Cryptographic Engineering

Challenges in Embedded and Hardware Cryptography

Markku-Juhani O. Saarinen
mjos@pqshield.com

PQShield Ltd. – Oxford, UK

Introduction to Cryptology
November 21, 2019 – Mathematical Institute, Oxford, UK

Talk Outline

I. **Introduction**

II. Symmetric Algorithms – Reasonably Stable

III. Asymmetric (Public Key) Algorithms – In Transition

IV. Engineering Case Study – PQC Energy [*skimming*]

V. Measuring NIST Post-Quantum Crypto [*skimming*]

VI. Final Notes

Cryptographic Engineering: My Day Job for 20 Years

PQ SHIELD

Cryptographic Engineering

1. **Specification:** Actual algorithms and protocols are not just "mathematical discoveries", but carefully engineered to meet a wide range of *requirements*.
2. **Architecture:** The entire system needs to be designed to be secure: User access control, Key lifecycle (generation, storage), target platform, etc.
3. **Implementation:** Robustness, (formal) correctness, mitigating side-channel leakage from timing and other channels, tamper-proofing hardware etc.
4. **Evolution:** Bugs are found, attacks improve, requirements change. Security is a continuous process and every change requires equivalent, careful attention.

Designing and applying cryptography for real-world products and services.

Requirements: Security

Security is #1 and usually depends on the **weakest link** of the entire system.

In addition to cryptanalytic strength ("bits") one may use "insurance" risk metrics:

$$\text{Risk} = \sum_{\text{Attacks}} Pr(\text{Attack success}) \times \text{Resulting loss £}.$$

The adversary will use the **easiest** attack – typically the endpoint / implementation. Security can be measured by estimating the **minimum cost** of a successful attack.

Algorithms and "key bits" are free but high-quality implementation work requires skill, effort, and money. However, security investments are intended to *save money*.

Consider remediation costs, product recall, reputation, etc. Despite costs it it is usually cheaper to engineer good cryptography to products than bad – or none.

Implementation Vulnerabilities: Some Terminology

Logical error is a failure in failure in program flow or other logic. Often caused by a failure to correctly validate input (checking for errors and special conditions).

Side-channel attack derives a secret information from implementation channels such as timing variance (most targets), electromagnetic emissions (esp. RF devices such as phones), or fluctuations in power use (esp. smart cards).

Timing attack is a leak of confidential information via timing – often based on cache access patterns. Therefore "constant time" execution is a cryptographic design goal.

Static analysis is an automated method for examining program logic directly from source code (without execution). Data-flow analysis (secret $\rightarrow$ non-secret) and abstract model checking (code = model) most relevant in cryptography.

Fuzzing is an automated method for rapidly trying out a large number of inputs to a function and observing behavior (flow, crashes, memory leaks, and other failures).

Requirements: Software Implementation Targets

	Unit Price	Representative CPU / MCU	Cores / Threads	Example applications
Interoperability! ↕	£1	Cortex M0	1 SoC	Toaster, lightbulb, cheap toy.
	£3	M4, RV32	1 MCU	Home Appliance, Smart Card.
	£10	ARMv7 (32 bit)	1 CPU	Automotive, Security Element.
	£30	ARMv8 (64 bit)	4	Low-end phone / Raspberry Pi.
	£100	Celeron N3060	2/2	Consumer desktop or laptop.
	£300	CORE i7-8700	6/12	Professional desktop or laptop.
	£1000	CORE i9-7940X	14/28	High-end gamer, designer.
	£10000	XEON 8180	28/56	Amazon AWS Data Center.

“Internet of Things” – interoperable cryptography everywhere.

Implementations: Basic Efficiency Metrics

- **Cycle counts.** For public key operation (encrypt / verify), private key operation (decrypt / sign). These are dependant on the capabilities of CPU/CPU.
- **Throughput.** Cycles per byte for symmetric encryption, decryption, keyed authentication, and hashing. Parallelism helps (e.g. SIMD vector instructions).
- **Footprint.** For MCUs, implementation footprint (bytes in Flash) and stack / RAM usage in bytes. Typically restricted to few kB (S)RAM and 10s kB Flash.
- **Circuit Area** in terms of Gate Equivalents (GE). Gates do not have unit cost: NOT $\approx 1/2$ GE. NAND, NOR ≈ 1 GE. AND, OR $\approx 1\ 1/3$ GE. XOR $\approx 2\ 2/3$ GE.
- **Power** is the product of voltage and current: $P_{(\text{Watts},W)} = V_{(\text{Volts},V)} \times I_{(\text{Amps},A)}$.

  For many (CMOS) circuits the current draw is almost **linearly dependent** upon the clock frequency and circuit area for a fixed voltage. Trade-offs are possible.

Hardware: Chip & Pin cards, Car Keys, RFID tokens, etc.

PQ SHIELD

- **Physically secure and uncloneable.** Cards, keys, and SIMs protect their secret keys even against attacks by sophisticated adversaries in a laboratory setting.
- **Fast.** Time for key exchange or authentication is limited – from tens or hundreds of μs in RFID up to about 200 ms for smooth end-user experience.
- **Power efficient.** Proximity-based passive RFID authentication tokens are powered by backscatter radiation from the reader. Battery powered authentication devices must run for years without a charge.
- **Cheap to manufacture.** It's a big, highly competitive market (few £ per unit).

A cryptographic algorithm must fit all of these software and hardware targets and simultaneously satisfy all of the security, efficiency, price requirements.

Talk Outline

I Introduction

II **Symmetric Algorithms – Reasonably Stable**

III Asymmetric (Public Key) Algorithms – In Transition

IV Engineering Case Study – PQC Energy [*skimming*]

V Measuring NIST Post-Quantum Crypto [*skimming*]

VI Final Notes

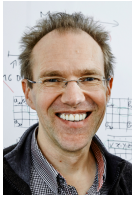
Symmetric Encryption

Advanced Encryption Standard (AES, FIPS 197)

- Secure against all known forms of *mathematical cryptanalysis*, recommended for use **everywhere**.
- Result of a competition to replace an older standard.
  - DES (Data Encryption Standard) – Lucifer **1975**.
  - AES (Advanced Encryption Standard) – Rijndael **1998**.
- Child of 1990's: Designed for **32-bit** arch ("T-table").
- AES has 8-bit S-Boxes, difficult to implement in secure way (no table look-ups) without **AES-NI** instructions; `AESENC` (full AES round) in CPUs.
- Some TLS implementations switch from **AES-GCM** AEAD to **ChaCha20-Poly1305** AEAD if there is no hardware support for AES and GCM – for security.

Fresh PhDs in 90s:

Now Professors Daemen and Rijmen



The AES Timing Problem

Final round AES Proposal, September 1999

Joan Daemen and Vincent Rijmen: **Rijndael.**

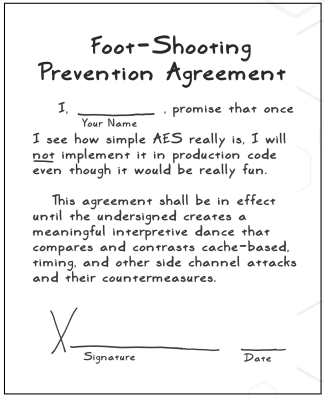
"To prevent timing attacks, attention must be paid that xtime is implemented to take a fixed number of cycles [..] in practice this can be achieved by using a dedicated table-lookup."

· · · few years pass · · ·

After standardization, April 2005

Daniel J. Bernstein: **Cache-timing attacks on AES**

"This paper demonstrates complete AES key recovery from known-plaintext timings of a network server on another computer. This attack should be blamed on the AES design.."



Don't implement AES if you're not 100% sure you know how..

The agreement: `http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html`

Specially Designed Lightweight Ciphers

PQ SHIELD

NSA: Simon (and Speck)

Designed by a group within NSA.
`https://eprint.iacr.org/2013/404`

- → Detailed design methodology is classified. Hence rejected from ISO.
- → However these algorithms have withstood public cryptanalysis.
- → Hardware profile is significantly smaller and more secure than AES.

NSA allows use for securing classified:

"Simon 128/256 and Speck 128/256 have been deemed to provide the security necessary for National Security Systems."

Comparison

"GIFT: A Small Present." CHES 2017
`https://eprint.iacr.org/2017/622`.

128/128	Gift	Simon	AES
Area (GE)	1997	2064	7215
Delay (ns)	1.85	1.87	3.83
Cycles	41	69	11
TP (Gbit/s)	1.730	1.007	3.038
Power (μW)	116.6	105.6	730.3
Energy (pJ)	478.1	728.6	803.3

Notes: Synthesized with STM 90nm Standard cell library. The throughput is absolute peak. Power consumption @ 10 Mhz.

Data from Thomas Peyrin (NTU, Singapore), 2018.

Summary: Symmetric Cryptography

PQ SHIELD

- **Symmetric cryptography has well understood cryptanalytic security criteria.** Reject any algorithm whose security < key brute force or hash birthday attack.
- **When in doubt, use AES**. Don't implement AES yourself. It's difficult to do right and it is already implemented in the processor and standard libraries.
- **Use Authenticated Encryption** modes (AEADs) such as AES-GCM for transport. There are special modes such as AES-XTX for data storage.
- **For hashing use SHA-2 or SHA-3.** SHA algorithms fortunately don't have timing problems but a library implementation is probably better for speed.
- **Stream ciphers.** AES-CTR is a stream cipher; GCM just adds authentication to it. ChaCha20 is popular semi-standard (RFC 7539) if no AES-NI.
- **Lightweight ciphers** are evolving as the NIST standardization work is ongoing. `https://csrc.nist.gov/projects/lightweight-cryptography`
  NSA's SIMON/SPECK are reasonable but can be "politically" bad for some.

Talk Outline

I Introduction

II Symmetric Algorithms – Reasonably Stable

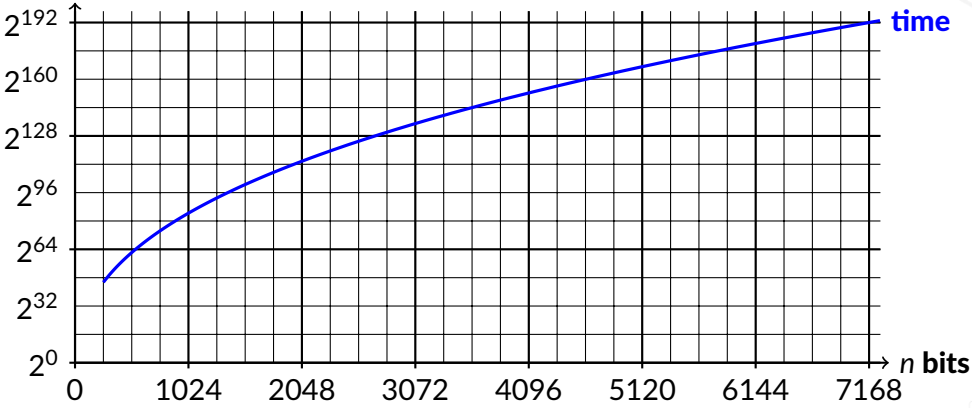
III **Asymmetric (Public Key) Algorithms – In Transition**

IV Engineering Case Study – PQC Energy [*skimming*]

V Measuring NIST Post-Quantum Crypto [*skimming*]

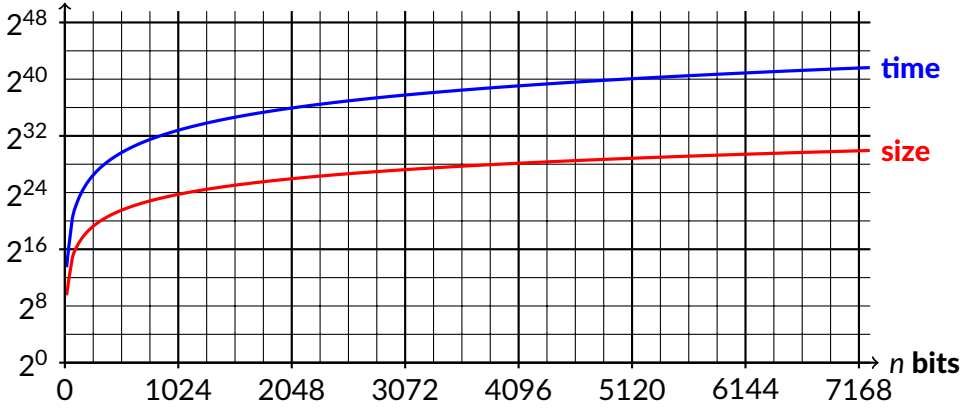
VI Final Notes

Best Known "Classical" Factoring Algorithm



General Number Field Sieve has $O\left(\exp\left(\sqrt[3]{\frac{64}{9}n(\log n)^2}\right)\right)$ complexity. This curve matches NIST's RSA key size recommendations (SP 800-57 Pt. 1 Rev. 4, Jan 2016).

Factoring with a Quantum Computer



Shor's quantum factoring algorithm requires $O(n^3 \log n)$ time with a quantum circuit of $O(n^2 \log n \log\log n)$ gates. It therefore has polynomial complexity.

Post-Quantum Cryptography: Background and Terminology

Quantum Computing (QC) uses **quantum superpositions**, rather than binary digits, to perform computations. This computational model was first considered in 1980s.

Quantum Algorithms are algorithms for Quantum Computers. They often have **different performance asymptotics from classical algorithms.**

Shor's Algorithm (1994) can factor integers and compute discrete logarithms efficiently (polytime). It also applies to the the Elliptic Curve DL Problem. Shor's algorithms can be **devastating to most of current public key cryptography.**

Grover's Search Algorithm (1996) can be used to search for a k-bit secret key with $\sqrt{2^k} = 2^{k/2}$ quantum effort. It effectively **doubles the required key sizes for ciphers.** However, symmetric crypto is much safer against QC than RSA, (EC)DL.

Post-Quantum or "**Quantum Resistant**" cryptography consists of algorithms that run efficiently on classical computers but are hard to break with QC.

August 2015: Official Post-Quantum Transition Starts

In August 2015 the Committee on National Security Systems (CNSS) and National Security Agency (NSA) **suddenly revised** their cryptographic recommendations in CNSSAM 02-15.

"Based on analysis of the effect of quantum computing [..] the set of authorized algorithms is [changed] as we anticipate **a need to shift to quantum-resistant cryptography** *in the near future."*

The recommendation also killed off shorter key lengths (AES-128, SHA-256, RSA-2048, DL-2048, ECC P-256) allowed in "**Suite B**".

The interim set of algorithms is called **"Commercial National Security Algorithm Suite"** and is approved up to **TOP SECRET**:

RSA 3072, DH 3072, ECDH/DSA P-384, SHA-384, AES-256.

(**Only** these algorithms, only these key sizes are in **CNSA**)

NIST Post-Quantum Cryptography Project

A NIST-run project to develop quantum-resistant **Public Key Encryption** and **Digital Signature** standards for to replace RSA and ECDSA.

- → **20.12.2016**: Call for proposals released.
- → **30.11.2017**: Candidate submission deadline.
- → **21.12.2017**: ~~**82**~~ *69* submissions accepted.
- → **11-13.04.2018**: First standardization conf.
- → **30.01.2019**: **26** semifinalists announced.
- → **22-24.08.2019**: Second standardization conf.

  – – – *We are here. Evaluation is ongoing.* – – –
- → **2020 / 2021**: Round 3 begins.
- → **2022 / 2024**: Draft standards available.

Classification of Candidates

There are 17 public-key encryption key establishment algorithms (replacement for ECDH and RSA), and 9 signature algorithms (replacement for ECDSA and RSA).

- **Lattice based: 12 (9 KEM + 3 Sig.)** Based on problems analogous to quantum shortest vector and other lattice problems, known to be hard. Fastest.
- **Code-Based: 7 (KEM).** These algorithms are based on coding theory problems. McEliece (1978) is in this set. Also QC-MDPC, QCSD, QC-LDPC, LRPC, IRSD ..
- **Multi-Variate: 4 (Sig).** Based on systems of equations, previously seen as rather *ad hoc*. Some interesting signature algorithms.
- **Symmetric-based: 2 (Sig).** There are two hash based signature proposals in the competition, but some additional ones are already being standardized (XMSS, LMH). Often seen as having best security assurances for signatures.
- **Isogenies: 1 KEM.** SIKE is based on isogeny problem of supersingular curves. ECC people love Isogeny systems because they use elliptic curves. Slowest.

Engineering Assumptions

November 2019: *The NIST PQC submission deadline was two years ago, and we're roughly halfway through the project. Standards will be among these 26 algorithms.*

Assumption 1: Little impact on symmetric cryptography

- Grover: Most bulk data transfer still with AEADs (e.g. AES-GCM) and stream ciphers (e.g. ZUC). PQC Impacts mainly handshake and authentication.

Assumption 2: No fundamental protocol re-engineering

- IETF and ETSI have been sitting on the fence, waiting for NIST to finish.
- Quantum-secure signature and KEM algorithms can use equivalent external APIs to current standard ECDSA, ECDH, RSA cryptography.
- Drop-in replacement to most current applications and protocols.

Quantification of Post-Quantum Security

Five categories of NIST: "Any attack [...] must require computational resources comparable to or greater than those required for ...

1. ... key search on a block cipher with a 128-bit key (e.g. AES-128)"
2. ... collision search on a 256-bit hash function (e.g. SHA-256 / SHA3-256)"
3. ... key search on a block cipher with a 192-bit key (e.g. AES-192)"
4. ... collision search on a 384-bit hash function (e.g. SHA-384/ SHA3-384)"
5. ... key search on a block cipher with a 256-bit key (e.g. AES-256)"

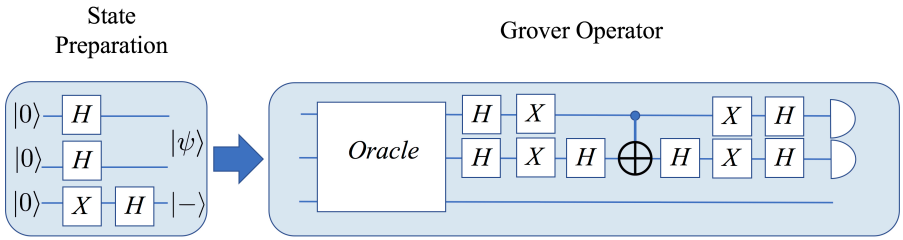
NIST suggests considering attacks "**limited by MAXDEPTH**" (analogous to constant runtime of attack $< 2^{96}$) and **estimating the number of quantum gates**.

Which gates ? What is MAXDEPTH ? What is "time" ? What about reliability ?

So how secure is AES-256? (Against Quantum Computing)

Grover's complexity is $O(2^{\frac{n}{2}})$ so rule of thumb estimates "128 quantum bit security".

The cipher itself needs to implemented as an oracle; NIST states an estimate of 2^{298}/MAXDEPTH quantum gates for 2^{272} classical gate operations.



Note! *The best attack technique against [PQC] may depend on completely different features of a quantum computer than the best quantum circuit for breaking AES.*

(Quantum) Complexity Estimation

In most cases the method is exactly the same as with RSA and ECDL:

1. Find the best classical and best quantum methods to break X.
2. Estimate their computational requirements in different (quantum) cost models.

Example: The Learning With Errors (LWE) problem was invented by **Oded Regev** [Re05,Re09] who showed its connection to worst case shortest vector problems in a quantum setting. These ideas were extended to **ring setting** (RLWE) with [LPR10]. The connection between a uniform secret **s** and a secret chosen from χ is provided by Applebaum et al. [ACP+09] for LWE case, and for the ring setting in [LPR13].

(R)LWE and NTRU parameter selection is typically based on estimating the complexity of the **Schnorr-Euchner BKZ** [SE94] algorithm in classical and quantum setting under a number of different cost models. I use Albrecht's estimation scripts [ACD+18]: `https://estimate-all-the-lwe-ntru-schemes.github.io/docs/`

Often breaking the "hard problem" isn't the easiest way of breaking a scheme!

Example [Sa17]: The BLISS signature algorithm [DDLL13] is based on the Ring-SIS – with appropriate lattice parameters. (**Note**: Not proposed as Post-Quantum!)

The signature consists of various pieces of data, including ring values. However, examining the signature verification algorithm, we observe that after some trivially forgeable checks, a signature is considered valid when an "verification oracle" agrees with a verification vector consisting of κ out of n elements.

Attack: An exhaustive search on $\binom{n}{\kappa}$ signature component values with Grover's algorithm requires $O(\sqrt{\binom{n}{\kappa}})$ time. This is **faster than breaking the lattice problem**.

Summary: Asymmetric Cryptography is in Transition

PQ SHIELD

- **RSA:** From 1990s to 2015 public-key cryptography standards were stable and RSA dominated both digital signatures and public key encryption. Key sizes were regularly bumped upwards up to current requirement of 2048/3072.
- **ECC:** From 2005 onwards ECDH was used ephemeral (forward secure) key exchange and ECDSA (with various standard curves) for digital signatures.
- **PQC**: In 2015 National security authorities (NSA and NCSA/GCHQ) determined in that it was necessary to start a post-quantum transition and phase out RSA and ECDSA/ECDH. NIST started a standardization process for Post-Quantum Cryptography that will finish up in early 2020s.

Prediction: Post-quantum transition challenges will dominate public-key cryptography engineering until end of 2020s.

Talk Outline

I Introduction

II Symmetric Algorithms – Reasonably Stable

III Asymmetric (Public Key) Algorithms – In Transition

IV Engineering Case Study – PQC Energy [*skimming*]

V Measuring NIST Post-Quantum Crypto [*skimming*]

VI Final Notes

Engineering Case Study: PQC Energy

Since this is just a single talk, I can't cover hugely important and complex topics like **protocol implementation techniques** and **EM/DPA attacks and countermeasures**.

However I'd like to talk about a current issue that (sometimes) forces real-world cryptographers to have their feet firmly on the ground: **actual power consumption**.

- **Battery life** of mobile devices and IoT sensors during the **RSA/ECDL** ⇒ **PQC** transition / standardization process has raised questions from the industry.
- **Example**: How many times a key fob can be used without changing batteries? This is pretty important to the makers of key fobs (authentication tokens) !
- Academic cryptographers: "It's polytime whatever ¯_(ツ)_/¯"
- What was problematic is an apples/oranges thing caused by the fact that the slowest KEM proposal also has the shortest messages: *"Transmitting by radio is so much more expensive than computation that you actually save power!"*

Quick recap: Power and Energy

These physical measures are surprisingly often confused..

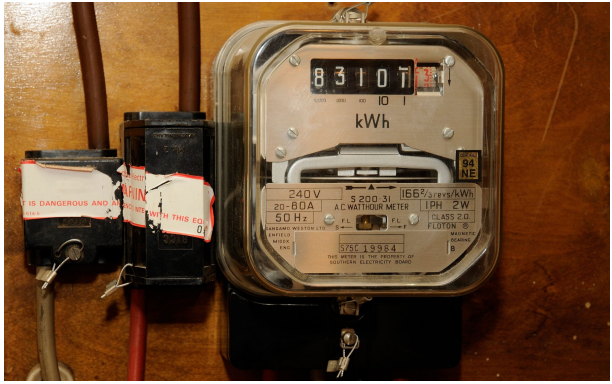
Electrical Power and Energy

$P = V \times I$ Power (W: Watt) = Voltage (V: Volt) $\times$ Current (A: Ampere)
$E = P \times t$ Energy (J: Joule) = Power (W: Watt) $\times$ Time (s: Second)

Lovely older units: Calorie (1 cal = 4.184 J), horsepower (1 hp = 764 W), etc.

- **Power** is momentary, **energy** is cumulative (*think velocity vs. distance*).
- To measure energy (J) we integrate (or "sum") power (W) over time.
- Voltage (V) is usually a known, regulated value (such as 3 V). We can use an *ammeter* to measure the current (Amps). Power is the product.

Common Derived Units



An older energy integrator (meter) using kWh = 3.6 MJ.

3.85 V × 3 Ah × 3600 s/h = 41.6 kJ.

- Derived units: e.g. **kWh** (kilowatt hour) = 1000 W × 3600 s/h = 3.6 MJ.
- Batteries are often specified in **mAh** (milliampere hour). One needs to know the voltage to compute the actual energy that the battery has.

PQC Computations: Dynamic power

From integrated circuit theory:

Dynamic Power Equation

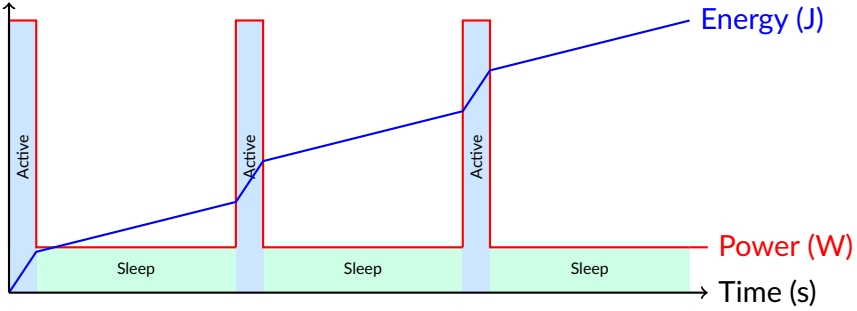
$$P_{\text{dyn}} = \alpha \cdot C \cdot V^2 \cdot f$$

P = Power, α = activity, C = Capacitance, V = Voltage, f = Frequency.

- **Dynamic power** dissipation P_{dyn} is caused by activity in the circuit.
- P_{dyn} is generally linear to frequency and area, quadratic to voltage.
- **Activity** α is sometimes called "**switching factor**" as the energy is consumed when the circuit transitions from one state to another.
- The α of a processor can vary a lot, *depending on what it is doing*.
- **Static power** dissipation P_{stat} when the circuit is idle. $P = P_{\text{stat}} + P_{\text{dyn}}$.

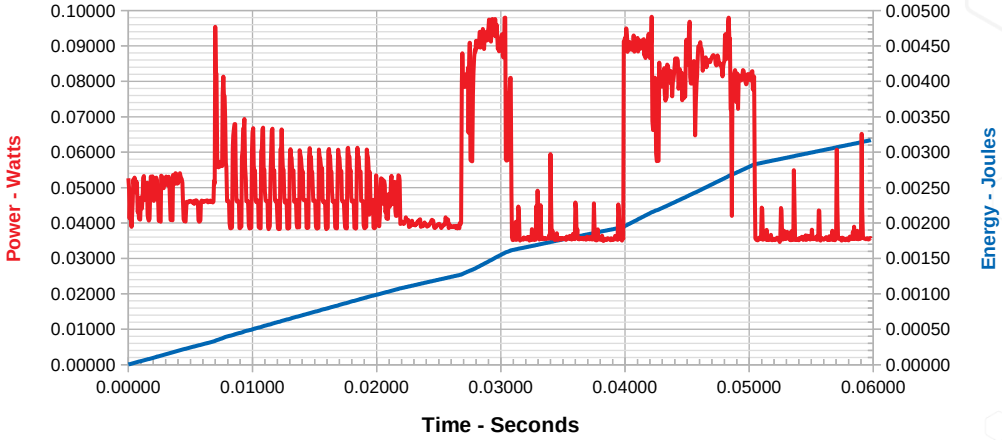
Sleep States and Power Management

- Most CPUs have one or more **sleep states**, also affecting peripherals.
- When asleep, instructions are not executed: P_{dyn} is very low.
- MCUs typically wake up only via an **interrupt** (timer or external event).
- Modern CPUs can also control ("scale") their clock frequency f (and V).



A typical power consumption model for IoT microcontrollers ("sleepy edge node").

Reality: 60 ms of NTRU-HPS 4096-821



- This is what "**active**" can look like on a real MCU (ARM Cortex M4).
- Power is rapidly fluctuating between 35mW and 100mW (3× range).
- Cycle count is clearly **not** telling the full story about this algorithm.

Talk Outline

I Introduction

II Symmetric Algorithms – Reasonably Stable

III Asymmetric (Public Key) Algorithms – In Transition

IV Engineering Case Study – PQC Energy [*skimming*]

V **Measuring NIST Post-Quantum Crypto [*skimming*]**

VI Final Notes

Cortex M4: Post-Quantum Power Sandwich

PQSHIELD

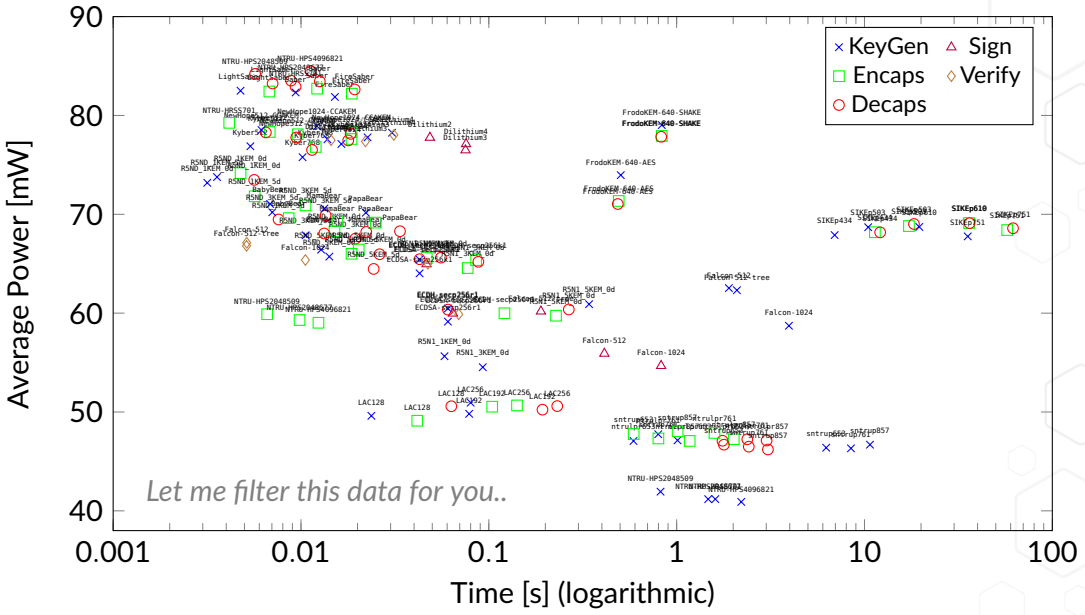
New PQC "IoT" Energy Measurements

- LPM01A "PowerShield" £50 power measurement board is also used for the EEMBC IoTConnect™ benchmarks.
- STM32F411RE target has a Cortex M4 core, the reference embedded platform of the NIST PQC project.
- I measured PQC implementations from the PQM4 project, also Ken MacKay's "micro-ecc" ECDSA & ECDH code.
- **Goal:** *Precise, independently repeatable.*

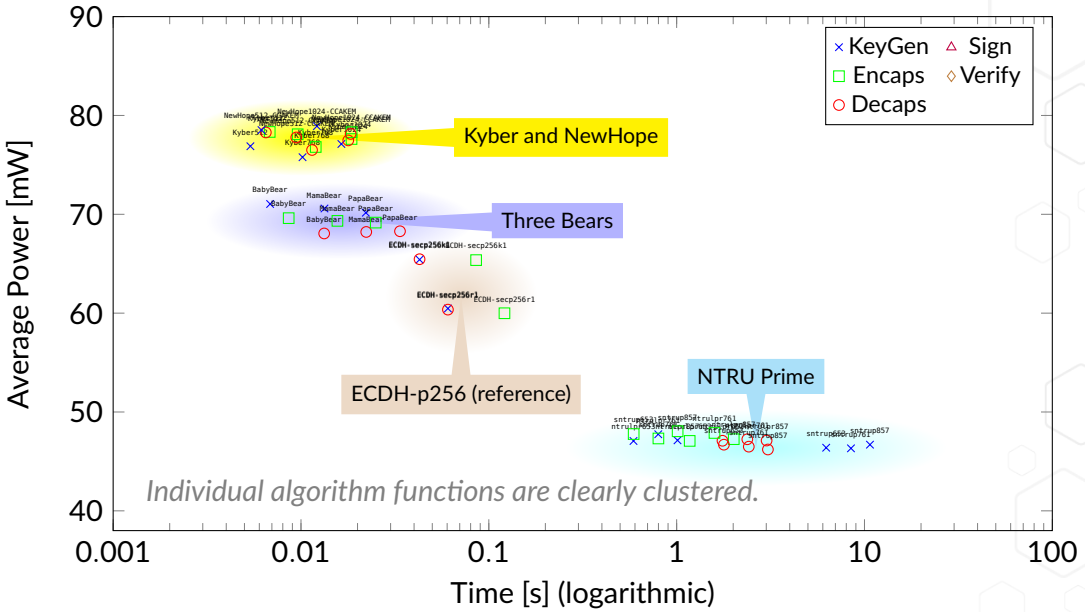
Source code and a description of the lab:
`https://github.com/mjosaarinen/pqps`

It's a dev board sandwich: LPM01A sits on top of the Nucleo64 target.

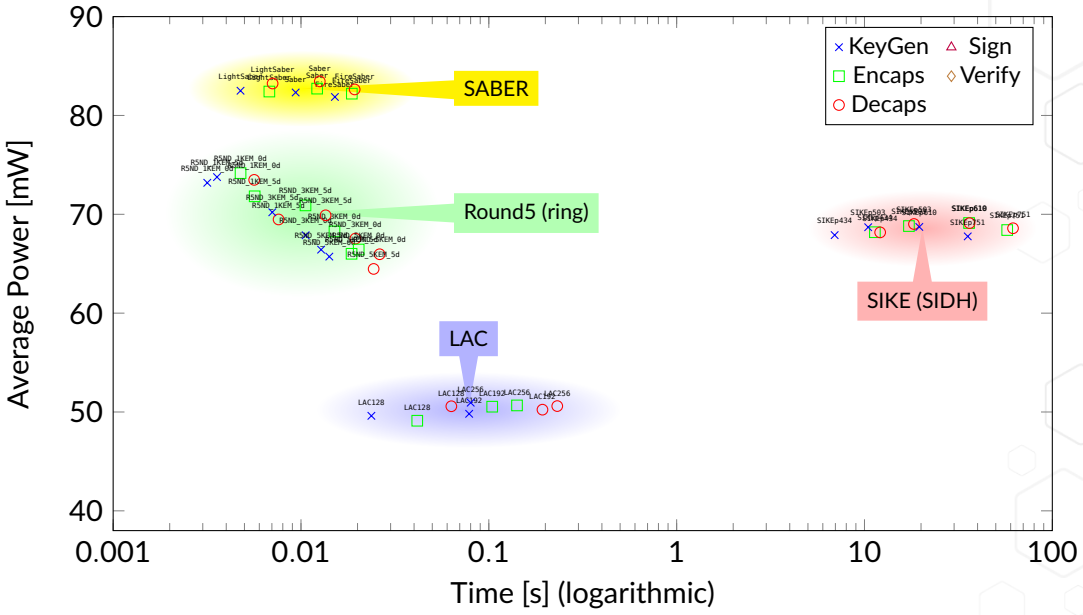
STM32F411RE: Average Power @ 96 MHz



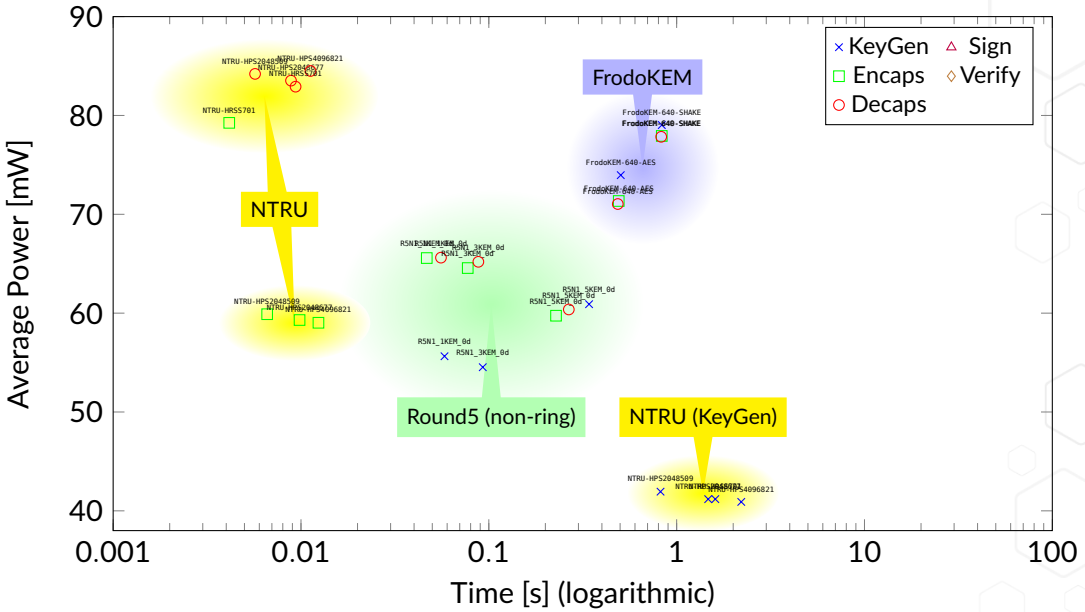
Cortex M4: Distinctive KEM Clusters (1/3)



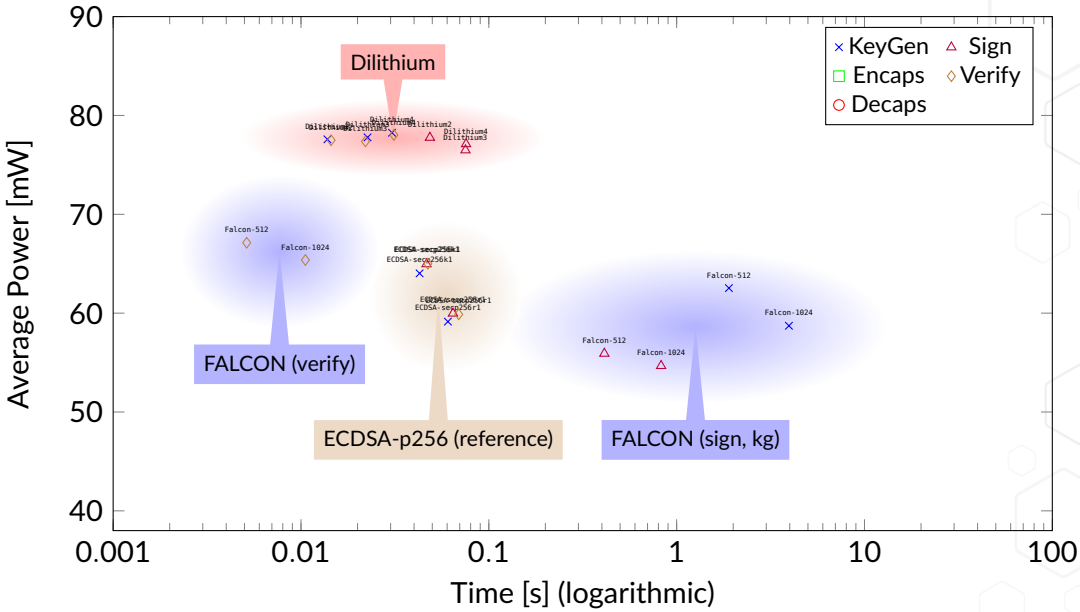
Cortex M4: Distinctive KEM Clusters (2/3)



Cortex M4: Distinctive KEM Clusters (3/3)



Cortex M4: Some Signature Algorithms



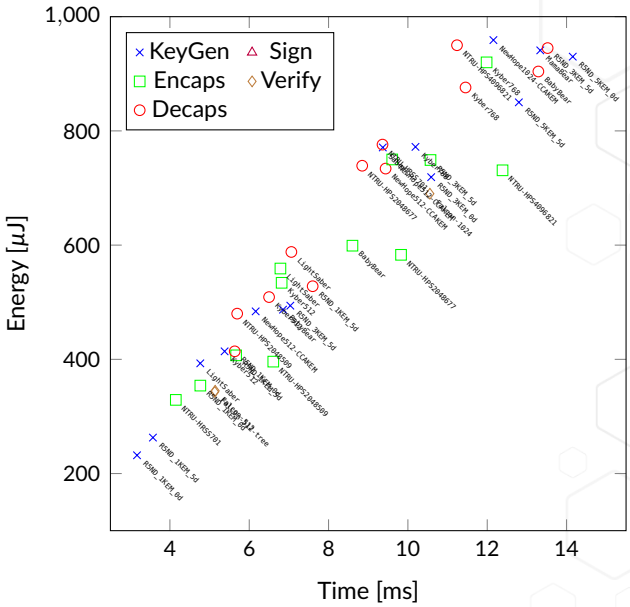
Cortex M4: Energy vs Time – Sub-mJ (μJ) Range

In "microjoule" range there are cases where algorithm's timing rank is different from energy rank.

Meaningful, but:

Are there general techniques to trade power for time?

Do the very distinctive power profiles translate to other microcontroller targets?

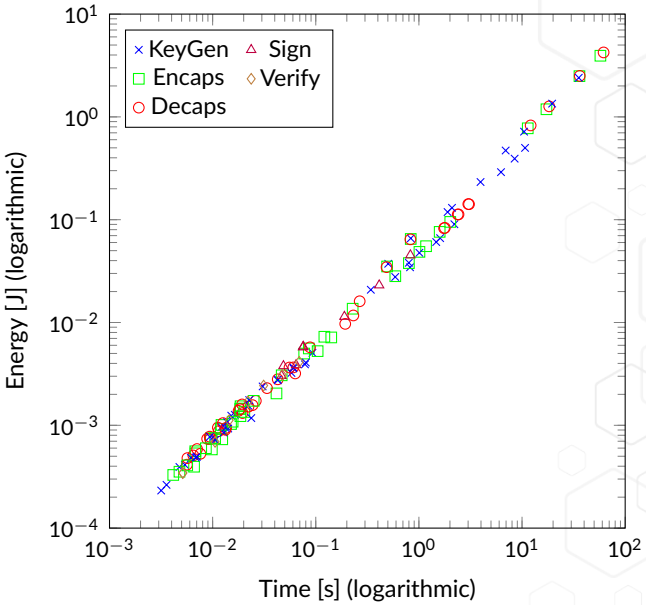


Cortex M4: Energy vs Time – Whole Range

There is a range of over **four orders of magnitude** in the complexity of PQC algorithms.

This completely dwarfs the observed ≈ 50% range in power. So “cycle counts” can be used to estimate energy, but results have **less than one digit of precision.**

A log-log plot of the NIST PQC 2nd round set looks quite linear.



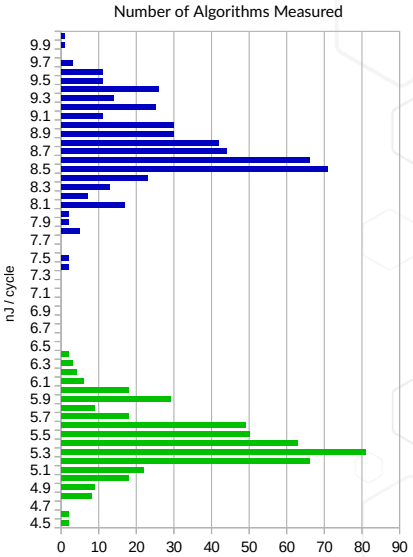
Intel PCs: RAPL (Running Average Power Limit)

Intel PC/Server Measurements

- Inspired by [1], I modified the "official" SUPERCOP benchmarking system to record energy usage via Intel's RAPL.
- Profiled 159 variants of about 20 NIST PQC algorithms in the benchmark.
- Power is **highly dependent on target**, but within that target **not as varied** as with IoT MCUs. Platform $[nJ/\text{cycle}]$ and cycle count leads to a good estimate.

`https://github.com/mjosaarinen/pqps/tree/master/suppercop`

[1] C. A. Roma, C. A. Tai, and M. A. Hasan: *"Energy Consumption of Round 2 Submissions for NIST PQC Standards"*, NIST PQC 2019.



Blue: *i7-8700 @ 4.6 GHz (Desktop)*
Green: *i5-8250U @ 3.4 GHz (Laptop)*

A *very* Simplistic Model for Wireless

All PQC candidate algorithms have larger key- and message sizes than current RSA and Elliptic Curve cryptography. **How much is too much?**

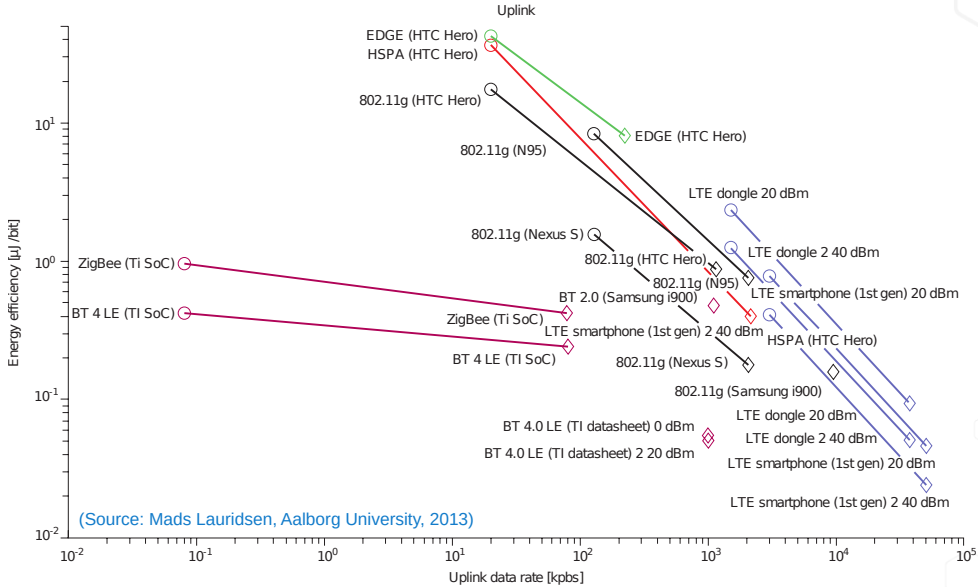
Total Energy: Compute it + Transmit it

E_{KG}	+	$e_{tx}\|\text{pubkey}\|$	Key generation.
E_{Enc}	+	$e_{tx}\|\text{ciphertext}\|$	Encapsulation.
E_{Sign}	+	$e_{tx}\|\text{signature}\|$	Authentication.

.. or whatever is relevant in the protocol in question.

- Uplink (transmit) energy $e_{tx} \ggg e_{rx}$ downlink (receive), both in $^{\text{Joule}}/_{\text{bit}}$.
- **Two** algorithm factors (complexity, message sizes) and **two** platform factors (computational efficiency and communication efficiency).
- We need **measurement data** to determine their relative importance.

Communication efficiency $e_{tx} \approx 0.1^{\mu J}/_{bit}$ (pre 5G)

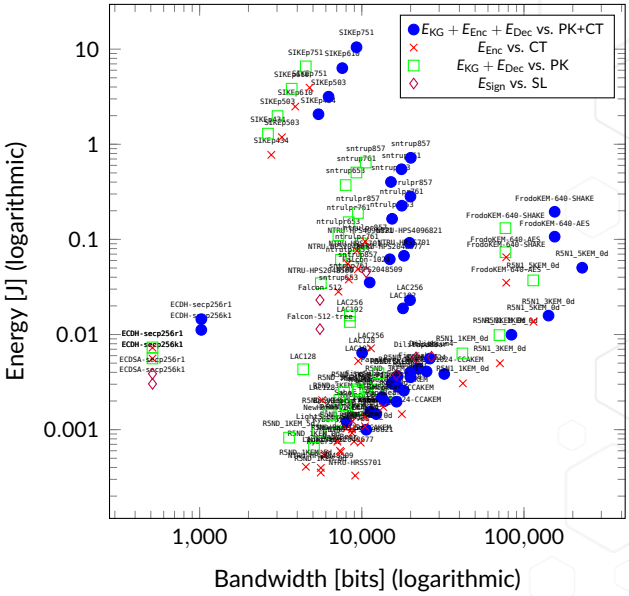


Energy and Message Lengths: “It Depends”

KEMs: Need to relate KG, Enc, Dec computation energy to transmit energy of Public Key and Ciphertext (PK, CT).

For *ephemeral key exchange* we may consider total energy $E_{\mathrm{KG}} + E_{\mathrm{Enc}} + E_{\mathrm{Dec}} + e_{\mathrm{tx}}(\mathrm{PK} + \mathrm{CT})$.

For signer (auth): Consider $E_{\mathrm{Sign}} + e_{\mathrm{tx}} \cdot \mathrm{SL}$, where SL is signature length. **Verifier** E_{Ver} also downloads (e_{rx}) some certificates which have public keys and signatures.. depends.

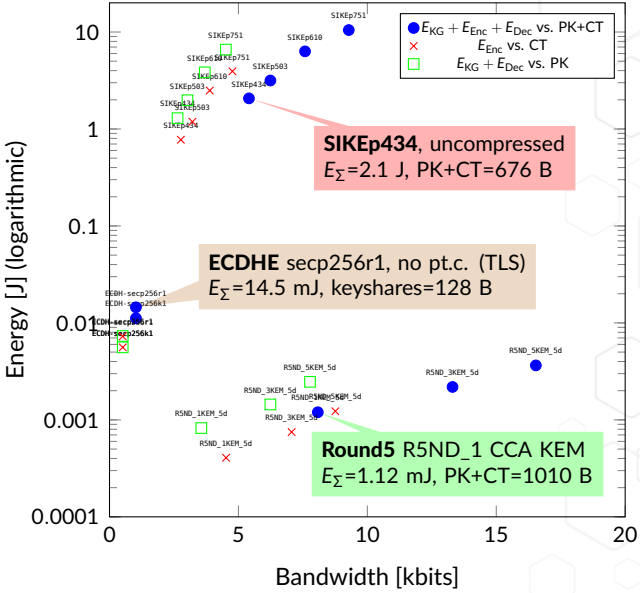


Example: Ephemeral Key Exchange

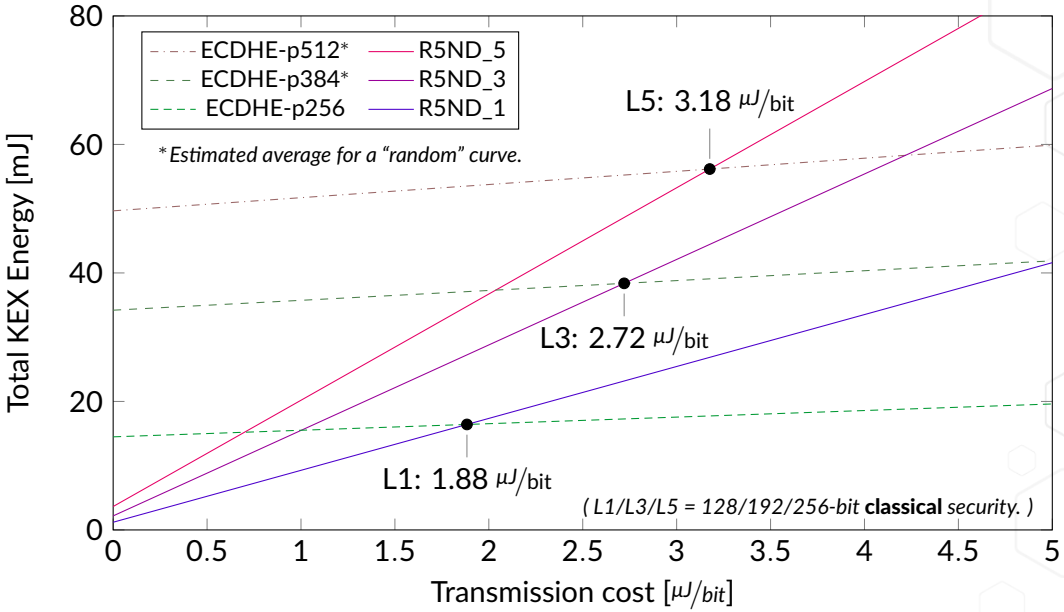
Use Case Observations

- PQC Lattice (RLWR) can be $\approx 10\times$ more efficient than current ECDHE but needs $\approx 10\times$ bytes.
- PQC Isogeny (SIDH) needs 30-60% of RLWR Lattice bytes, 300-2000 times more energy.

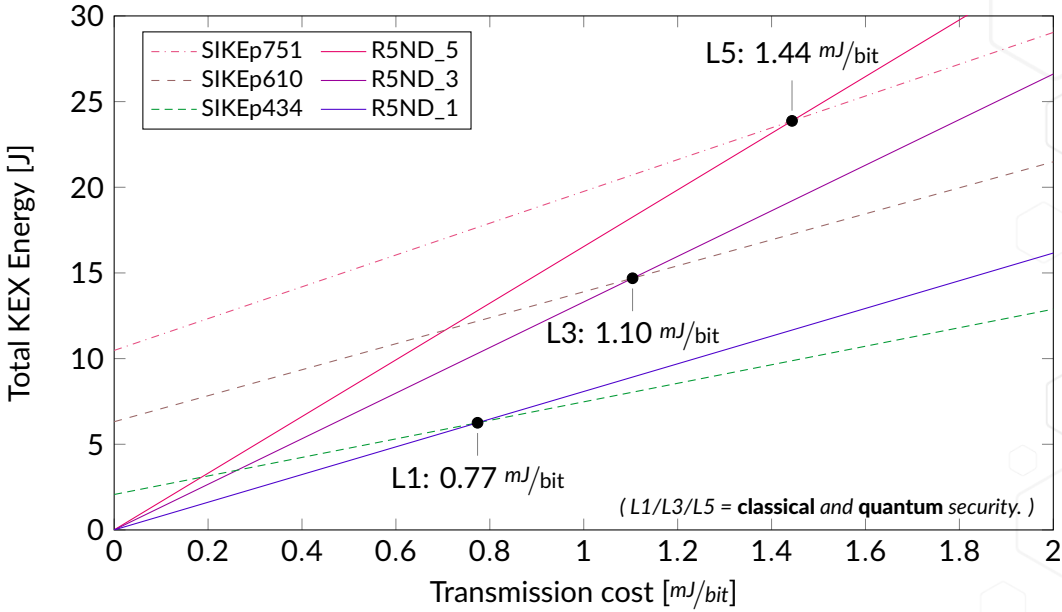
We can determine energy **crossover points** for e^{tx} [J/bit] from lattice schemes to elliptic curves and SIDH.



Crossover from Round5 to ECDHE at $e_{tx} \approx 2\mu J/bit$



Crossover from Round5 to SIKE at $e_{tx} \approx 1^{mJ}/_{bit}$



Summary: PQC Energy

What we found out:

- PQC algorithms have *really* distinctive "IoT" MCU power profiles !
- Energy usage range is **4 orders of magnitude** in NIST PQC 2nd round.
  - Energy ranking differs from time ranking most with $< 1mJ$ algorithms.
  - Some PQC schemes actually need **significantly less energy** than ECC.
- Not so much power variation in our Intel PC / Server measurements.
- Consider both **transmission cost** [J/bit] and **computation cost** [J/cycle].
- Plotting **total energy** cost as a function of e_{tx} is a good way to compare algorithms A and B, where other is faster but needs more bandwidth.

Cryptography engineering needs real-life measurements and data!

Talk Outline

I Introduction

II Symmetric Algorithms – Reasonably Stable

III Asymmetric (Public Key) Algorithms – In Transition

IV Engineering Case Study – PQC Energy [*skimming*]

V Measuring NIST Post-Quantum Crypto [*skimming*]

VI **Final Notes**

Recap

- **Security has a cost.** Implementation cost, non-implementation cost (loss), re-implementation cost, user experience cost, energy cost, ..
- **Use standard algorithms and protocols.** These have deep analysis, are understood by clients, and will reduce specification & implementation effort.
- **Avoid implementing cryptography yourself.** Use standard, frequently updated security providers and libraries like Microsoft CryptoAPI or OpenSSL.
- **Key generation and storage** needs special attention: Pseudorandom number generators and their seeds, handling of secret keys, tokens, and metadata.
- **Secure coding practices** and/or safer languages (rust) and platforms are helpful.
- **Automated tools** like static analysis and fuzzing help to catch bugs.
- **Independent outside assessment** before and after implementation, updates.

Use cryptography if you can. But don't *make it* unless you have to.

“Don’t roll your own crypto”

PQ SHIELD

“One of the most singular characteristics of the art of deciphering is the strong conviction possessed by every person, even moderately acquainted with it, that he is able to construct a cipher which nobody else can decipher.”

– **Charles Babbage** *(1864)*

However – if designing cryptosystems is your job:

- Map out the actual technical requirements of target platforms and applications.
- Design in a way that makes security proofs and arguments easy to make.
- Design for secure, efficient hardware and software implementation techniques.
- Security proofs have limited value against unknown attacks – attackers do not care about your “security model”. Think with adversarial mindset, outside box.
- Try all known attacks, publish, and trust peer review – Kerchoff’s principle.
- **Only few percent of cryptographic proposals end up in “production use”.**