CONSTANT TIME HARDWARE IMPLEMENTATION OF STREAMLINED NTRU PRIME

CARDIS PRESENTATION

ADRIAN MAROTZKE

NOVEMBER 2020

Hamburg University of Technology

Agenda

- Introduction to Streamlined NTRU Prime
- Algorithm description
- Implementation overview
- Implementation details
- Conclusion & future work

Hamburg University of Technology

Streamlined NTRU Prime – A primer

- Alternate finalist in round 3 of NIST PQC Standardization
- Key Encapsulation Mechanism
- Lattice based scheme
- Ancestry: 1998 NTRU, with several improvements

Streamlined NTRU Prime – A primer

- Reducing attack surface at low cost :

No decryption failures $\longrightarrow$ No Gaussian sampling. Rounding of ciphertexts, fixed weight secrets

No cyclotomic rings $\longrightarrow$

$$\mathcal{R}/q = (\mathbb{Z}/q)[x]/(x^p - x - 1)$$
$$\mathcal{R}/3 = (\mathbb{Z}/3)[x]/(x^p - x - 1)$$

p, q both prime

Key confirmation hash $\longrightarrow$ Hash public key + secret input, simplifies security analysis

Streamlined NTRU Prime – A primer

- Polynomials with all coefficients $\in \{-1, 0, 1\}$ called *small*.
- Polynomials have *weight* w iff exactly w coefficients are non-zero
- Polynomials are *short* iff they are both *small* and have *weight* w
- Hash_a(x): SHA-512 of x, prefixed by single byte value a
- Encode/Decode: Field Elements ←→ Byte strings

CORE-SVP Security Level	p	q	w
2^{129}	653	4621	250
2^{153}	761	4591	286
2^{175}	857	5167	322

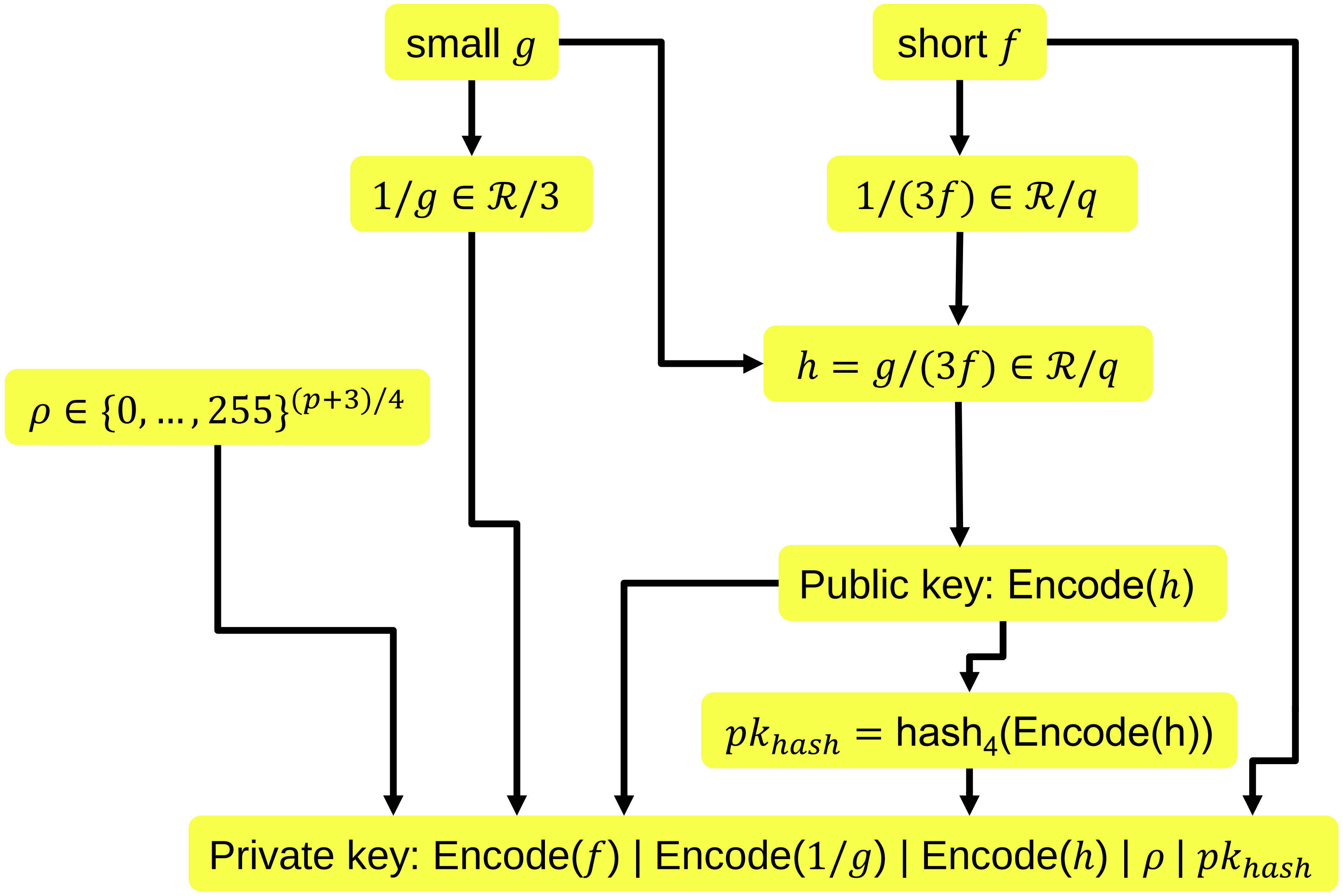
TUHH Hamburg University of Technology

Agenda

- Introduction to Streamlined NTRU Prime
- Algorithm description
- Implementation overview
- Implementation details
- Conclusion & future work

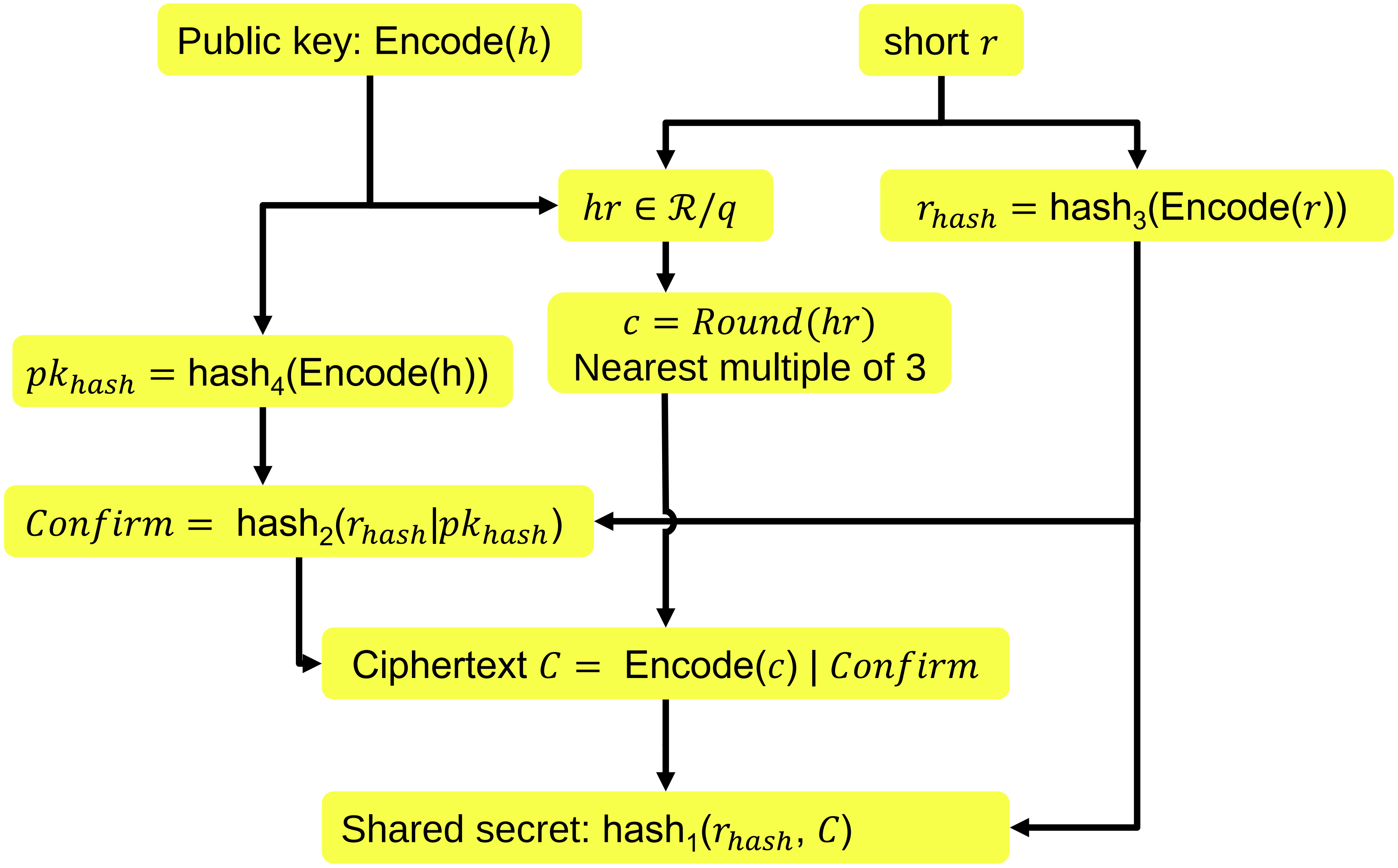
Hamburg University of Technology

Key Generation

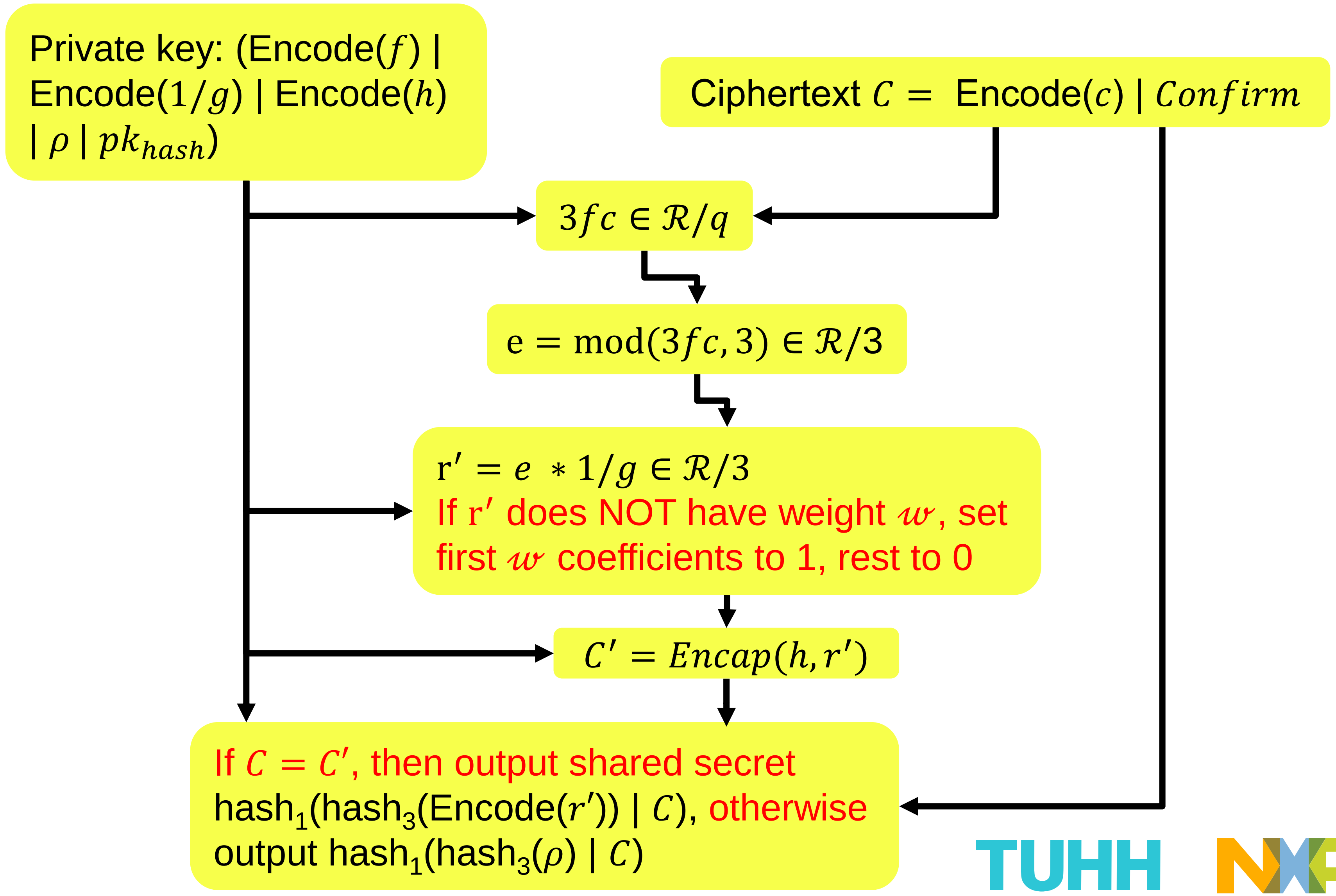

small g
short f
$1/g \in \mathcal{R}/3$
$1/(3f) \in \mathcal{R}/q$
$h = g/(3f) \in \mathcal{R}/q$
$\rho \in \{0, \ldots, 255\}^{(p+3)/4}$
Public key: Encode(h)
pk_{hash} = hash$_4$(Encode(h))
Private key: Encode(f) | Encode($1/g$) | Encode(h) | ρ | pk_{hash}

Hamburg University of Technology

Encapsulation


Public key: Encode(h)
short r
$hr \in \mathcal{R}/q$
$r_{hash} = \text{hash}_3(\text{Encode}(r))$
$c = Round(hr)$
Nearest multiple of 3
$pk_{hash} = \text{hash}_4(\text{Encode(h)})$
$Confirm = \text{hash}_2(r_{hash} | pk_{hash})$
Ciphertext $C = \text{Encode}(c) \mid Confirm$
Shared secret: $\text{hash}_1(r_{hash}, C)$

Decapsulation



Private key: (Encode(f) | Encode($1/g$) | Encode(h) | ρ | pk_{hash})

Ciphertext $C =$ Encode(c) | $Confirm$

$3fc \in \mathcal{R}/q$

$e = \mathrm{mod}(3fc, 3) \in \mathcal{R}/3$

$r' = e * 1/g \in \mathcal{R}/3$
If r' does NOT have weight w, set first w coefficients to 1, rest to 0

$C' = Encap(h, r')$

If $C = C'$, then output shared secret hash$_1$(hash$_3$(Encode(r')) | C), otherwise output hash$_1$(hash$_3$(ρ) | C)

Agenda

Hamburg University of Technology

Overview

- Hardware Implementation of Streamlined NTRU Prime
- All operations are supported, all round 2 parameter sets
- For small embedded systems, e.g. smartcards
- Source code: https://github.com/AdrianMarotzke/SNTRUP

Overview

- All numbers are for parameter set SNTRUP761 and Xilinx Zynq ZCU102

Operation	Clock Cycles	@ 269 MHz
Key generation	1 304 742	4847 us
Encapsulation	142 238	528 us
Decapsulation	259 945	965 us

	Slices	LUT	FF	BRAM	DSP
Paper	1841	9528	7803	14	19
New	1596	8933	5221	13	19

Hamburg University of Technology

Agenda

- Introduction to Streamlined NTRU Prime
- Algorithm description
- Implementation overview
- Implementation details
- Conclusion & future work

Inversion during Key Generation

- Extended GCD algorithm [1] for inversion

$$\mathcal{R}/q = (\mathbb{Z}/q)[x]/(x^p - x - 1)$$

$$\mathcal{R}/3 = (\mathbb{Z}/3)[x]/(x^p - x - 1)$$

- Significantly faster than e.g. inversion using Fermat’s little theorem
- Key Gen cycle count is dominated by the two polynomials inversion (89.6%)

[1] Bernstein, Daniel J., and Bo-Yin Yang. "Fast constant-time gcd computation and modular inversion." IACR Transactions on Cryptographic Hardware and Embedded Systems (2019)

Multiplication

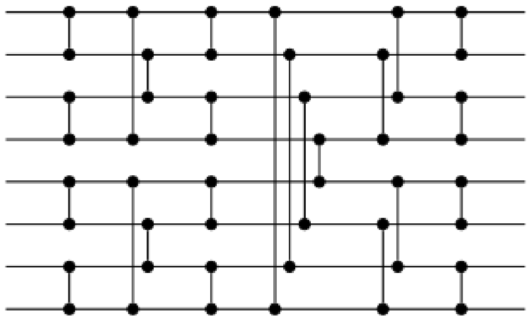
- Using NTT in NTRU Prime is not straightforward

- Mixture of Karatsuba and Schoolbook multiplication
- 78 132 cycles with 1 Karatsuba layer

- $\mathcal{R}/q * \mathcal{R}/3$: Coefficients are 13 bit and 2 bit (-1, 0, 1)
- 13 bit $*$ (-1, 0, 1) is essentially nothing
- Schoolbook multiplication is very resource light
- $\mathcal{R}/3 * \mathcal{R}/3$ is performed by same circuit, with a modulo 3 at the end

Generation short polynomials

- No sampling needed, instead:
- Take p 32-bit random integers
- Of the first w numbers, modify last 2 bit so they are always even
- Of the rest, set to odd
- Then sort with a constant time algorithm
  - Constant time with regards to input
  - Sorting network from [2]
- Only use last two bits, and subtract by 1
- 50 927 cycles



[2] Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. “NTRU Prime: reducing attack surface at low cost”. International Conference on Selected Areas in Cryptography

Encoding & Decoding

- Trivial in $\mathcal{R}/3$ → Simple shift register
- $\mathcal{R}/q$ encoder: bitshift, 16-bit addition, 16-bit multiplication
- $\mathcal{R}/q$ decoder requires 32-by-16 division with remainder
- But: divisors can be precalculated → 42 in total
- Replaced with division by constant → multiplication & bit shift
- Divisors are not secret dependent
- Division by 4591:

$$\mathrm{y} = x * 3831885438 \gg (13 + 31),\ 0 \leq x \leq 2^{32}$$
$$\mathrm{r} = x - \mathrm{y} * 4591$$

Hamburg University of Technology

SHA-512

- Hashing time is short (325 cycles per 1024-bit block)
- But: the SHA-512 used to be the single largest module
- Some optimization since

- Optimal sized SHA-512 implementation is critical
  - Bonus if your crypto coprocessor already has SHA-512

Constant Time Implementation

- All operations are constant time
- Sorting allows constant time generation of short polynomials
- Special care to:
  - Check if the polynomial r' has exactly w non-zero coefficients
  - Ciphertext equivalence check

- No side channels through decryption failures
- No further protections against advanced side channel attacks

Hamburg University of Technology

Comparison

- First full implementation
- [1] did not have key generation or decoding

	Slices	LUT	FF	BRAM	DSP	Clock	Encap	Decap
This	1 596	8 933	5 221	13	19	269 MHz	528 us	965 us
No key gen/ decoding	1 028	5 743	3 823	8	3	280.2 MHz	483 us	901 us
[1]	10 319	70 066	38 144	9	0	263 MHz	56.3 us	53.3 us

[1] Viet Ba Dang, Farnoud Farahmand, Michal Andrzejczak, Kamyar Mohajerani, Duc Tri Nguyen, and Kris Gaj. Implementation and Benchmarking of Round 2 Candidates in the NIST Post-Quantum Cryptography Standardization Process Using Hardware and Software/Hardware Co-design Approaches. Cryptology ePrint Archive, Report 2020/795. https://eprint.iacr.org/2020/795. 2020.

Agenda

- Introduction to Streamlined NTRU Prime
- Algorithm description
- Implementation overview
- Implementation details
- Conclusion & future work

TUHH
Hamburg University of Technology

Conclusion & Future work

- Lightweight implementation is possible
  - Suitable for embedded systems

- New round 3 parameter sets
- Optimal SHA-512 implementation
- Advanced side channel protections
- Suitability of NTT multiplication
- Batch inversion using Montgomery Trick

Hamburg University of Technology

TUHH
Hamburg University of Technology

NXP

SECURE CONNECTIONS
FOR A SMARTER WORLD