2nd NIST PQC Standardization Conference
22–24 August 2019, Santa Barbara, CA

A Lightweight Implementation of NTRUEncrypt for 8-bit AVR Microcontrollers

Johann Großschädl

(joint work with Hao Cheng, Peter Roenne, and Peter Ryan)

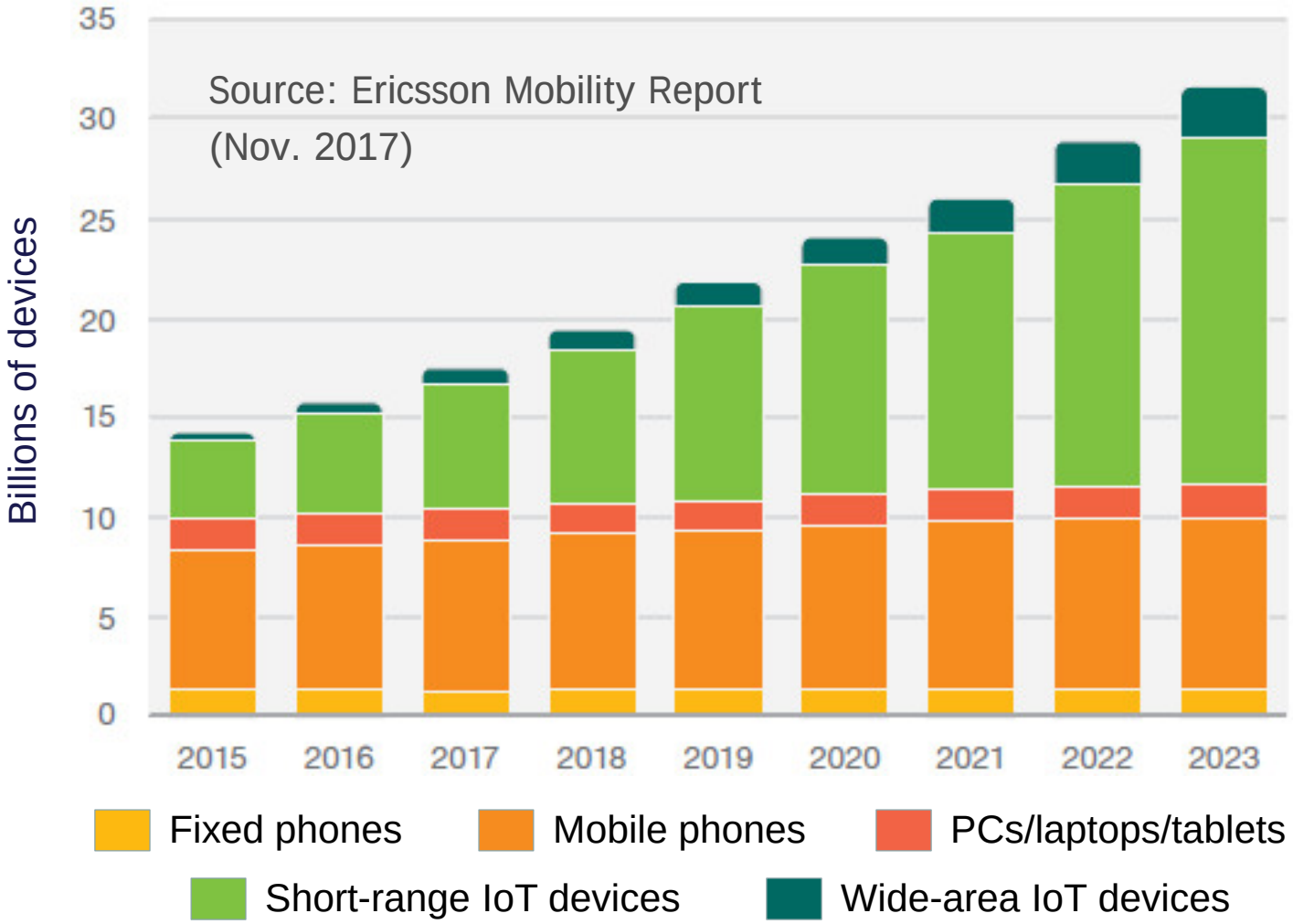
UNIVERSITÉ DU LUXEMBOURG

Laboratory of Algorithmics, Cryptology and Security (LACS)
University of Luxembourg
http://lacs.uni.lu/members/johann_groszschaedl

PQC for the IoT

- AVRNTRU: NTRUEncrypt for 8-bit AVR
  - Compliant with EESS #1 version 3.1 (Sept. 2015)
  - Supports product-form parameter sets with SHA256, e.g. **`ees443ep1`** (128b) and **`ees743ep1`** (256b)
  - Scalable: change parameter set w/o re-compilation
  - Resistance against timing attacks
- Polynomial Arithmetic
  - Optimized for products of sparse ternary polynomials
  - “Hybrid” multiplication of Gura et al. (CHES 2004)
- Auxiliary Functions (SHA256)

IoT Connections Outlook



Resource Constraints

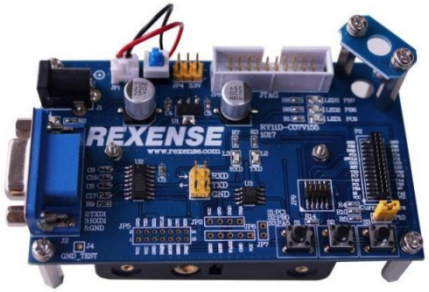
- RFC7228: Three Categories of IoT Devices

Category	RAM	Flash/ROM
Class 0 (C0)	‹‹ 10 kB	‹‹ 100 kB
Class 1 (C1)	~ 10 kB	~ 100 kB
Class 2 (C2)	~ 50 kB	~ 250 kB

- Examples of C1 Devices

Memsic Iris
8-bit AVR ATmega1281
8 kB RAM, 128 kB Flash

Zolertia Z1
16-bit MSP430F2716
10 kB RAM, 96 kB Flash



Rexense Def10
32-bit STM32W108 (M3)
12 kB RAM, 128 kB Flash

Ring Multiplication ("Convolution")

- Main Arithmetic Operation: Ring Multiplication
  - Truncated polynomial ring $R = Z_q[x]/(x^N-1)$
  - Typical instantiation (128b): $N = 443$, $q = 2^{11} = 2048$
- Polynomial Multiplication with Reduction
  - Operands of degree $N-1$, product has degree $2N-2$
  - Reduction modulo x^N-1 to get result of degree $N-1$
  - Reduction of coefficients modulo q
- Implementation Options
  - Operand scanning, product scanning: $O(N^2)$
  - Karatsuba, Toom-Cook, etc: $O(N \log N)$

Optimization

- Convolution $z(x) = u(x) \cdot v(x)$ in NTRU
  - $v(x)$ is ternary polynomial, i.e. $v_j = -1$, 0, or 1
  - Convolution: addition and subtraction of coefficients
  - Execution time depends on the number of non-0 coefficients of $v(x)$
- Product-Form Polynomials
  - $v(x) = v_1(x) \cdot v_2(x) + v_3(x)$
  - $v_1(x)$, $v_2(x)$, $v_3(x)$ can be sparse (i.e. have few non-0 coefficients) since coefficients cross-multiply
  - Extremely efficient: $O(N \log N)$
  - No security implications (at least in theory!)

Problem: Timing Attacks

> “The use of product-form parameter sets was originally intended to provide improved performance by allowing a specialized multiplication algorithm that used knowledge of the indices of the non-zero coefficients […]. However, this index-based multiplication proves to be ***very hard to implement in a constant-time fashion without losing the speed benefits***, so in this paper we concentrate on other approaches of multiplication.”

Wei Dai, William Whyte, and Zhenfei Zhang. **Optimizing Polynomial Convolution for NTRUEncrypt.** IEEE Transactions on Computers, vol. 67, no. 11, pp. 1572−1583, November 2018

Towards Timing-Attack Resistance

- Sources of Timing Leakage
  - Calculation of indices (i.e. pointer arithmetic) for accessing the coefficients u_i of polynomial $u(x)$
  - Data-dependent RAM accesses (cache hits/misses)
- Constant-Time Multiplication
  - Microcontrollers used in C1 devices have no cache!
  - Implementation of index calculation without any conditional statements (e.g. if-then-else)
- Fast Constant-Time Multiplication
  - Hybrid method processing 8 coefficients at a time

Sparse Ternary Multiplication (CT)

```
#define INTMASK(x) (~((x) - 1))

void mul_tern_sparse(uint16_t *r, const uint16_t *u, const uint16_t *v, int vlen, int N)
{
  int index[vlen], i, j, k;

  for (i = 0; i < vlen; i ++) index[i] = INTMASK(v[i] != 0) & (N - v[i]);

  for (i = 0; i < N; i += 8) {
    for (j = 0; j < vlen/2; j ++) {
      k = index[j];
      r[i  ] += u[k  ]; r[i+1] += u[k+1]; r[i+2] += u[k+2]; r[i+3] += u[k+3];
      r[i+4] += u[k+4]; r[i+5] += u[k+5]; r[i+6] += u[k+6]; r[i+7] += u[k+7];
      index[j] = k + 8 - (INTMASK(k + 8 >= N) & N);
    }
    for (j = vlen/2; j < vlen; j ++) {
      k = index[j];
      r[i  ] -= u[k  ]; r[i+1] -= u[k+1]; r[i+2] -= u[k+2]; r[i+3] -= u[k+3];
      r[i+4] -= u[k+4]; r[i+5] -= u[k+5]; r[i+6] -= u[k+6]; r[i+7] -= u[k+7];
      index[j] = k + 8 - (INTMASK(k + 8 >= N) & N);
    }
  }
}
```

Auxiliary Functions of NTRU

- Index Generation Function (IGF)
  - Generates indices for a sparse ternary polynomial
  - Calls internally a hash function (SHA-2)
- Blinding Poly Generation Method (BPGM)
  - Generates *r(x)* from a seed using IGF
- Mask Generation Function (MGF)
  - Generates a non-sparse ternary polynomial (mask)
  - Mask is added to message *m(x)*
- Efficient Implementation of SHA256
  - Compression function in ASM: 24k clock cycles

Timings on 8-bit ATmega1281

Operation	ees443 (128b)	ees743 (256b)
Poly-Arith	192,577	509,227
BPGM	308,801	492,940
MGF	189,525	293,138
Encryption	816,527	1,506,132
Decryption	1,022,093	2,037,124

- Ring mul only 23.6% − 33.8% of total encryption time
- Auxiliary functions (SHA-256) dominate execution time
- Code size: 8.9 kB (including two parameter sets)
- RAM footprint: 2.9 kB (128b encr) − 6.4 kB (256b decr)

Comparison

Reference	Algorithm	Sec.	Platform	Encryption	Decryption
This work	NTRUEnc	128b	ATmega1281	816,527	1,022,093
This work	NTRUEnc	256b	ATmega1281	1,506,132	2,037,124
Boorghany [9]	NTRUEnc	128b	ATmega64	1,390,713	2,008,678
Boorghany [9]	NTRUEnc	128b	ARM7TDMI	693,720	998,760
Guillen [20]	NTRUEnc	128b	ARM CortexM0	588,044	950,371
Guillen [20]	NTRUEnc	192b	ARM CortexM0	1,040,538	1,634,821
Guillen [20]	NTRUEnc	256b	ARM CortexM0	1,411,557	2,377,054
Gura [21]	RSA-1024	80b	ATmega128	3,440,00	87,920,000
Düll [17]	ECC-255	254b	ATmega2560	13,900,397	13,900,397
Liu [37]	RingLWE	106b	ATxmega128	796,872	215,031

Some Final Words

- Concluding Remarks
  - Product-form parameters are useful in practice!
  - NTRUEncrypt is suitable for Class-1 IoT devices
  - Main problem: high RAM consumption
- Ongoing Work: Masking for DPA Protection
  - Polynomial arithmetic is easy to mask
  - SHA256 is extremely costly to mask
- Future Work
  - NTRUEncrypt for MSP430 and ARM Cortex-M3
  - NTRUPrime and Three Bears

The End

Thanks for your Attention!

Questions?