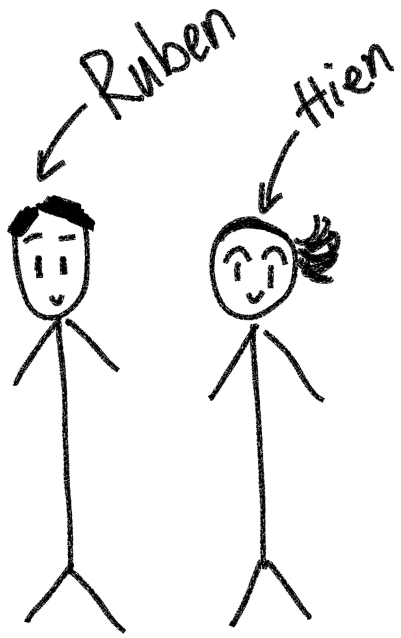
- bn.mulvm takes 12 cycles and bn.mulv takes 4 cycles

But...We can still do better

- bn.mulvm takes 12 cycles and bn.mulv takes 4 cycles
- Sequential vector adder design leads to reduced maximum frequency



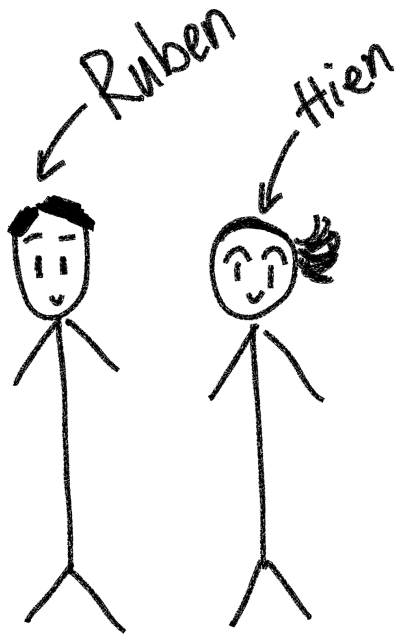
Paper Code



Shift modular multiplication back to SW:

- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}

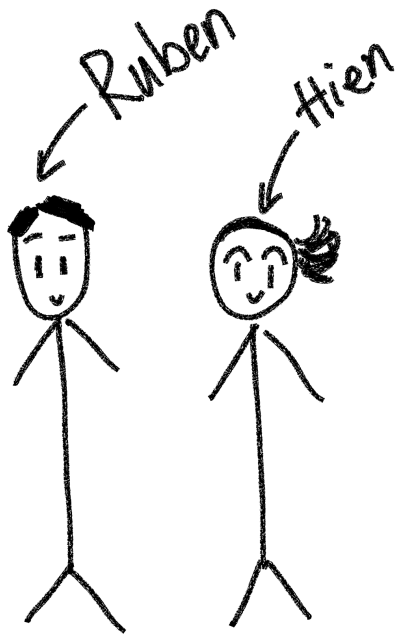
Paper Code



Shift modular multiplication back to SW:

- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}
- For a Montgomery modular multiplication:
  - 32-bit: ~~12 cycles~~ 7 cycles

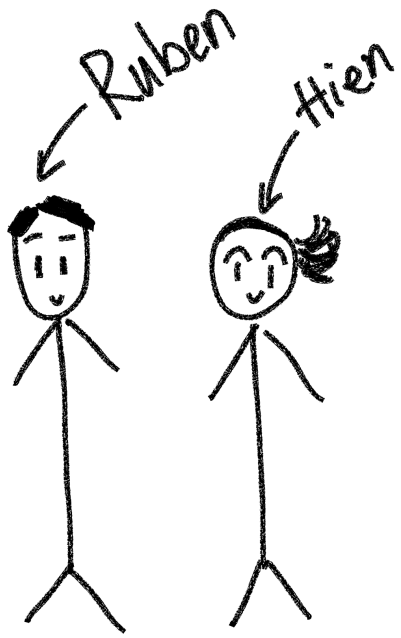
Paper Code



Shift modular multiplication back to SW:

- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}
- For a Montgomery modular multiplication:
  - 32-bit: ~~12 cycles~~ 7 cycles
  - 16-bit: ~~12 cycles~~ 5 cycles

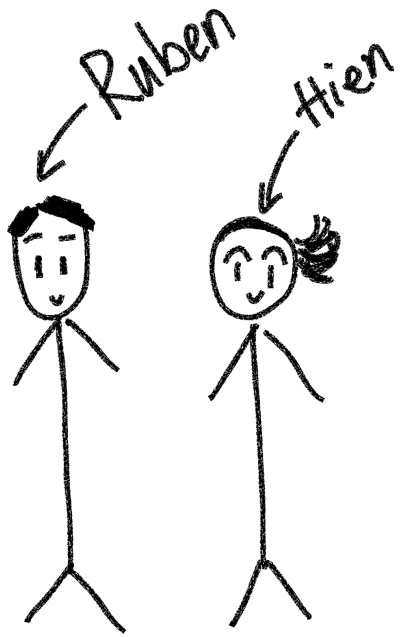
Paper Code



Shift modular multiplication back to SW:

- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}
- For a Montgomery modular multiplication:
  - * 32-bit: ~~12 cycles~~ 7 cycles
  - * 16-bit: ~~12 cycles~~ 5 cycles
- For non-modular multiplication:
  - * 32-bit: ~~4 cycles~~ 2 cycles

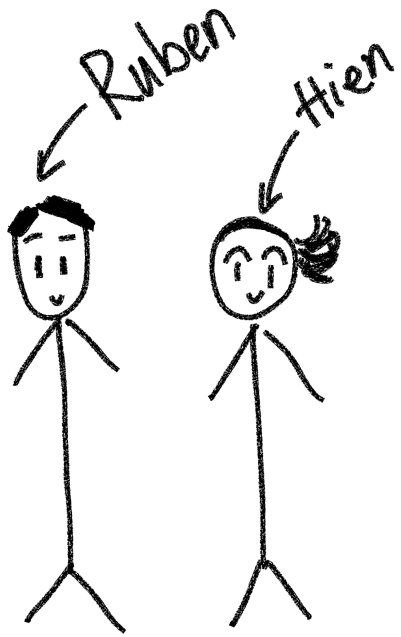
Paper Code



Shift modular multiplication back to SW:

- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}
- For a Montgomery modular multiplication:
  - 32-bit: ~~12 cycles~~ 7 cycles
  - 16-bit: ~~12 cycles~~ 5 cycles
- For non-modular multiplication:
  - 32-bit: ~~4 cycles~~ 2 cycles
  - 16-bit: ~~4 cycles~~ 1 cycle

Paper Code

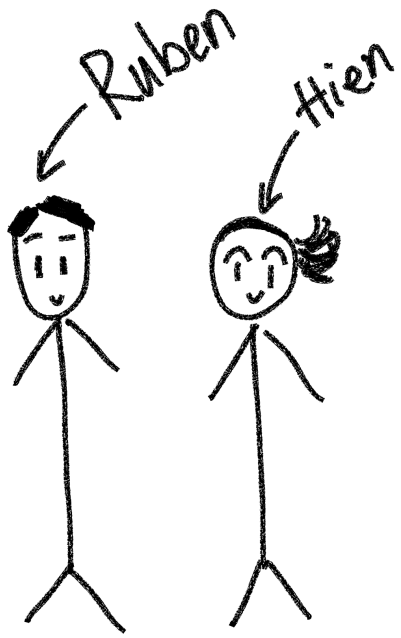


Shift modular multiplication back to SW:

- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}
- For a Montgomery modular multiplication:
  - * 32-bit: ~~12 cycles~~ 7 cycles
  - * 16-bit: ~~12 cycles~~ 5 cycles
- For non-modular multiplication:
  - * 32-bit: ~~4 cycles~~ 2 cycles
  - * 16-bit: ~~4 cycles~~ 1 cycle

=> ↓latency, ↑code size

Paper Code



Shift modular multiplication back to SW:

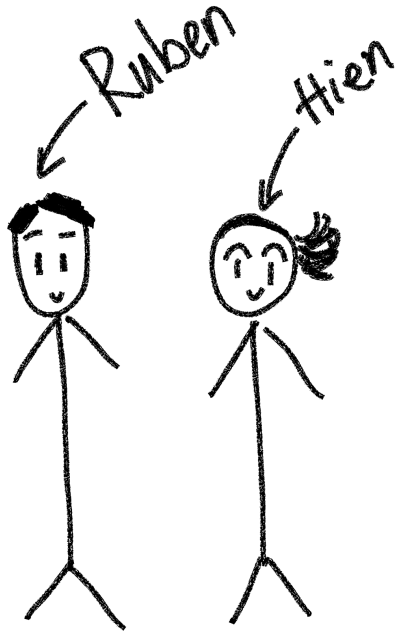
- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}
- For a Montgomery modular multiplication:
  - * 32-bit: ~~12 cycles~~ 7 cycles
  - * 16-bit: ~~12 cycles~~ 5 cycles
- For non-modular multiplication:
  - * 32-bit: ~~4 cycles~~ 2 cycles
  - * 16-bit: ~~4 cycles~~ 1 cycle

=> ↓latency, ↑code size

Improved vector adder designs (tailored for target platform ASIC/FPGA)

Paper Code

Improving ML-KEM and ML-DSA on OpenTitan (CHES 2026)



Shift modular multiplication back to SW:

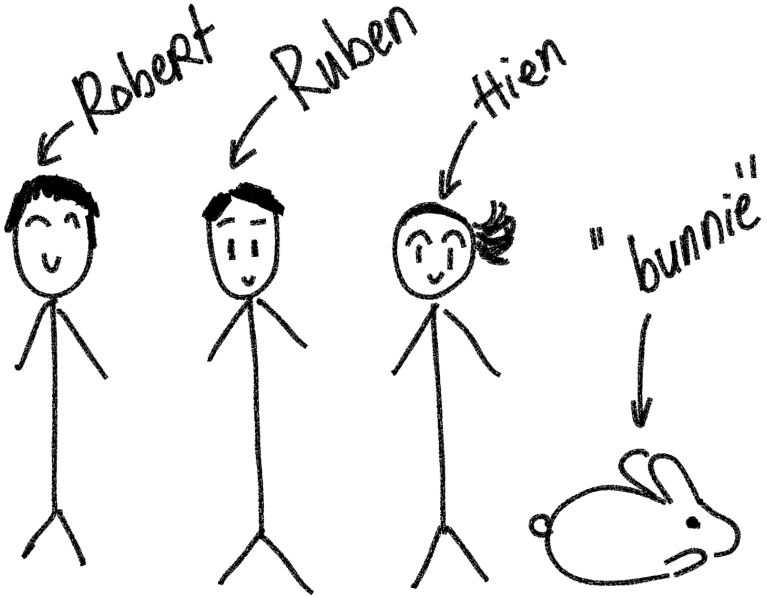
- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd} -> V{1,2,3}
- For a Montgomery modular multiplication:
  - 32-bit: ~~12 cycles~~ 7 cycles
  - 16-bit: ~~12 cycles~~ 5 cycles
- For non-modular multiplication:
  - 32-bit: ~~4 cycles~~ 2 cycles
  - 16-bit: ~~4 cycles~~ 1 cycle

=> ↓latency, ↑code size

Improved vector adder designs (tailored for target platform ASIC/FPGA)

Paper Code

Improving ML-KEM and ML-DSA on OpenTitan (CHES 2026)



Paper Code

Shift modular multiplication back to SW:

- ~~bn.mulv(m)~~ bn.mulv{.lo,.hi}{.even,.odd}
- ↓latency, ↑code size

Improved vector adder designs (tailored for target platform ASIC/FPGA)

ASIC synthesis on other technology nodes and confirmed that the results are consistent with what we had using Cadence and OpenROAD

Results

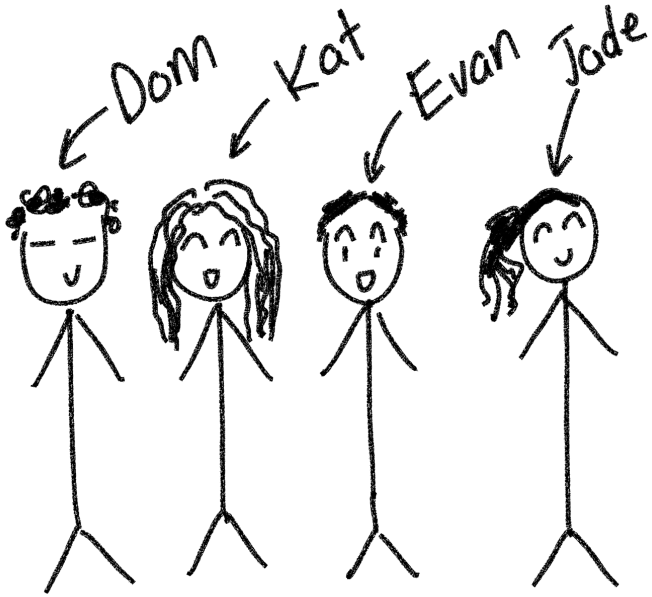
- Up to 17% speedup in cycle count and 16% increase in code size for ML-KEM and ML-DSA compared to previous paper on OTBNV2

Results

- Up to 17% speedup in cycle count and 16% increase in code size for ML-KEM and ML-DSA compared to previous paper on OTBNV2
- Similar area while improving maximum frequency of the whole OTBN

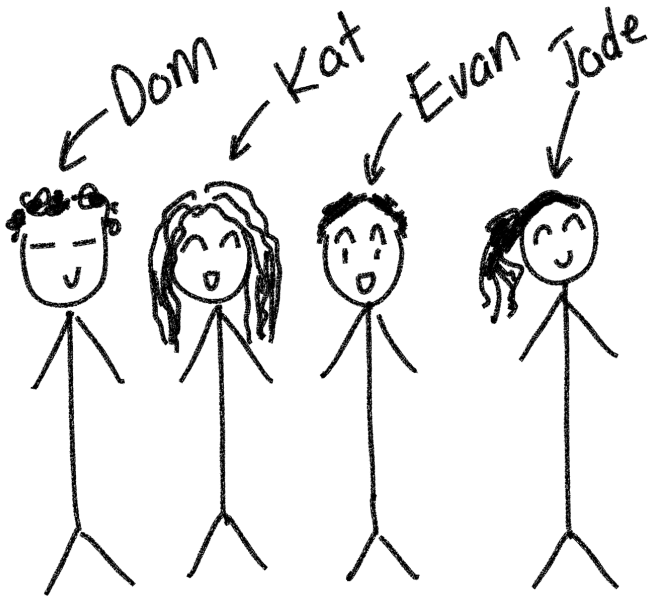
Ahem...but we can still do better!!!

Here comes the ZeroRISC refinement squad...



← Blog post: From Artifact to Production: Integrating and Refining Lattice Cryptography Acceleration

Here comes the ZeroRISC refinement squad...

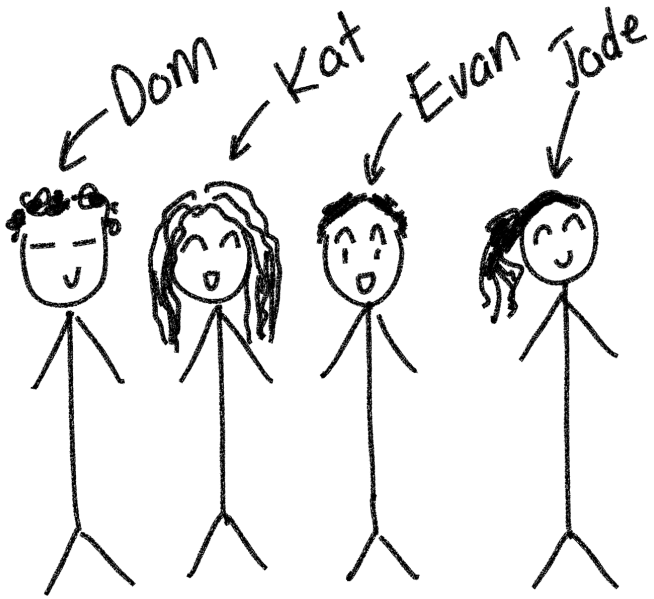


Stack optimized ML-DSA

- 121kB -> 11kB (91% reduction) for ML-DSA-87 signing
- Tradeoff: 80%-140% slowdown for sign (negligible for other ops)

← Blog post: From Artifact to Production: Integrating and Refining Lattice Cryptography Acceleration

Here comes the ZeroRISC refinement squad...



Stack optimized ML-DSA

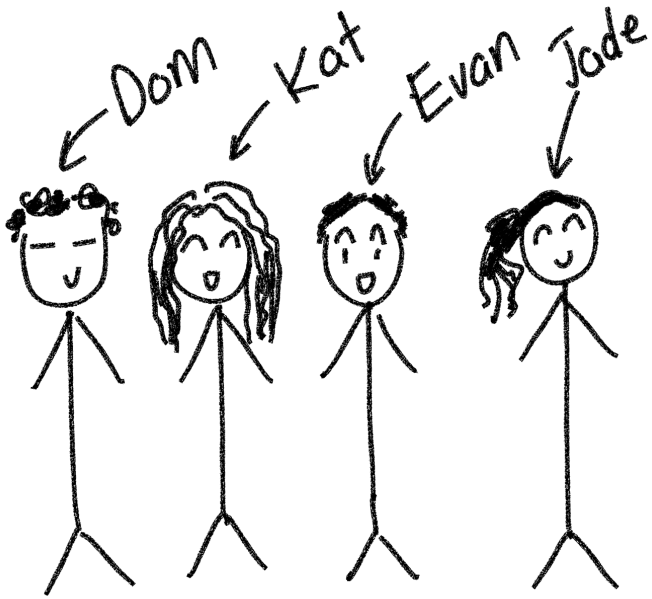
- 121kB -> 11kB (91% reduction) for ML-DSA-87 signing
- Tradeoff: 80%-140% slowdown for sign (negligible for other ops)

Faster rejection sampling for

- ML-KEM: ~13% speedup (all ops)
- ML-DSA: ~52% speedup (all ops)

← Blog post: From Artifact to Production: Integrating and Refining Lattice Cryptography Acceleration

Here comes the ZeroRISC refinement squad...



Stack optimized ML-DSA
- 121kB -> 11kB (91% reduction) for ML-DSA-87 signing
- Tradeoff: 80%-140% slowdown for sign (negligible for other ops)

Faster rejection sampling for
- ML-KEM: ~13% speedup (all ops)
- ML-DSA: ~52% speedup (all ops)

Redesigned SHA3/SHAKE interface
- ~6% speedup (all ops)

Integration with Design Verification tools

← Blog post: From Artifact to Production: Integrating and Refining Lattice Cryptography Acceleration

Beyond bignums...

OTBN + ML-DSA + ML-KEM = ACC

Asymmetric
Cryptography
Coprocessor

Beyond bignums...

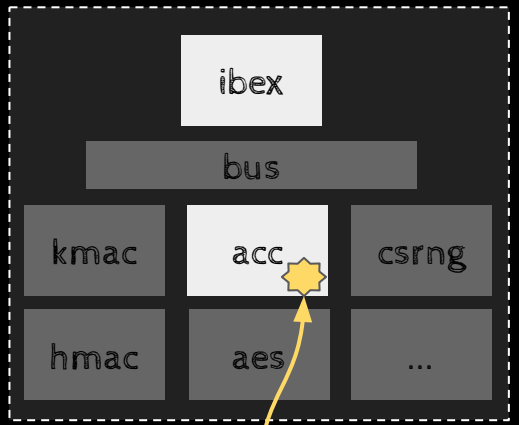
OTBN + ML-DSA + ML-KEM = ACC

Asymmetric
Cryptography
Coprocessor

ML-KEM-768 decapsulation: 0.7ms at 100MHz
ML-DSA-65 signing (median): 6.5ms at 100MHz
Memory: 32KB IMEM, 32KB DMEM

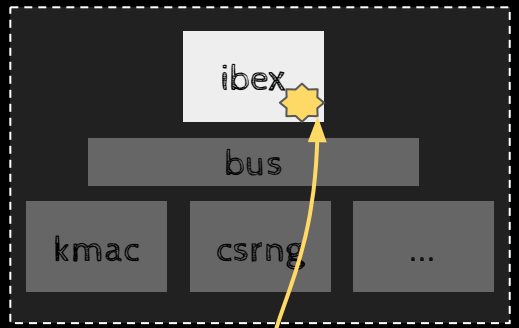
Multiple integration options for flexibility

Ibex + ACC

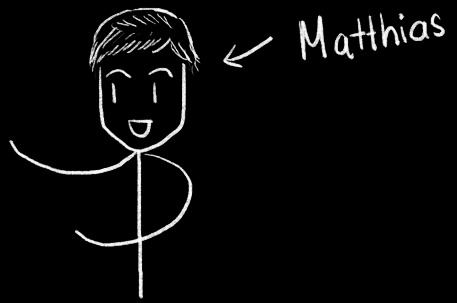


fast ML-KEM and ML-DSA on ACC

Ibex only

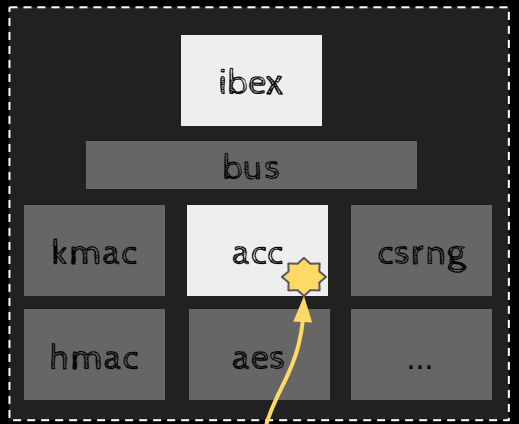


mlkem-native and mldsa-native on Ibex



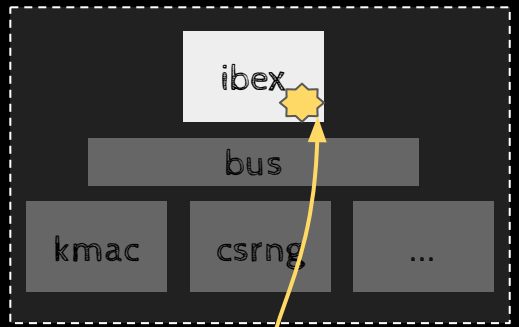
Multiple integration options for flexibility

Ibex + ACC



fast ML-KEM and ML-DSA on ACC

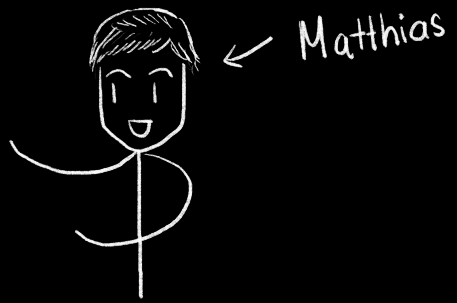
Ibex only



mlkem-native and mldsa-native on Ibex

Lower area, but slower asymmetric cryptography:

- ~10x for ML-KEM, ML-DSA
- ~20x for RSA



Ongoing Work, Lessons Learned


ACC

Preparing for
industry and
research tapeouts

Preparing for
industry and
research tapeouts

ISE for
UOV/Mayo,
FN-DSA

Preparing for industry and research tapeouts

Iasmin support for ACC

ISE for UOV/Mayo, FN-DSA

Preparing for industry and research tapeouts

Iasmin support for ACC

SLOTHY for automatic leakage mitigation

ISE for UOV/Mayo, FN-DSA

Preparing for industry and research tapeouts

Iasmin support for ACC

SLOTHY for automatic leakage mitigation

ISE for UOV/Mayo, FN-DSA

Side-channel attacks & evaluation

Preparing for industry and research tapeouts

Iasmin support for ACC

SLOTHY for automatic leakage mitigation

ISE for UOV/Mayo, FN-DSA

Side-channel attacks & evaluation

Masking for ML-KEM & ML-DSA (Hien's internship)

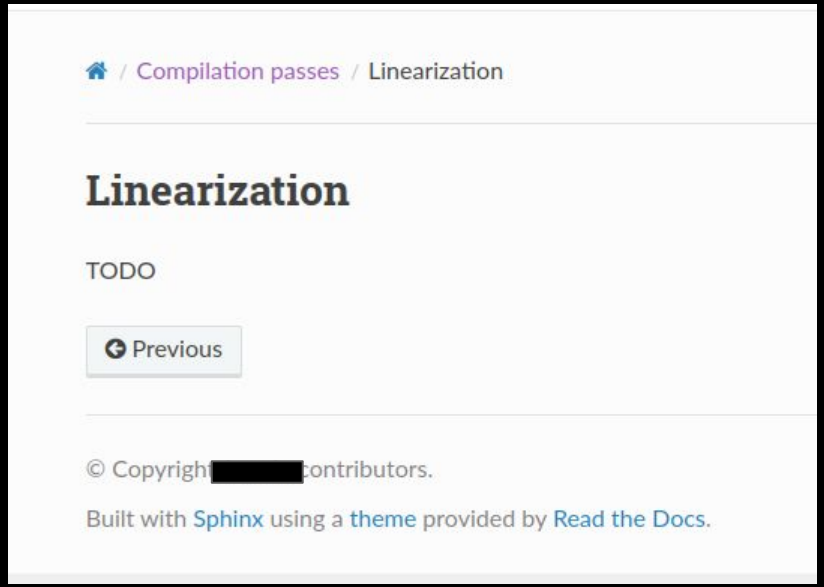
Lesson 1

Open-source removes friction

Lesson 2

Documentation saves ...lives

Official Documentation



“Official” Documentation

Hi Santiago,

Hi Santiago,

Hi Santiago,

Hellu Santiago,

Hi Santiago (again),

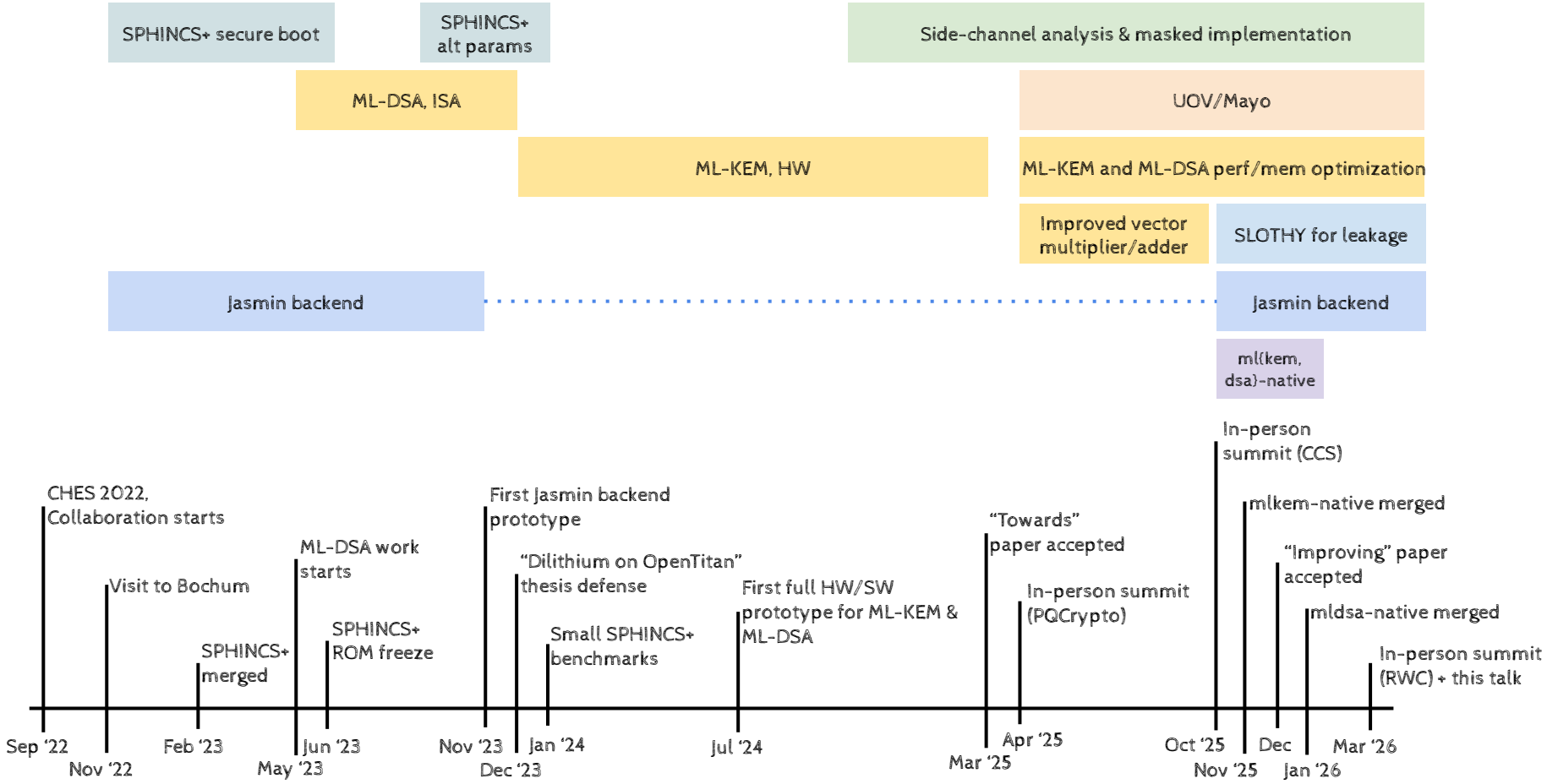
AskSantiago Premium™ 🤓

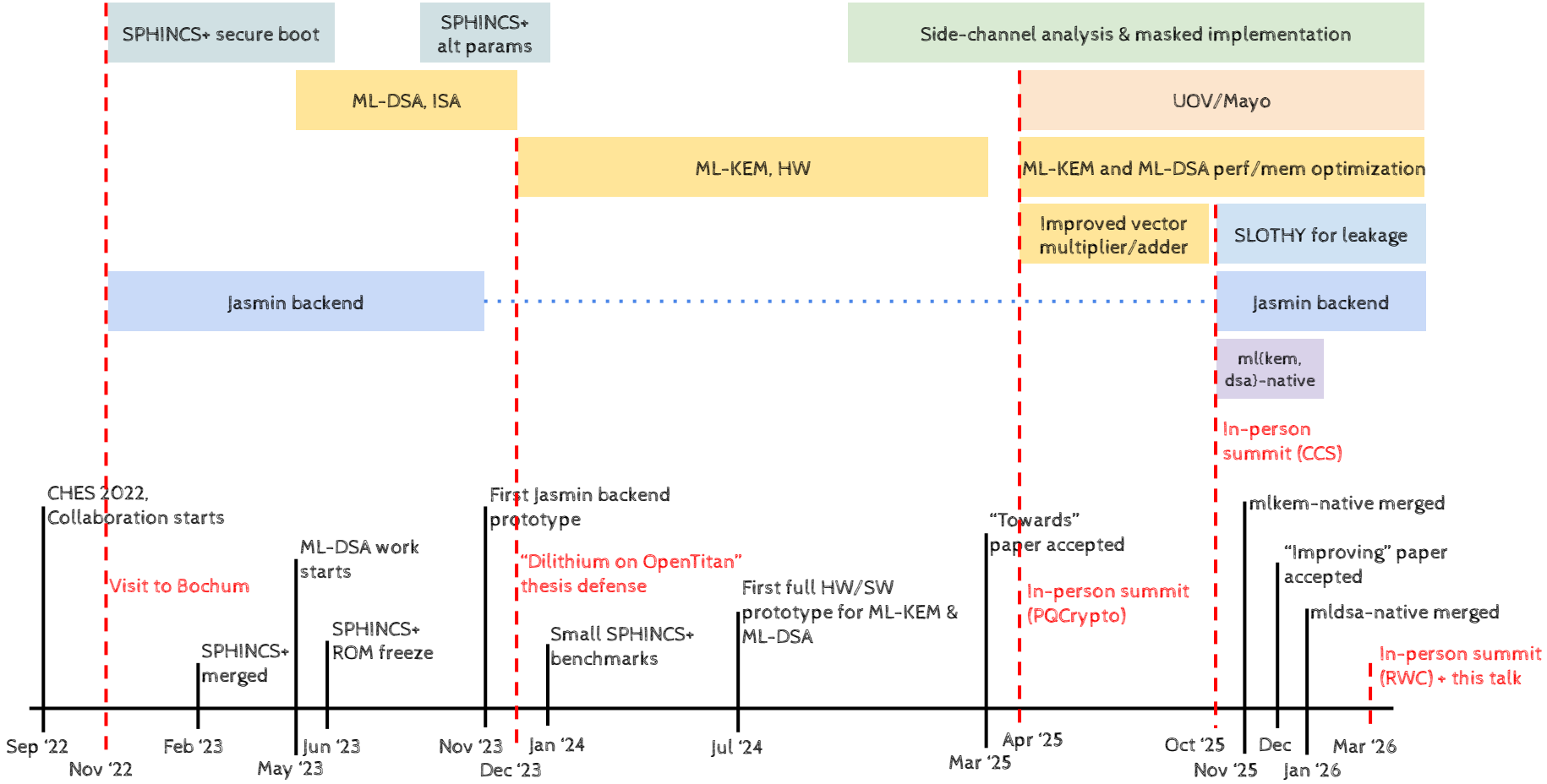
Lesson 3

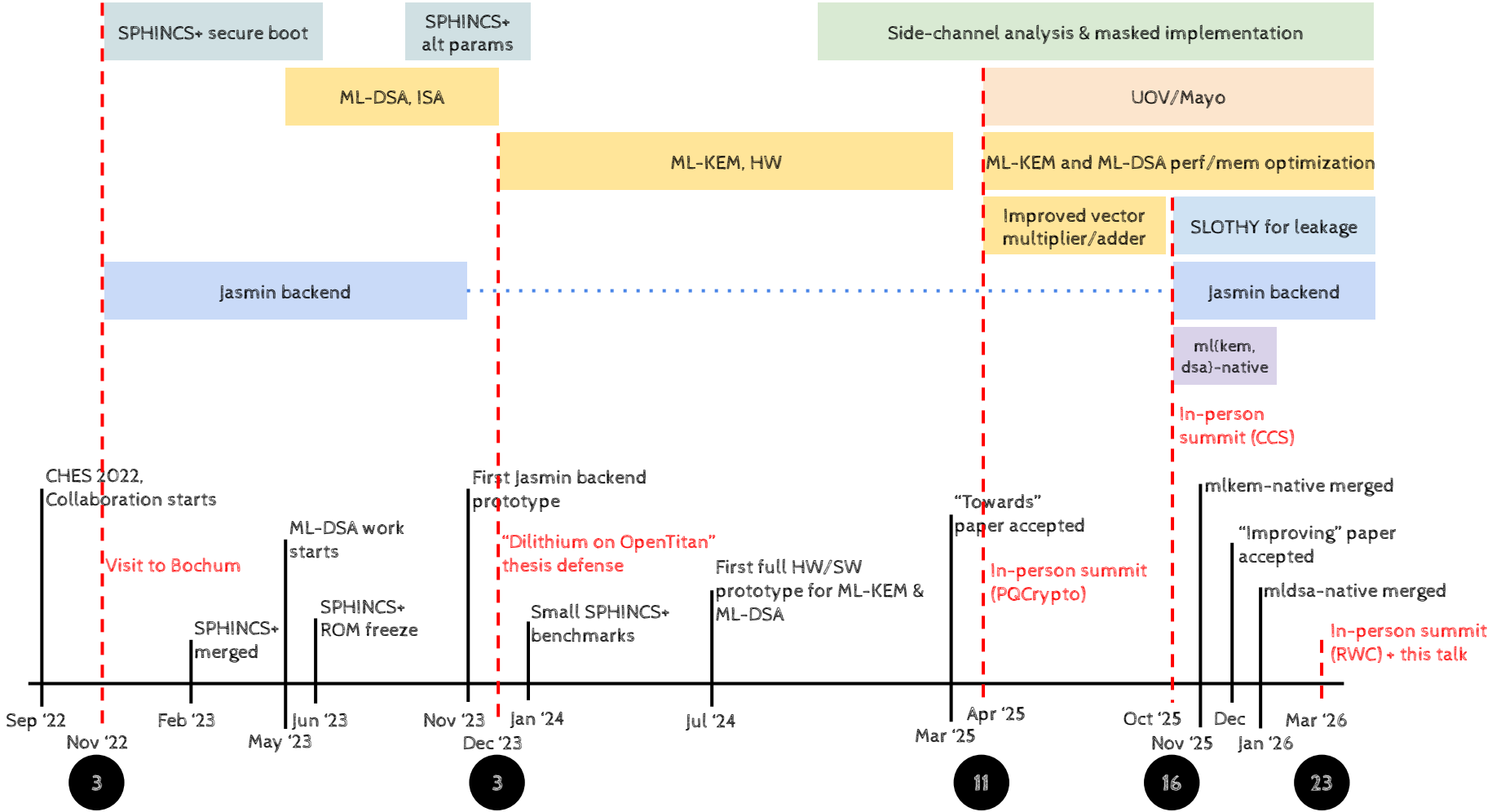
Research sparks, industry scales

Lesson 4

Meeting in person MATTERS!!!

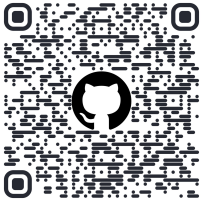

SPHINCS+ secure boot
SPHINCS+ alt params
Side-channel analysis & masked implementation
ML-DSA, ISA
UOV/Mayo
ML-KEM, HW
ML-KEM and ML-DSA perf/mem optimization
Improved vector multiplier/adder
SLOTHY for leakage
Jasmin backend
Jasmin backend
ml(kem, dsa)-native
CHES 2022, Collaboration starts
Visit to Bochum
SPHINCS+ merged
ML-DSA work starts
SPHINCS+ ROM freeze
First Jasmin backend prototype
"Dilithium on OpenTitan" thesis defense
Small SPHINCS+ benchmarks
First full HW/SW prototype for ML-KEM & ML-DSA
"Towards" paper accepted
In-person summit (PQCrypto)
In-person summit (CCS)
mlkem-native merged
"Improving" paper accepted
mldsa-native merged
In-person summit (RWC) + this talk
Sep '22
Nov '22
Feb '23
May '23
Jun '23
Nov '23
Dec '23
Jan '24
Jul '24
Mar '25
Apr '25
Oct '25
Nov '25
Dec
Jan '26
Mar '26


SPHINCS+ secure boot
SPHINCS+ alt params
Side-channel analysis & masked implementation
ML-DSA, ISA
UOV/Mayo
ML-KEM, HW
ML-KEM and ML-DSA perf/mem optimization
Improved vector multiplier/adder
SLOTHY for leakage
Jasmin backend
Jasmin backend
ml(kem, dsa)-native
In-person summit (CCS)
CHES 2022, Collaboration starts
Visit to Bochum
SPHINCS+ merged
ML-DSA work starts
SPHINCS+ ROM freeze
First Jasmin backend prototype
"Dilithium on OpenTitan" thesis defense
Small SPHINCS+ benchmarks
First full HW/SW prototype for ML-KEM & ML-DSA
"Towards" paper accepted
In-person summit (PQCrypto)
mlkem-native merged
"Improving" paper accepted
mldsa-native merged
In-person summit (RWC) + this talk
Sep '22
Nov '22
Feb '23
May '23
Jun '23
Nov '23
Dec '23
Jan '24
Jul '24
Mar '25
Apr '25
Oct '25
Nov '25
Dec
Jan '26
Mar '26


SPHINCS+ secure boot
SPHINCS+ alt params
Side-channel analysis & masked implementation
ML-DSA, ISA
UOV/Mayo
ML-KEM, HW
ML-KEM and ML-DSA perf/mem optimization
Improved vector multiplier/adder
SLOTHY for leakage
Jasmin backend
Jasmin backend
ml(kem, dsa)-native
In-person summit (CCS)
CHES 2022, Collaboration starts
Visit to Bochum
SPHINCS+ merged
ML-DSA work starts
SPHINCS+ ROM freeze
First Jasmin backend prototype
"Dilithium on OpenTitan" thesis defense
Small SPHINCS+ benchmarks
First full HW/SW prototype for ML-KEM & ML-DSA
"Towards" paper accepted
In-person summit (PQCrypto)
mlkem-native merged
"Improving" paper accepted
mldsa-native merged
In-person summit (RWC) + this talk
Sep '22
Nov '22
Feb '23
May '23
Jun '23
Nov '23
Dec '23
Jan '24
Jul '24
Mar '25
Apr '25
Oct '25
Nov '25
Dec
Jan '26
Mar '26
3
3
11
16
23


饒河街觀光夜市
Raohe St. Night Market
Welcome Open-source Silicon Collaborators
哈囉海鮮鍋燒
粥。意麵。雞絲。烏龍
申龍帆布
2596-2406
"bunnie"
Hanno
Lejla
Santiago
Benjamin
Inez
Kat
Mathias
Jade
Ruben
Peter
Evan
Francisca
Dom
Pierre-Yves
Amin
Hiier
Julius
Augustine
Robert
Richard
Azade
Parisa

Thank you for your attention

Towards ML-KEM & ML-DSA on OpenTitan (IEEE S&P 2025)

ZeroRISC's OpenTitan fork


Improving ML-KEM & ML-DSA on OpenTitan (CHES 2026)

ZeroRISC's Blog

Max Planck Institute for Security and Privacy (MPI-SP) Blog

Presentation art: Hoang Nguyen Hien Pham

Contact us:

- Hien: nguyenhien.phamhoang@gmail.com
- Jade: jade@semiprecious.net
- MPI-SP: office@csp.mpg.de
- ZeroRISC: info@zerorisc.com