May, 2019

A Brief Introduction to Lattice-Based Cryptography in Hardware

James Howe
PQShield, UK and University of Bristol, UK

NIST PQC Hardware Day, 2nd May 2019

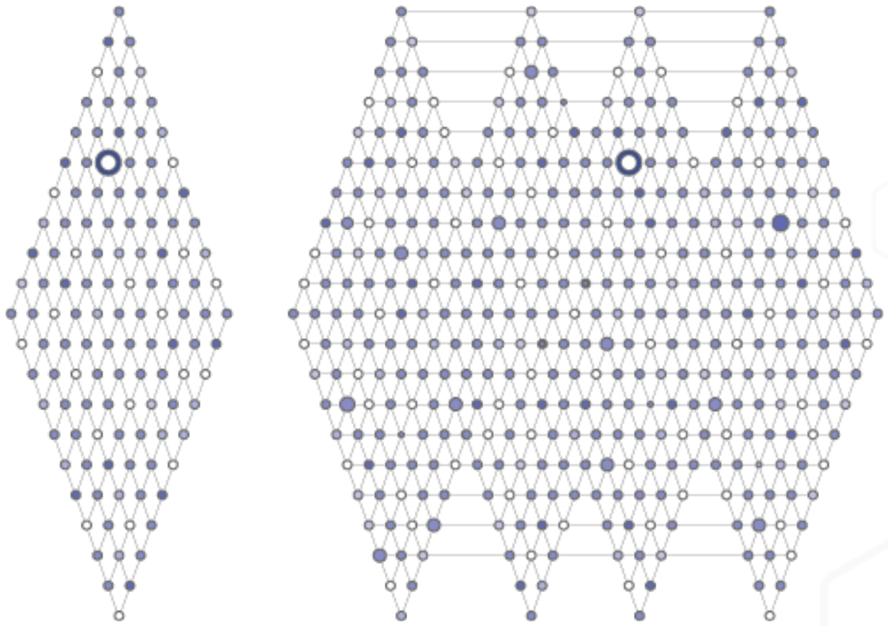
Content

- Why is lattice-based cryptography hard?
    - Operations / components / sizes required.
- What's changed within candidates?
    - M-LWE and LWR.
- Designs pre-/post-standardisation announcement.
    - Specifically, some lattice-based signature and KEM hardware designs.
- Other / miscellaneous.

Please interrupt me with questions, comments, or (more likely) errors.

Why Lattices?

- Mathematics easier to understand (vs e.g. ECC).
- Operations require simple multiplication, addition, modular reduction.
- Simple parameter selection / scalable to fit security needs.
- Average-case to worst-case hardness.
- Offers KEM, signatures, FHE, IBE, etc.
- Highest candidate numbers submitted to NIST.
- No major security issues in 30+ years.
- Already used by Google, strongSwan VPN, etc.
- Efficient KeyGen, Encrypt/Sign, Decrypt/Verify.
- Relatively small keys, ciphertexts, and signatures.



Learning With Errors

- There is a secret vector $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$.
- An oracle (who knows $\boldsymbol{s}$) generates a uniform matrix $\boldsymbol{A}$ and noise vector **e** distributed normally with standard deviation αq.
- The oracle outputs: $\boldsymbol{A}$ and $\boldsymbol{b} = \boldsymbol{A} \times \boldsymbol{s} + \boldsymbol{e} \bmod q$.
- The distribution of $\boldsymbol{A}$ is uniformly random, $\boldsymbol{b}$ is pseudo-random.
- Can you find $\boldsymbol{s}$, given access to $(\boldsymbol{A}, \boldsymbol{b})$?
- Can you distinguish $(\boldsymbol{A}, \boldsymbol{b})$ from a uniformly random $(\boldsymbol{A}, \boldsymbol{b}')$?

Ideal and Module Lattices

- Standard lattices deal with matrices / vectors.
- Adding additional structure, one can deal with *ideal* or *module* lattices.
- Thus, (cyclic) matrices can be replaced with polynomials.
- Efficiencies are then gained using polynomial multiplication (e.g. NTT) over the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ for $q = 1 \bmod 2n$.
- Multiplication complexity reduces from O(n^2) to O($n \log(n)$).

Classification of Lattices (Simplified)

- Lattice-based cryptographic schemes generally fall under three classes:

LWE ←→ Module-LWE ←→ Ring-LWE

- Added structures hinder security:

LWE $\geq^{\text{sec.}}$ Module-LWE $\geq^{\text{sec.}}$ Ring-LWE

- However, it can also enhance performance:

LWE $\leq^{\text{per.}}$ Module-LWE $\leq^{\text{per.}}$ Ring-LWE

Modules: Multiplication

- NTTs typically aren't generic; require ad-hoc designs.
- Research done investigating high-performance vs. low-cost designs.
- Some candidates specify NTTs explicitly, i.e. NewHope.
- NTTs get modular reduction for free, but restrict parameters (e.g. requiring a prime modulus).
- Matrix / schoolbook / Karatsuba multiplication more generic.
- General multiplication has more liberal parameter selection, but requires modular reduction.
- Sparse multiplication is used often in signature schemes and LWR, using binary or ternary values, which can simply use shift-and-adds.

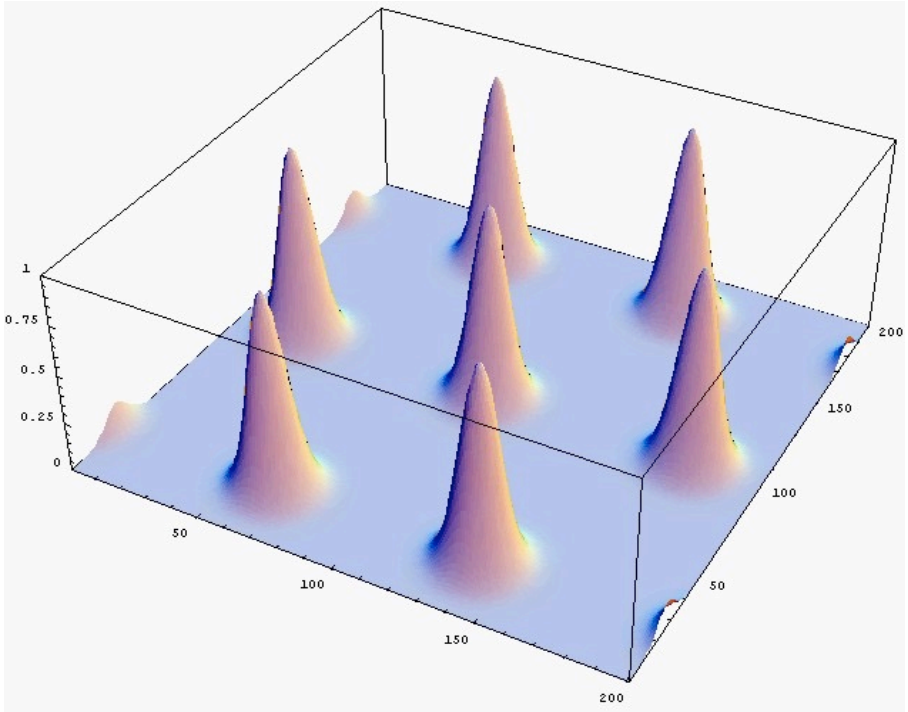
Modules: Multiplication

- Typically require <<32-bit integer multiplication (no floating points) most actually <16 bits.
- Thus, DSPs are ideal for MAC (or just multiply) operations.
- BRAMs typically used for key / input / output storage.
- Inputs drawn from memory, PRNG, and/or error sampler.
- Most candidates provide constant-time multiplication.

Modules: Error Samplers

- Error adds noise to computations on secret data; computationally hard.

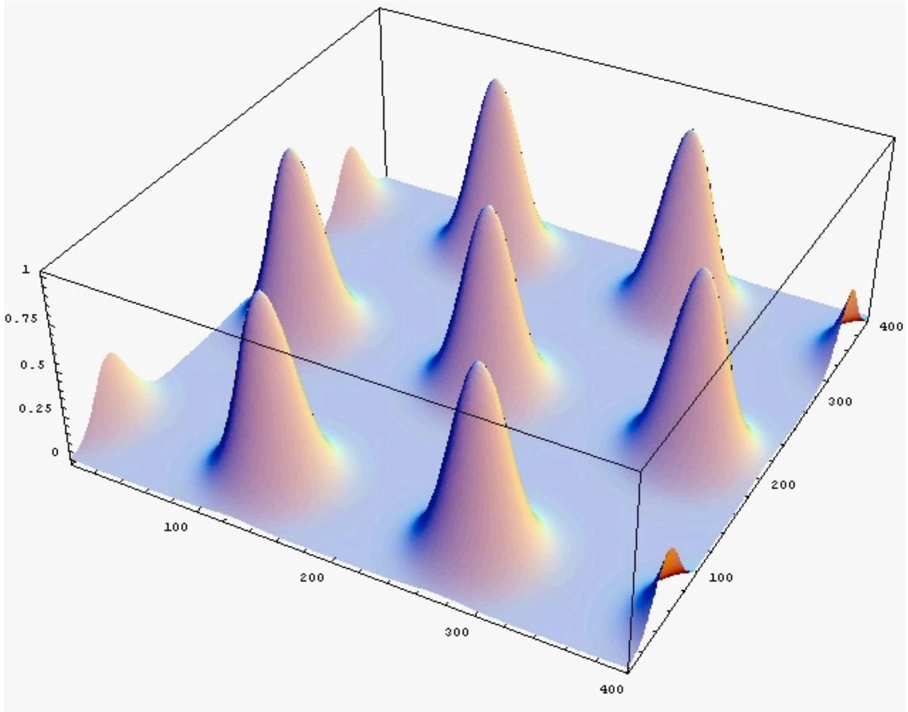
For $\boldsymbol{B} = \boldsymbol{A} * \boldsymbol{S} + \boldsymbol{E}$



Modules: Error Samplers

- Error adds noise to computations on secret data; computationally hard.

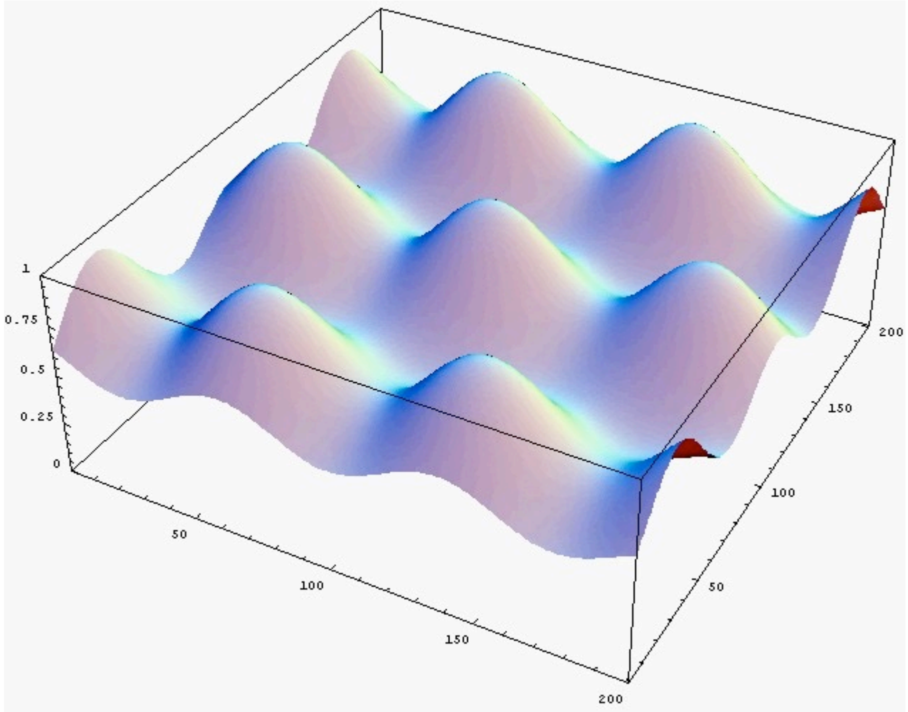
For $\boldsymbol{B} = \boldsymbol{A} * \boldsymbol{S} + \boldsymbol{E}$



Modules: Error Samplers

- Error adds noise to computations on secret data; computationally hard.

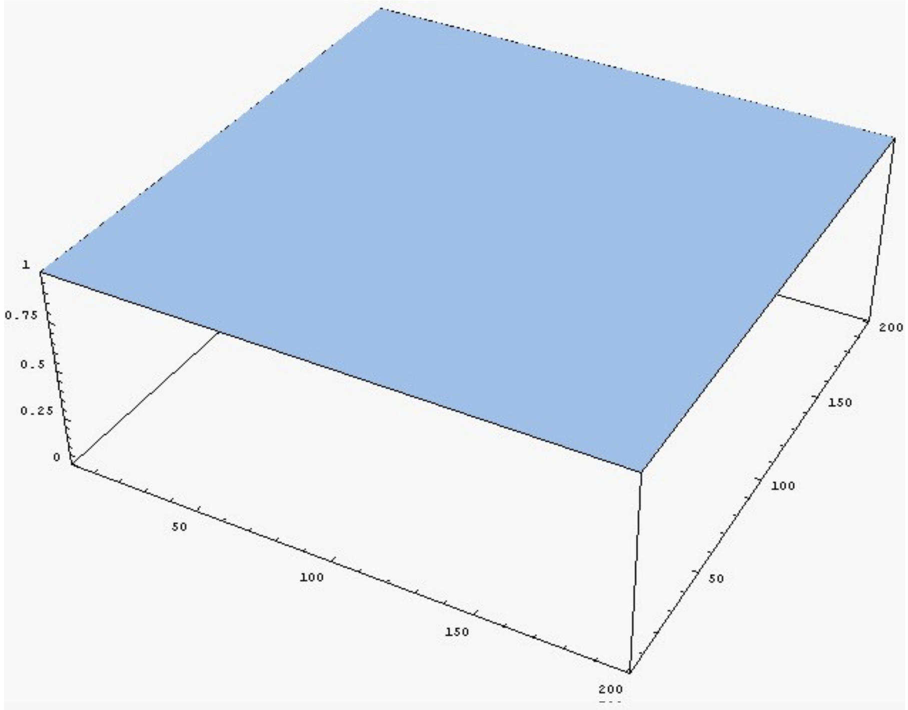
For $\boldsymbol{B} = \boldsymbol{A} * \boldsymbol{S} + \boldsymbol{E}$



Modules: Error Samplers

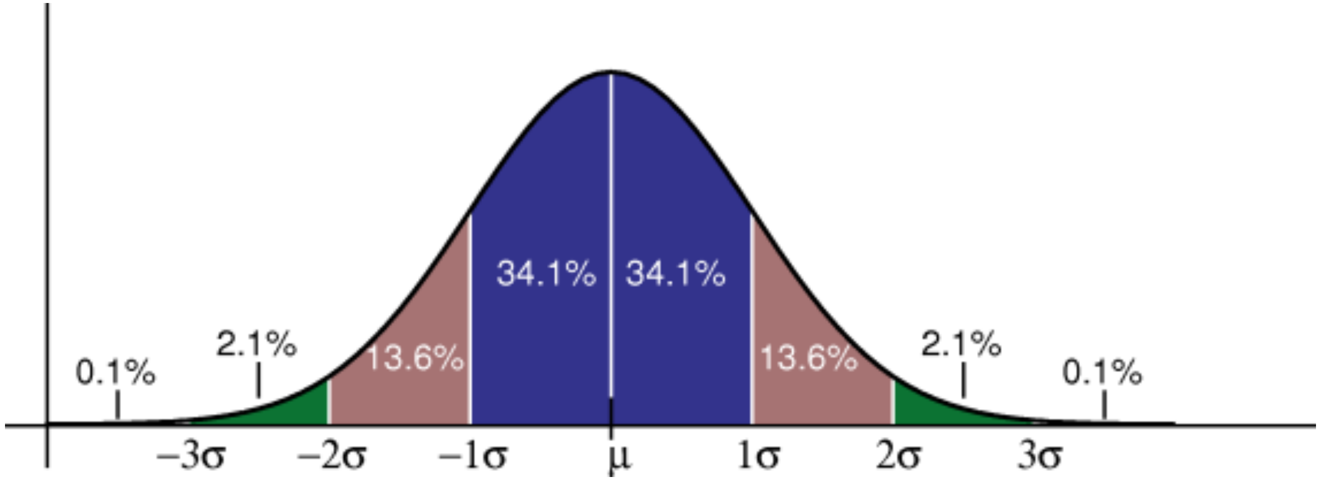
- Error adds noise to computations on secret data; computationally hard.

For $\boldsymbol{B} = \boldsymbol{A} * \boldsymbol{S} + \boldsymbol{E}$



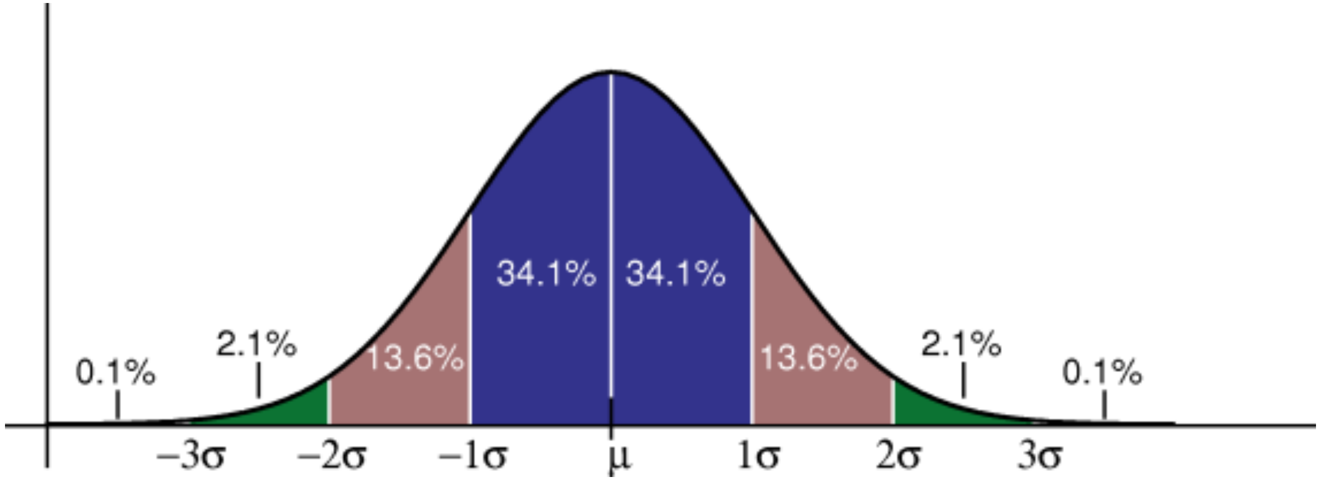
Modules: Error Samplers

- Error adds noise to computations on secret data; computationally hard.
- Error sampled from Gaussian-like or Binomial distribution.
- Look-up table methods: CDT sampler.
- Arithmetic-based methods: discrete Ziggurat sampler.
- Hybrid table / arithmetic methods: Bernoulli and Knuth-Yao samplers.
- Standard deviations depend on cryptographic schemes and parameters:



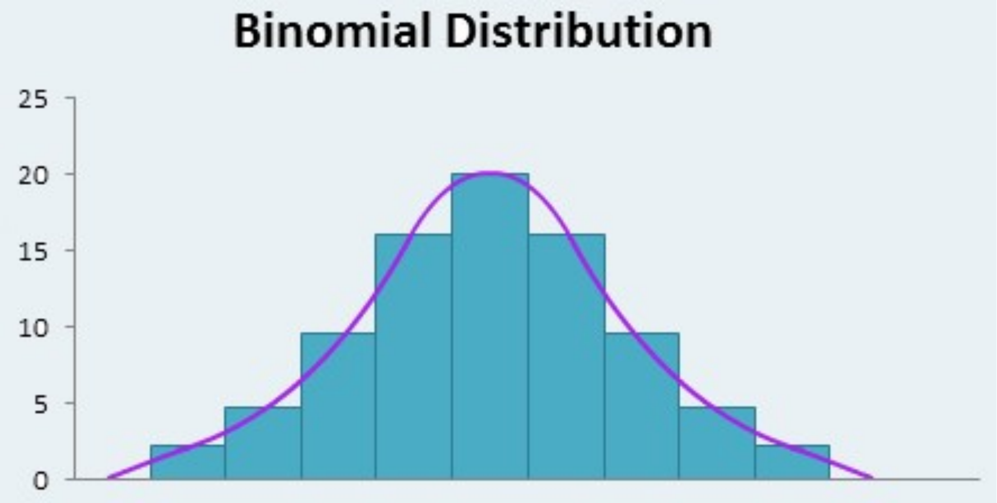
Modules: Error Samplers

- Error samplers linked to computational hardness, thus a side-channel target.
- Important to ensure independent-time design (e.g. constant time).
- Some recent research considers masked and fault attack protection for these modules.
- One can use Gaussian convolutions to make larger parameters efficient, e.g. for signature schemes.



Modules: Error Sampling

- Alternatively, some schemes (NewHope, Kyber) use Binomial sampling.
- One simply subtracts the Hamming weight of two uniform bit vectors.
- LWR schemes instead use ‘rounding’ instead of error addition.
- Dilithium (and maybe others?) uses uniform random noise.



Module Lattices

- (Ring-)LWE deals with vectors/polynomials in R_q^1 , for example $\boldsymbol{A} * \boldsymbol{S} + \boldsymbol{E}$.
- Module-LWE deals with polynomials in R_q^k , for example $k = 3$ in Kyber.
- Higher security parameters increase k, instead of n.
- Thus, virtually no re-implementation for changing security levels.
- “One way to informally view the MLWE problem is to take the RLWE problem and replace the single ring elements ($\boldsymbol{A}$ and $\boldsymbol{s}$) with module elements over the same ring. Using this intuition, RLWE can be seen as MLWE with module rank $k = 1$.”

$$\begin{bmatrix} A_1(x) \in R_q & A_2(x) \in R_q \\ A_3(x) \in R_q & A_4(x) \in R_q \end{bmatrix} \times \begin{bmatrix} S_1(x) \in R_q \\ S_2(x) \in R_q \end{bmatrix} + \begin{bmatrix} \mathrm{E}_1(x) \in R_q \\ \mathrm{E}_2(x) \in R_q \end{bmatrix}$$

Learning With Rounding

- SABER uses module-LWR problem.
- Polynomials are always of $n = 256$ coefficients.
- Flexibility: matrix dimensions (k) is parameterizable.
- 2-by-2 for 115-bit post-quantum security

Light SABER

- 3-by-3 for 180-bit post-quantum security

SABER

- 4-by-4 for 245-bit post-quantum security

Fire SABER

Differences in LWE and LWR

- SABER uses module-LWR problem.

$$\left\lfloor \frac{p}{q} \boxed{\text{Uniform in [0, q-1]}} \right\rceil$$

where p < q

- Rounds a product $\boldsymbol{p} = \boldsymbol{a} * \boldsymbol{s}$ to the nearest integer.

Prime q introduces rounding bias

- Cannot use prime q ☹
- Hence, no NTT-based fast polynomial multiplication

→ Thus, one needs to use generic polynomial multiplication algorithm.

Generic Polynomial Multiplication

- SABER uses hybrid of Toom-Cook, Karatsuba, and schoolbook multiplication.
- Generic techniques (Toom-3, Toom-4, Karatsuba) can be applied to SABER, NTRU-HRSS, and NTRUEncrypt.
- NTRU Prime also uses non-NTT multiplication.
- Round5 only requires LHW shift-and-add multiplication.
- Generic hardware techniques have been researched for Ring-TESLA.

Lattice-based Signatures in Hardware

A hardware design of Ring-TESLA

Generic Polynomial Multiplication

- qTESLA is (somewhat) based upon the signature scheme; Ring-TESLA.

KeyGen($\boldsymbol{a_1}, \boldsymbol{a_2}$):
Discrete Gaussian polynomials: $\boldsymbol{s}, \boldsymbol{e}_1, \boldsymbol{e}_2 \leftarrow D_\sigma^n$, $\mathbf{t}_1 \equiv \boldsymbol{a}_1 \boldsymbol{s} + \boldsymbol{e}_1 \bmod q$, $\mathbf{t}_2 \equiv \boldsymbol{a}_2 + \boldsymbol{e}_2 \bmod q$
Secret-Key: $(\boldsymbol{s}, \boldsymbol{e}_1, \boldsymbol{e}_2)$ // Public-Key: $(\boldsymbol{t}_1, \boldsymbol{t}_2)$.

Sign($\boldsymbol{\mu}; \boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{s}, \boldsymbol{e}_1, \boldsymbol{e}_2$):

Uniform polynomial: $\boldsymbol{y} \leftarrow \mathbb{Z}_q[\boldsymbol{x}]/(\boldsymbol{x}^n + 1)$
- $\boldsymbol{v}_1 \equiv \boldsymbol{a}_1 \boldsymbol{y} \bmod q, \boldsymbol{v}_2 \equiv \boldsymbol{a}_2 \boldsymbol{y} \bmod q$

Compute the hash function:
- $\boldsymbol{c} = H(\boldsymbol{v}_1 || \boldsymbol{v}_2, \mu)$

Compute signature/rejections:
- $\boldsymbol{z} \equiv \boldsymbol{y} + \boldsymbol{sc}$ ←signature
- $\boldsymbol{w}_1 \equiv \boldsymbol{v}_1 + \boldsymbol{e}_1 \boldsymbol{c} \bmod q$
- $\boldsymbol{w}_2 \equiv \boldsymbol{v}_2 + \boldsymbol{e}_2 \boldsymbol{c} \bmod q$

Verify($\boldsymbol{\mu}; \boldsymbol{z}, \boldsymbol{c}; \boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{t}_1, \boldsymbol{t}_2$):

Compute hash inputs:
- $\boldsymbol{w}_1' \equiv \boldsymbol{a}_1 \boldsymbol{z} + \boldsymbol{t}_1 \boldsymbol{c} \bmod q$
- $\boldsymbol{w}_2' \equiv \boldsymbol{a}_2 \boldsymbol{z} + \boldsymbol{t}_2 \boldsymbol{c} \bmod q$

Compute the hash function:
- $\boldsymbol{c}'' = H(\boldsymbol{w}_1' || \boldsymbol{w}_2', \mu)$

Accept/reject signature:
- **If** $\boldsymbol{c}' = \boldsymbol{c}''$

128-bit security parameters:
$n = 512$,
$q = 51750913$,
$\sigma = 52$.

Signature is 11.9 kb,
public-key is 26 kb,
and secret-key is 13.7 kb.

1) Sedat Akleylek, Nina Bindel, Johannes A. Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. An efficient lattice-based signature scheme with provably secure instantiation. In AFRICACRYPT, pages 44–60, 2016.
2) Howe, J., Rafferty, C., Khalid, A. and O'Neill, M., 2017. Compact and provably secure lattice-based signatures in hardware. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1-4). IEEE.

Ring-TESLA in Hardware

Algorithm 1 Ring-TESLA Sign

procedure $\textsc{Sign}(\mu, \mathbf{a}_1, \mathbf{a}_2, \mathbf{s}, \mathbf{e}_1, \mathbf{e}_2)$

$\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$

$\mathbf{v}_1 \equiv \mathbf{a}_1\mathbf{y} \mod q$

$\mathbf{v}_2 \equiv \mathbf{a}_2\mathbf{y} \mod q$

$c = H(\lfloor \mathbf{v}_1 \rceil_{d,q}, \lfloor \mathbf{v}_2 \rceil_{d,q}, \mu)$

$\mathbf{c} = F(c)$

$\mathbf{z} \leftarrow \mathbf{y} + \mathbf{s}\mathbf{c}$

$\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$

$\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$

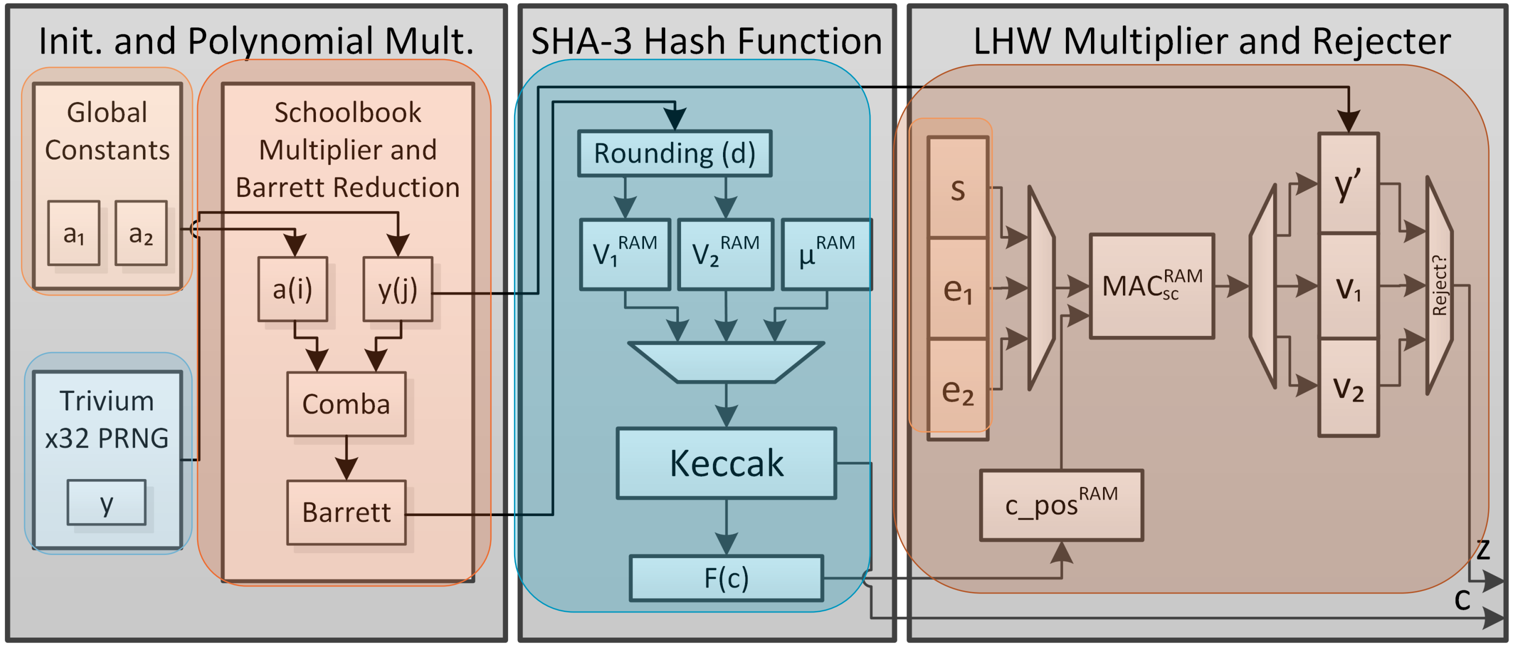
if $[\mathbf{w}_1]_{2^d}, [\mathbf{w}_2]_{2^d} \notin \mathcal{R}_{2^d-L}$ or $\mathbf{z} \notin \mathcal{R}_{B-U}$ **then**

Restart

end if

return $(\mathbf{z}, c)$

end procedure



Finite-State Machine of Ring-TESLA

Algorithm 1 Ring-TESLA Sign

procedure $\textsc{Sign}(\mu, \mathbf{a}_1, \mathbf{a}_2, \mathbf{s}, \mathbf{e}$

$\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$

$\mathbf{v}_1 \equiv \mathbf{a}_1\mathbf{y} \mod q$

$\mathbf{v}_2 \equiv \mathbf{a}_2\mathbf{y} \mod q$

$c = H(\lfloor \mathbf{v}_1 \rceil_{d,q}, \lfloor \mathbf{v}_2 \rceil_{d,q}, \mu)$

$\mathbf{c} = F(c)$

$\mathbf{z} \leftarrow \mathbf{y} + \mathbf{sc}$

$\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$

$\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$

Pre-Hash

Hash

Post-Hash

23

Finite-State Machine of Ring-TESLA

Algorithm 1 Ring-TESLA Sign

procedure $\textsc{Sign}(\mu, \mathbf{a}_1, \mathbf{a}_2, \mathbf{s}, \mathbf{e}$

$$\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$$
$$\mathbf{v}_1 \equiv \mathbf{a}_1\mathbf{y} \mod q$$
$$\mathbf{v}_2 \equiv \mathbf{a}_2\mathbf{y} \mod q$$
$$c = H(\lfloor \mathbf{v}_1 \rceil_{d,q}, \lfloor \mathbf{v}_2 \rceil_{d,q}, \mu)$$
$$\mathbf{c} = F(c)$$
$$\mathbf{z} \leftarrow \mathbf{y} + \mathbf{s}\mathbf{c}$$
$$\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$$
$$\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$$

- Pipeline created for pre-hash computations.
- After pre-hash polynomial multiplication;
  - $\boldsymbol{y}$ is copied to another register for $\boldsymbol{z}$.
  - $\boldsymbol{y}$ is generated for next signature in parallel.
- Hash, LHW calculations of $\boldsymbol{z}$, $\boldsymbol{w_1}$, and $\boldsymbol{w_2}$, and rejections then outside the critical path.
- Sign/Verify critical path thus pre-hash phase.

Signature ♯1	Poly. Mult. ⇒	Hash ⇒	LHW					
Signature ♯2		Poly. Mult. ⇒		Hash ⇒	LHW			
⋮				⋱				
Signature ♯n					Poly. Mult. ⇒		Hash ⇒	LHW

Ring-TESLA Hardware Results

- Ring-TESLA, ideal lattice-based signatures on a Spartan 6 – LX25.
- Smaller than other lattice-based signature designs, suffers in throughput.
- Significantly smaller and faster in comparison to RSA and ECDSA.
- Further work generated hardware friendly parameters.

	Operation, Configuration	Security	Device	LUT/FF/SLICE	BRAM/DSP	MHz	Cycles	Ops/sec
1	Ring-TESLA (Sign, SB-I)	128-bits	S6 LX25	4447/3345/1257	3/6	190	1835540	104
2	Ring-TESLA (Sign, SB-II)	128-bits	S6 LX25	4828/3790/1513	4/8	196	917771	214
4	Ring-TESLA (Sign, SB-IV)	128-bits	S6 LX25	5071/3851/1503	4/12	187	458891	408
8	Ring-TESLA-(Sign, SB-VIII)	128-bits	S6 LX25	6848/5457/2254	4/20	180	229446	785
1	Ring-TESLA (Verify, SB-I)	128-bits	S6 LX25	3714/3023/1172	3/6	188	1835540	102
2	Ring-TESLA (Verify, SB-II)	128-bits	S6 LX25	3917/3253/1238	3/8	194	917771	212
4	Ring-TESLA (Verify, SB-IV)	128-bits	S6 LX25	4793/3939/1551	3/12	186	458891	406
8	Ring-TESLA (Verify, SB-VIII)	128-bits	S6 LX25	6473/5582/2103	3/20	178	229446	776
	GLP (Sign, Schoolbook x3)	80-bits	S6 LX16	7465/8993/2273	30/28	162	-	931
	GLP (Verify, Schoolbook x3)	80-bits	S6 LX16	6225/6663/2263	15/8	158	-	998
	BLISS (Sign, NTT)	128-bits	S6 LX25	7193/6420/2291	6/5	139	15864	8761
	BLISS (Verify NTT)	128-bits	S6 LX25	5065/4312/1687	4/3	166	16346	17101
	RSA (Sign)	103-bits	V5 LX30	3237 slices	7/17	200	-	89
	ECDSA (Sign)	128-bits	V5 LX110	32299 LUT/FF pairs	10/37	139	-	-
	ECDSA (Verify)	128-bits	V5 LX110	32299 LUT/FF pairs	10/37	110	-	

Frodo: Take off the Ring!

Practical post-quantum key exchange and key encapsulation from LWE.

Frodo: Why Should We Take off the Ring?

The design philosophy of FrodoKEM combines:

- Conservative yet practical post-quantum constructions.
- Security derived from cautious parameterizations of the well-studied learning with errors problem.
- Thus, close connections to conjectured-hard problems on generic, “algebraically unstructured” lattices.
- Parameter selection is far less constrained than vs ideal lattice schemes.
- FrodoKEM multiplication can also be generic.

Frodo: Why Should We Take off the Ring?

These qualities are appealing for practitioners;

- Probably the most secure lattice-based candidate.
    - Many IoT use cases require long-term, efficient cryptography.
- Frodo is ideal for long-term security and constrained platforms.
    - Suitable for use cases such as satellite communications and V2X.
- Frodo is extremely versatile and theoretically sound.
- However, it has less implementations than ideal lattice schemes.
    - And how do we manage the larger keys and no NTT...

Frodo: Why Should We Take off the Ring?

- Simple design:
  - Free modular arithmetic ($q = 2^{16}$).
  - Simple Gaussian sampling.
  - Parallelisable matrix-vector operations.
  - Key encapsulation without reconciliation.
  - Simple code, no complex use of NTT.
- CCA-secure with negligible error rate.
- Flexible, fine-grained choice of parameters.
- Dynamically generated $\boldsymbol{A}$ to defend against all-for-the-price-of-one attacks (AES and cSHAKE variants).

Frodo: Why Should We Take off the Ring?

- Round 2 changes add high-security parameters and use of SHAKE.
- Main operations are of the form from before:

$$\boldsymbol{B} = \boldsymbol{S}' * \boldsymbol{A} + \boldsymbol{E} \bmod q$$

- $\boldsymbol{S}'$ is a matrix with dimensions 8-by-640 (or 8-by-976).
- $\boldsymbol{A}$ is a matrix with dimensions 640-by-640 (or 976-by-976).
- Thus, we design a LWE vector-matrix multiplication core, and repeat.
- DSPs are ideal; Artix-7 FPGAs have 48-bit MAC operations.
- q is always a power-of-two, thus modular reduction is free!
- Uniform and “Gaussian” error generation.
- Random oracles via cSHAKE for CCA security.

FrodoKEM in Hardware

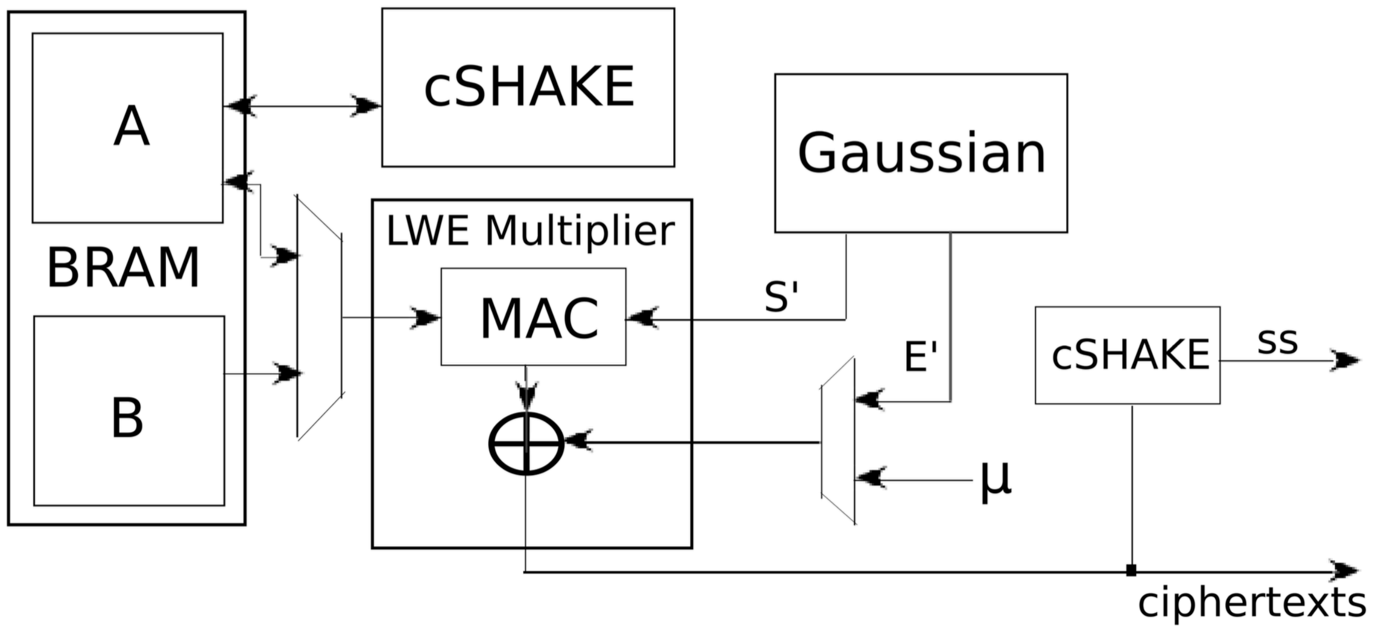
"A massive design challenge was to balance **memory utilisation**, whilst not deteriorating the **performance** too much to not overexert the limited computing capabilities of the embedded devices."

FrodoKEM in Hardware

- Proposes a generic LWE multiplication core which computes vector-matrix multiplication and error addition.
- Generates future random values in parallel, minimising delays between vector-matrix multiplications.
- Hybrid pre-calculated / on-the-fly memory management is used, which continuously updates previous values.
- Ensures constant runtime by parallelising other modules with multiplication.
- FrodoKEM-640 has a total execution time of 60 ms, running at 167MHz.

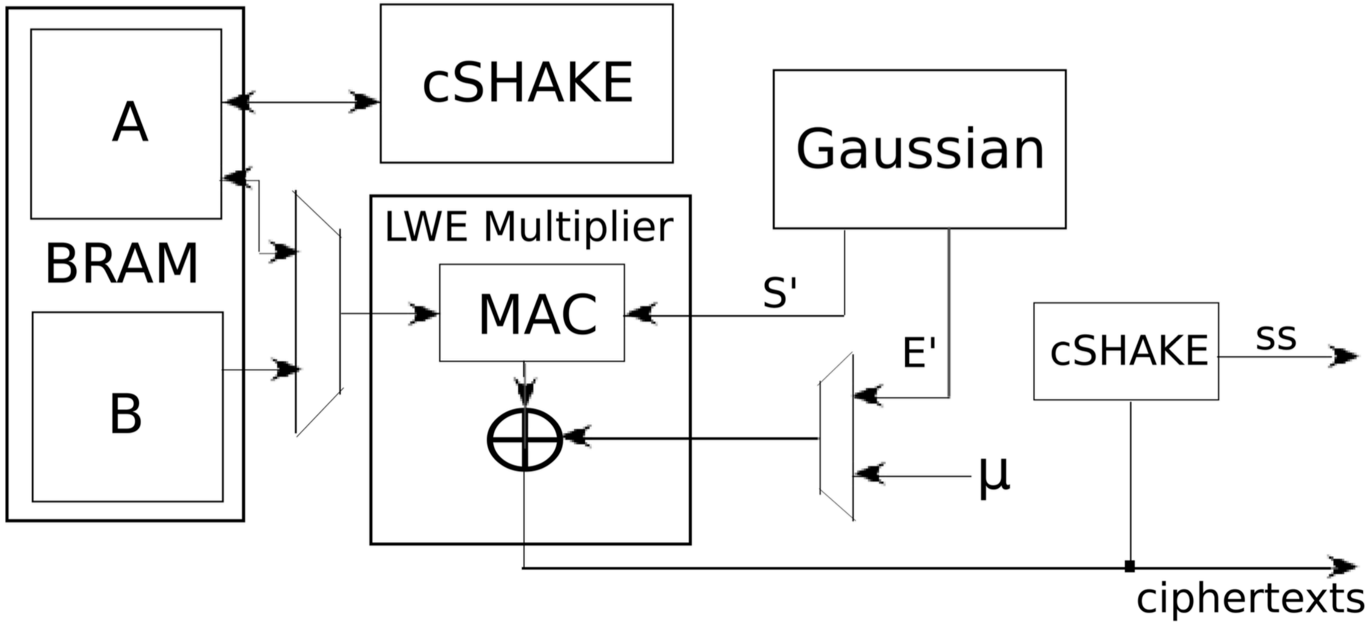
FrodoKEM in Hardware

- Similarities in KeyGen, Encaps, and Decaps mean much of this is reused.
- Most of the generation of $\boldsymbol{A}$ is done on-the-fly to save BRAM.
- LWE multiplier is reused in all modules and all LWE calculations.



FrodoKEM in Hardware

- For $\boldsymbol{S'} * \boldsymbol{A}$ we generate the first row of S' and enough randomness in $\boldsymbol{A}$.
- Whilst they multiply, we use ping-pong buffering to generate future values.
- This removes latency and ensures a practical constant-time design.



FrodoKEM in Hardware

- Competes with NewHope area consumption, but much slower performance.
- Due to memory optimisations, we have huge savings in BRAM compared to LWE Encryption [HMO+16].
- Results also provided for FrodoKEM's modules; that is cSHAKE and Error sampling.

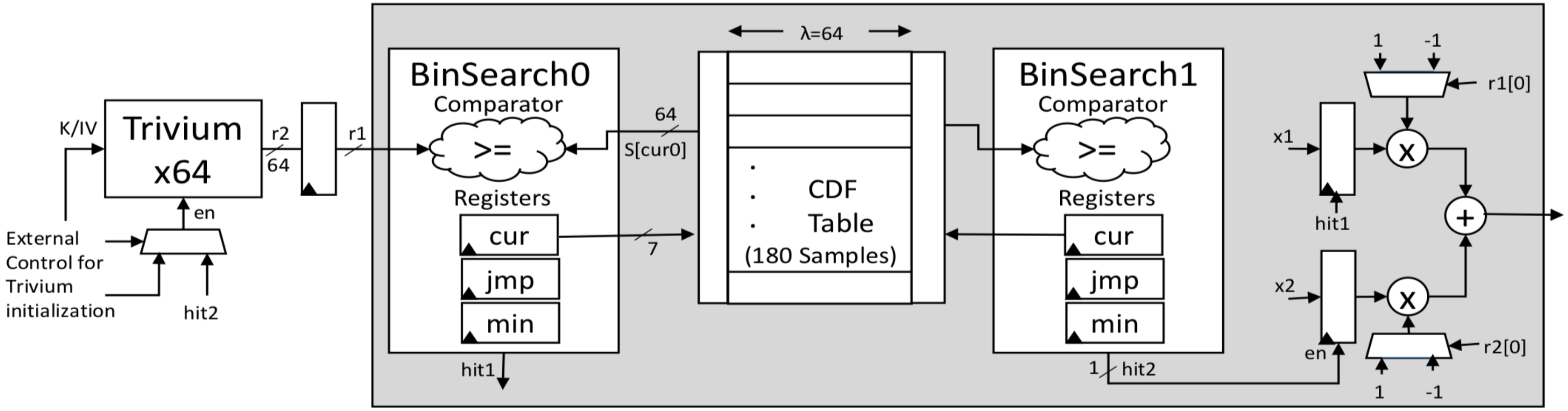
Cryptographic Operation	LUT/FF	Slice	DSP	BRAM	MHz	Ops/sec
FrodoKEM-640 Keypair[2]	3771/1800	1035	1	6	167	51
FrodoKEM-640 Encaps	6745/3528	1855	1	11	167	51
FrodoKEM-640 Decaps	7220/3549	1992	1	16	162	49
FrodoKEM-976 Keypair[2]	7139/1800	1939	1	8	167	22
FrodoKEM-976 Encaps	7209/3537	1985	1	16	167	22
FrodoKEM-976 Decaps	7773/3559	2158	1	24	162	21
cSHAKE*	2744/1685	766	0	0	172	1.2m
Error+AES Sampler*	1901/1140	756	0	0	184	184m
NewHopeUSENIX Server [OG17]	5142/4452	1708	2	4	125	731
NewHopeUSENIX Client [OG17]	4498/4635	1483	2	4	117	653
LWE Encryption [HMO+16]	6078/4676	1811	1	73	125	1272

Other / Miscellaneous

(Don’t worry, its nearly over!)

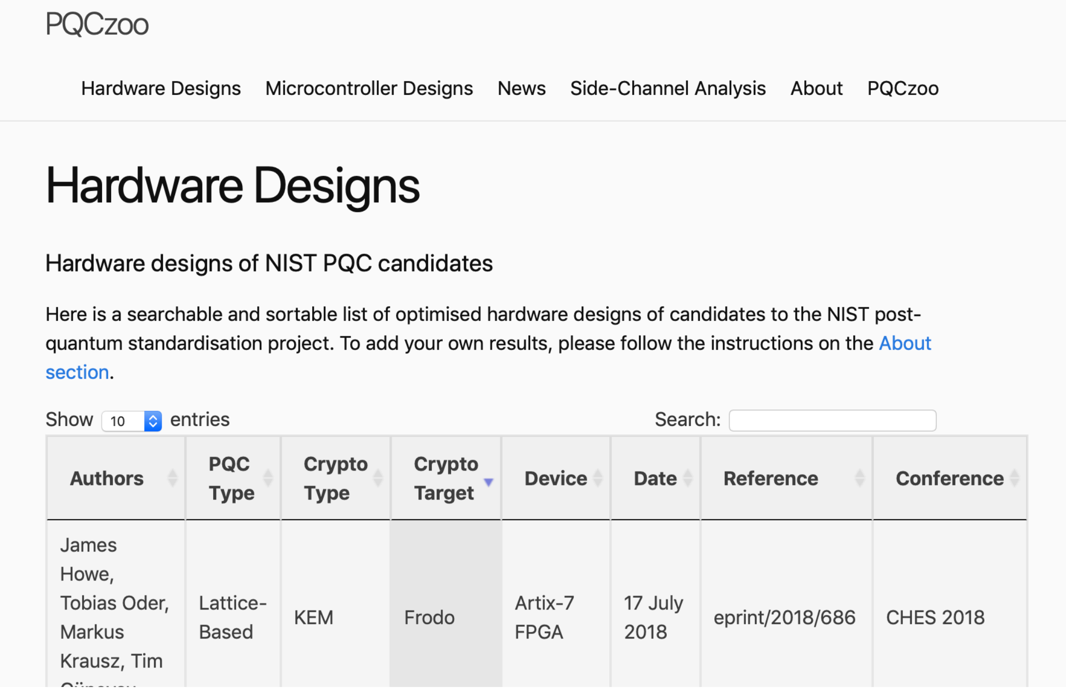
Other: Gaussian Sampling Designs

- In a comprehensive study we found CDT sampling the most efficient in hardware, running in constant-time is key for these modules.
- Survey available on error samplers for Round 1 candidates.
- Gaussian convolution tricks can be used to make these efficient for large parameters, which provide some ‘masking’ for free.
- Simple tricks can make these modules protected against fault attacks.



Other: PQCzoo.com

- PQCzoo.com is a website collecting results for optimised software and hardware designs as well as side-channel analysis papers.
- One can add their own results with a simple GitHub commit.
- Please add your own results!



PQCzoo

Hardware Designs   Microcontroller Designs   News   Side-Channel Analysis   About   PQCzoo

Hardware Designs

Hardware designs of NIST PQC candidates

Here is a searchable and sortable list of optimised hardware designs of candidates to the NIST post-quantum standardisation project. To add your own results, please follow the instructions on the About section.

Show 10 entries   Search:

Authors	PQC Type	Crypto Type	Crypto Target	Device	Date	Reference	Conference
James Howe, Tobias Oder, Markus Krausz, Tim	Lattice-Based	KEM	Frodo	Artix-7 FPGA	17 July 2018	eprint/2018/686	CHES 2018

Conclusion

- Most Round 2 schemes have yet to be implemented in hardware.
- But, many require aspects that have already been researched.
  - I’ve put together a list of references which should be helpful.
- Important that future designs specify design philosophy.
  - e.g. high throughput or low area.
- Also, these designs should be evaluated on the same FPGA.
  - e.g. the Xilinx Artix-7 FPGA.
- This ensures comparisons between hardware designs are fair and straightforward.

PQShield is hiring software/hardware post-quantum specialists.

Useful References: PhD Theses

- Howe, J., 2017. Practical Lattice-Based Cryptography in Hardware. https://jameshowe.eu/files/thesis.pdf
- Pöppelmann, T., 2017. Efficient implementation of ideal lattice-based cryptography. it-Information Technology, 59(6), pp.305-309. https://hss-opus.ub.ruhr-uni-bochum.de/opus4/frontdoor/index/index/docId/4917
- Roy, S. S., 2017. Public Key Cryptography on Hardware Platforms: Design and Analysis of Elliptic Curve and Lattice-based Cryptoprocessors. https://www.esat.kuleuven.be/cosic/publications/thesis-288.pdf

Useful References: Multiplication

- Pöppelmann, T. and Güneysu, T., 2012, October. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In International Conference on Cryptology and Information Security in Latin America (pp. 139-158). Springer, Berlin, Heidelberg.
- Aysu, A., Patterson, C. and Schaumont, P., 2013, June. Low-cost and area-efficient FPGA implementations of lattice-based cryptography. In 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) (pp. 81-86). IEEE.
- Kannwischer, M.J., Rijneveld, J. and Schwabe, P., 2018. Faster multiplication in Z2m [x] on Cortex-M4 to speed up NIST PQC candidates. Cryptology ePrint Archive, Report 2018/1018, 2018. https://eprint.iacr.org/2018/1018.
- Howe, J., Rafferty, C., Khalid, A. and O'Neill, M., 2017. Compact and provably secure lattice-based signatures in hardware. In 2017 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-4). IEEE.

Useful References: Error Samplers

- Dwarakanath, N.C. and Galbraith, S.D., 2014. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. Applicable Algebra in Engineering, Communication and Computing, 25(3), pp.159-180.
- Karmakar, A., Roy, S.S., Reparaz, O., Vercauteren, F. and Verbauwhede, I., 2018. Constant-time discrete gaussian sampling. IEEE Transactions on Computers, 67(11), pp.1561-1571.
- Howe, J., Khalid, A., Rafferty, C., Regazzoni, F. and O'Neill, M., 2018. On practical discrete Gaussian samplers for lattice-based cryptography. IEEE Transactions on Computers, 67(3), pp.322-334.
- Khalid, A., Rafferty, C., Howe, J., Brannigan, S., Liu, W. and O'Neill, M., 2018, October. Error Samplers for Lattice-Based Cryptography-Challenges, Vulnerabilities and Solutions. In 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS) (pp. 411-414). IEEE.
- Howe, J., Khalid, A., Martinoli, M., Regazzoni, F. and Oswald, E., Fault Attack Countermeasures for Error Samplers in Lattice-Based Cryptography. IACR Cryptology ePrint Archive 2019: 206 (2019).
- Schneider, T., Paglialonga, C., Oder, T. and Güneysu, T., Efficiently Masking Binomial Sampling at Arbitrary Orders for Lattice-Based Crypto. https://www.emsec.ruhr-uni-bochum.de/media/seceng/veroeffentlichungen/2019/02/01/crv.pdf

Useful References: Hardware Surveys

- Nejatollahi, H., Dutt, N., Ray, S., Regazzoni, F., Banerjee, I. and Cammarota, R., 2019. Post-Quantum Lattice-Based Cryptography Implementations: A Survey. ACM Computing Surveys (CSUR), 51(6), p.129.
- Howe, J., Pöppelmann, T., O'Neill, M., O'Sullivan, E. and Güneysu, T., 2015. Practical lattice-based digital signature schemes. ACM Transactions on Embedded Computing Systems (TECS), 14(3), p.41.
- Howe, J., Khalid, A., Rafferty, C., Regazzoni, F. and O'Neill, M., 2018. On practical discrete Gaussian samplers for lattice-based cryptography. IEEE Transactions on Computers, 67(3), pp.322-334.
- Khalid, A., Rafferty, C., Howe, J., Brannigan, S., Liu, W. and O'Neill, M., 2018, October. Error Samplers for Lattice-Based Cryptography-Challenges, Vulnerabilities and Solutions. In 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS) (pp. 411-414). IEEE.
- Dwarakanath, N.C. and Galbraith, S.D., 2014. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. Applicable Algebra in Engineering, Communication and Computing, 25(3), pp.159-180.

Useful References: Open Resources

- Post-Quantum Cryptography - Ruhr-Universität Bochum. https://www.seceng.ruhr-uni-bochum.de/research/projects/pqc/
- PQM4: Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4
- KECCAK in hardware. https://keccak.team/hardware.html
- PQCzoo. https://pqczoo.com