# ThreeBears post-quantum KEM

Mike Hamburg

12 April 2018

Rambus
Data · Faster · Safer

# What’s different about ThreeBears?

Integer MLWE

- \+ Can leverage existing bignum implementations
- \+ Simple to specify and implement
- – New and untried
- – Everyone hates carry chains

BCH 2-error-correcting code

- \+ Improves security, efficiency
- \+ Simple, fast, constant time
- – Adds complexity vs no ECC

Carefully designed CCA security mode (based on Fujisaki-Okamoto)

- \+ Simple and efficient
- \+ Provably secure (QROM)
- – Explicit rejection leaves room for screwups

# Advantage: security

Conservative goal -> conservative parameters

Don't let porridge get stolen

Very high security levels: Q-core-sieve difficulty (ADPS 2015)

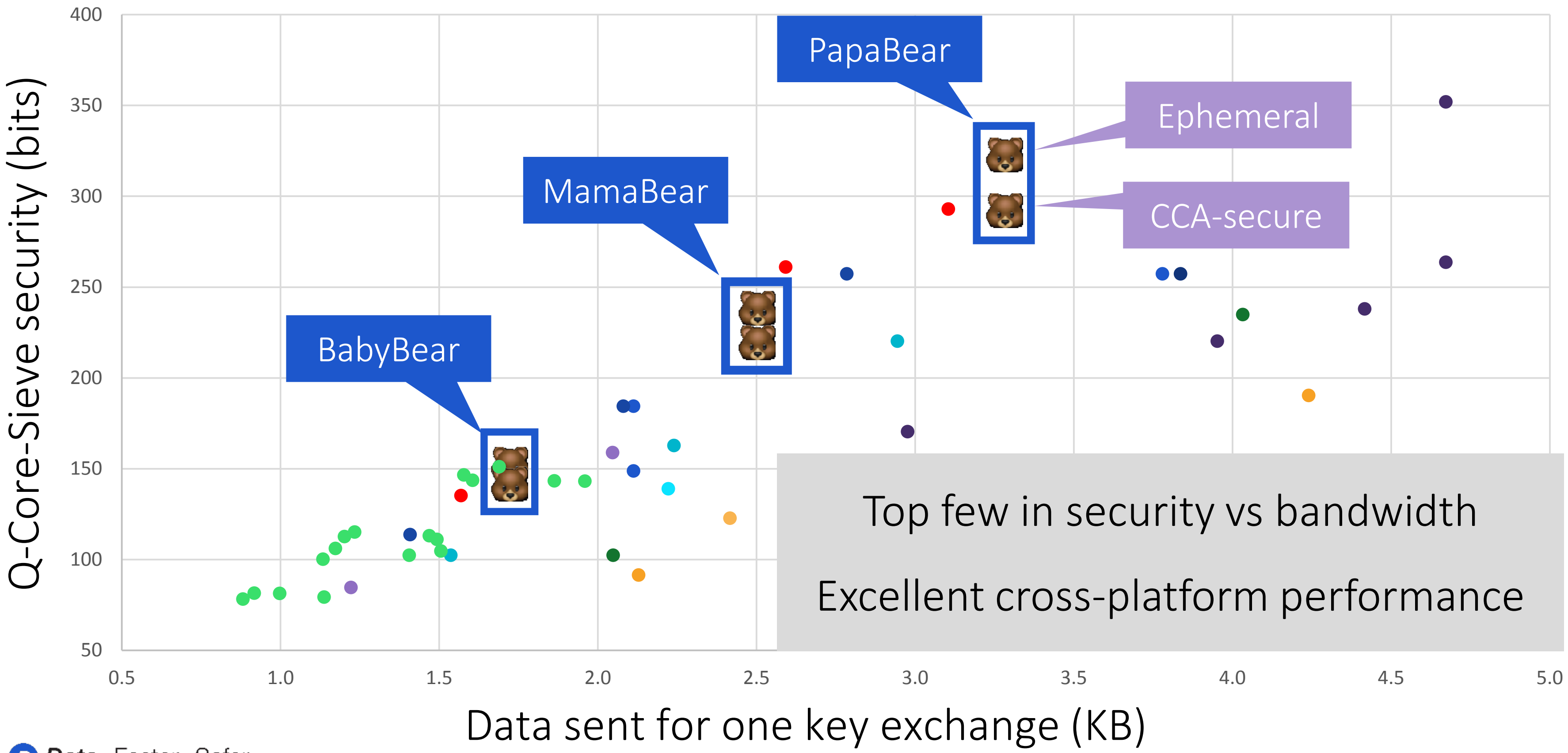
BabyBear: $2^{142}$ / $2^{152}$ MamaBear: $2^{220}$ / $2^{237}$ PapaBear: $2^{292}$ / $2^{320}$

Maybe overspecced?

Tiny failure probability to prevent CCA attacks

Enough noise to prevent lattice+MITM hybrid attacks

# Advantage: efficiency



# Advantage: simplicity

Complexity harms efficiency, security, trustworthiness

No magic constants, no special representations

Easy to optimize

- Designed for constant-time operation
- No vectorization required
- $\approx$1200 lines + 100 lines/instance optimized, cross-platform C
  - Includes comments and headers
  - Includes support for vectorized libkeccak if present

One compromise: BCH 2-error-correcting code

- +8% efficiency; more conservative
- < 100 lines constant-time optimized C, including header

# Details

Rambus
Data · Faster · Safer

# Integer module learning with errors

Polynomial MLWE: polynomials mod sparse low-weight polynomial $P(x)$

Lattice is spanned by powers of $x$

Reduce coefficients mod $q$

Integer MLWE: polynomials mod sparse low-weight polynomial $P(x)$

Lattice is spanned by powers of $x$

Evaluate at some particular $x$ to get an integer mod $N = P(x)$

ThreeBears: $x = 2^{10}, \quad N = P(x) = x^{312} - x^{312/2} - 1$

$N$ is prime, so no subrings

Fast bignum arithmetic: Karatsuba, Solinas

Easy to encode and decode, since $x$ is a power of 2

# Key exchange from MLWE à la LPR10

Private key:

Choose low-weight vector $\vec{a}, \vec{\epsilon} \in R^d$

Seed for random matrix $U \in R^{d \times d}$

Public key:

Seed for $U$; $A := U\vec{a} + \vec{\epsilon}$

Encrypt a message $m$:

Choose low-weight vectors $\vec{b}, \vec{\delta} \in R^d$

$B := \vec{b}^\top U + \vec{\delta}^\top$, high bits of $C := \vec{b}^\top A + \delta' + \text{encode}(m)$

Decrypt:

round and decode $C - B\vec{a}$

# Error correction

HILA5: XE5 5-error-correcting custom code

- Simple, but adds 240 bits to plaintext
- We only have 56 bits for redundancy

LAC: many-error-correcting BCH code

- Complex
- Hard to make constant-time

ThreeBears: 2-error-correcting BCH code (Melas variant)

- Adds 18 bits to plaintext = 9 bytes to ciphertext
- Small, fast, constant-time
- Adds ≈8% efficiency and reduces risk of hybrid attack

# Fujisaki-Okamoto transform for CCA version

Protected from multiple-target attacks

Hash the pubkey's matrix seed into everything

No key confirmation tag: it's not necessary

Explicit rejection: provably secure, in part because $N$ is prime

New post-quantum (QROM) security analysis

Probably still loose, but a big improvement!

$$\mathrm{Adv}_{\mathrm{CCA}} \leq O(q\sqrt{\mathrm{keyspace}} + q\sqrt{\mathrm{failure}} + \sqrt{q \cdot \mathrm{Adv}_{\mathrm{RLWE}}})$$

# Conclusion

ThreeBears is simple, conservative and efficient

- Worth studying even though I-MLWE is new
- Consider using its components as $2^{nd}$-round tweaks

Thanks for your attention!