LEDAcrypt

Marco Baldi[1], *Alessandro Barenghi*[2], Franco Chiaraluce[1],
Gerardo Pelosi[2], Paolo Santini[1]

[1]Università Politecnica delle Marche [2]Politecnico di Milano

Second NIST PQC Standardization Conference

The LEDAcrypt proposal

Merger of two proposals

- Merger of code based KEM (LEDAkem) and PKE (LEDApkc), using Quasi-Cyclic Low Density Parity Check (QC-LDPC) codes
- KEM built employing Niederreiter's trapdoor, PKE with McEliece's
- Targets:
  - Provide an IND-CCA2 KEM and IND-CCA2 PKE (NIST requires at least 2^{64} decryption oracle calls)
  - Provide an ephemeral key use-mode with IND-CPA security for perfect forward secrecy applications (e.g. TLS 1.3)

LEDAcrypt

Key Generation

1. Generate random $p \times n_0 p$ binary block circulant matrix $H = [H_0, \ldots, H_{n_0-1}]$ with $n_0 \in \{2, 3, 4\}$ circulant blocks, having column weight $d_v \ll n$, $n = n_0 p$, p prime
2. Generate a random, non-singular, $n_0 p \times n_0 p$ binary block circulant matrix Q made of $n_0 \times n_0$ circulant blocks, with total column weight $m \ll n$
3. Store private key: H, Q
4. $L = HQ = [L_0, \ldots, L_{n_0-1}]$; public key: $M = (L_{n_0-1})^{-1}[L_0, \ldots, L_{n_0-1}]$

Key Use

- In LEDAcrypt KEM and KEM-LT: employ M as a systematic parity-check matrix
- In LEDAcrypt PKE: employ M to obtain a systematic generator matrix

Questions from NIST (NISTIR 8240 + CBC'19)

Security related

- Is homogeneous syndrome decoding safe?
- Can you obtain a low enough DFR to provide IND-CCA2?
- Can you tackle somehow the additional structure of L w.r.t. QC-MDPC?

Performance related

- What is the cost (speed/bandwidth) of IND-CCA2 vs IND-CPA versions?
- What are the best computation vs bandwidth tradeoffs?
- Which n_0 should be picked?

What's new in round 2?

Security related

- (round 2) New decoder/code parameter sets to achieve low enough DFR
  - Parameter sets providing 2^{-64} and $2^{-seclevel}$ DFRs
- (round 2) Automated QC-LDPC parameter design procedure, employing ISD finite regime estimates
- (round 2+) Construction to match DFR to δ-correctness definition [HHK17] for IND-CCA2 KEM

Performance related

- (round 2) AVX2 implementation for decoder and arithmetic
- (round 2+) Further optimizations in AVX2 implementation (key generation phase)
- (this presentation) Highlight best tradeoffs in parameter choices

- Decision Syndrome Decoding (decision-SD) is NP-Complete [BMT78]
  - QC case proven NP-Complete in [BCGO09]
- Decision-Homogeneous-SD, a.k.a. decision codeword "finding", is NP-Complete [BMT78]
  - QC case can be proven NP-Complete (proof analogous to [BCGO09])
- All NP-Complete problems have a search to decision reduction [AB07, §2.5]
- LEDAcrypt problems, assuming public H is indistinguishable from random QC:
  - Decryption equivalent to search-Quasi Cyclic-Syndrome Decoding
  - Known key recovery techniques equiv. to search-QC-Homogenous-SD on dual code

Providing IND-CCA2 guarantees (1/2)

NIST Question: Can you obtain a low enough DFR to provide IND-CCA2?

- To obtain IND-CCA2 security decryption failures should be quantified (and few)
- In round 2 submission we proposed a new decoding strategy with a bounded Decoding Failure Rate (DFR), quantifying in turn decryption failures, providing
  - Parameter sets with 2^{-64} DFR to match the 2^{64} oracle calls requested by NIST
  - Parameter sets with $2^{-(\texttt{security-level})}$ DFR show the scalability up to the requirements for security proofs
- IND-CPA parameter sets for ephemeral key use were tuned to a $2^{-30} \approx 10^{-9}$ DFR
  - minimal hindrance even to high availability ($< 10^{-6}$ failures) applications

Providing IND-CCA2 guarantees (2/2)

NIST Question: Can you obtain a low enough DFR to provide IND-CCA2?

- A set of constructions provide IND-CCA2 guarantees in the ROM assuming that the underlying primitive is δ-correct [HHK17]
  - δ is the max-over-plaintexts, average-over-keys probability that an attacker (knowing the private key) is able to craft a valid ciphertext which fails decryption
  - δ-correctness does not match the usual definition of DFR of a code (average-over-error-vectors for a given key)
- If errors are randomly picked and DFR is bounded for all the keypairs we're ok
- To reconcile DFR and δ-correctness:
  - LEDAcrypt PKE: McEliece trapdoor, errors are randomly generated, plaintext independent$\rightarrow$ no need for modifications to reconcile
  - LEDAcrypt KEM: Niederreiter trapdoor: an attacker knowing the *private* key may choose plaintexts (i.e. error vectors) failing with Pr $>$ DFR $\rightarrow$ reconcile forcing the attacker to pick a random error vector (and verify that he does) with a construction

Parity-check matrix structure

NIST Question: Can you tackle somehow the additional structure of L w.r.t. QC-MDPC?

- Secret code in LEDAcrypt is defined by the product of two, low weight matrices, $L = HQ$, as opposed to a single, randomly drawn, moderate density (L')
- If size and weight of L match those of L', the keyspace for QC-LDPC is smaller than the one for corresponding QC-MDPC
  - Took into account in the parameter generation procedure (keyspace still $> 2^{400}$)
  - We also prevent separate enumeration of either H or Q alone
- The L matrix may have a column weight lower than expected
  - We perform rejection sampling to discard such keys (around 40%-50% rej. rate)
- No known methods to exploit the product structure to speed up ISD

Analyzing performance and bandwidth (Round 2+, all primitives)

Computation time

- Key generation: dominated by polynomial inverse (80% to 95+% of time)
    - [KTT12] and [BY19] inverse algorithms, batching techniques can be applied
- Encryption: dominated by polynomial multiplication (70%-90% of time)
- Decryption: dominated by syndrome decoding (85% to 90% of time)

Key sizes and required bandwidth

- Public keys are $(n_0 - 1)p$ bits wide, private keys compressed to `seed_size`
- Bandwidth requirements:
    - $n_0 p$ bits sent for KEM (ephemeral), p bits sent for KEM-LT
    - $n_0 = 3$ yields smallest KEM bandwidth for Cat. 1 and 3
    - $n_0 = 2$ yields smallest KEM bandwidth for Cat. 5

LEDAcrypt KEM-LT (IND-CCA2) Performance

Gray items refer to round 2 submission code, black ones to current optimizations.
Software running on an Intel i5-6500, 3.2 GHz

NIST Category	n_0	DFR	KeyGen (ms)	Encap. (ms)	Decap. (ms)	Enc+Dec time (ms)	Ctx size (kiB)
1	2	2^{-64}	6.87(295)	0.09(0.13)	0.33(0.41)	0.43	4.38
	2	2^{-128}	11.64(549)	0.16(0.16)	0.46(0.54)	0.63	6.37
3	2	2^{-64}	14.74(906)	0.24(0.25)	0.69(0.91)	0.99	7.07
	2	2^{-192}	30.17(1532)	0.42(0.54)	0.99(1.24)	1.42	11.75
5	2	2^{-64}	28.65(2521)	0.52(0.68)	1.33(1.41)	1.86	10.87
	2	2^{-256}	58.54(4252)	0.81(0.84)	2.04(2.28)	2.86	18.60

LEDAcrypt PKE (IND-CCA2) Performance

Gray items refer to round 2 submission code, black ones to current optimizations.
Software running on an Intel i5-6500, 3.2 GHz

NIST Category	n_0	DFR	KeyGen (ms)	Encap. (ms)	Decap. (ms)	Enc+Dec time (ms)
1	2	2^{-64}	6.87(290)	0.31(0.29)	0.69(0.76)	1.00
	2	2^{-128}	11.64(422)	0.44(0.42)	0.99(1.18)	1.42
3	2	2^{-64}	14.74(1187)	0.56(0.56)	1.30(1.70)	1.86
	2	2^{-192}	30.17(1538)	1.04(1.10)	2.03(2.39)	3.07
5	2	2^{-64}	28.65(2543)	1.03(1.02)	2.49(3.26)	3.52
	2	2^{-256}	58.54(4240)	1.62(1.53)	3.86(4.16)	5.48

LEDAcrypt KEM (IND-CPA) Performance

Gray items refer to round 2 submission code, black ones to current optimizations.
Software running on an Intel i5-6500, 3.2 GHz

NIST Category	n_0	KeyGen (ms)	Encap. (ms)	Decap. (ms)	Total exec. time (ms)	Ctx+kpub Size (kiB)
1	2	1.32(1.37)	0.06(0.04)	0.24(0.34)	1.62(1.75)	3.65
	3	0.50(0.56)	0.03(0.03)	0.23(0.42)	0.77(1.03)	3.04
	4	0.47(0.88)	0.02(0.04)	0.26(1.30)	0.76(2.23)	3.68
3	2	3.63(3.72)	0.12(0.09)	0.61(0.95)	4.37(4.76)	6.28
	3	1.72(1.79)	0.07(0.08)	0.54(1.11)	2.33(2.99)	5.91
	4	1.50(2.75)	0.07(0.11)	0.69(2.06)	2.27(4.93)	7.03
5	2	7.18(7.64)	0.20(0.17)	0.95(1.27)	8.35(9.09)	9.01
	3	4.64(4.96)	0.16(0.17)	1.05(1.62)	5.86(6.76)	10.05
	4	3.83(5.64)	0.13(0.21)	1.05(2.75)	5.02(8.61)	11.09

Cost of adding IND-CCA2

NIST Question: What is the cost (speed/bandwidth) of IND-CCA2 vs IND-CPA versions?

Comparison between IND-CPA and IND-CCA2 KEMs, synthetic metric μ computed as $\mu = \text{cycles} + 1000 \times B$, (B transmitted bytes). Ratio computed as $\frac{\mu_{CCA} - \mu_{CPA}}{\mu_{CPA}}$ selecting the best performing IND-CPA option (among $n_0 \in \{2, 3, 4\}$) for the security level. Red color highlights an extra cost for IND-CCA2, green highlights a saving.

NIST Category	n_0	DFR	$\frac{\text{cycles}_{CCA} - \text{cycles}_{CPA}}{\text{cycles}_{CPA}}$	$\frac{B_{CCA} - B_{CPA}}{B_{CPA}}$	$\frac{\mu_{cca} - \mu_{cpa}}{\mu_{cpa}}$
1	2	2^{-64}	-47.5%	44.6%	6.4%
	2	2^{-128}	-24.5%	109.7%	54.0%
3	2	2^{-64}	-58.2%	20.2%	-28.3%
	2	2^{-192}	-32.4%	99.5%	18.1%
5	2	2^{-64}	-69.3%	21.1%	-41.9%
	2	2^{-256}	-48.2%	106.7%	-1.2%

IND-CPA (ephemeral key) options require more computation but less bandwidth

Non-Algebraic, Hamming metric code-based KEMs, Long Term use

Figures from `supercop-20190816`, `Intel Xeon E3-1220 v3 (haswell)`, `hiphop`

Supercop tag	Time (kc) (**kcycles**)	transmitted (**B**)	cycles+1000×B
`ledakemlt10`	1512	4488	6000740
`hqc1281`	1603	6234	7837752
`ledakemlt11`	2292	6520	8812464
`ledakemlt30`	3260	7240	10500136
`hqc1921`	2789	10981	13770772
`hqc1922`	2901	11749	14650164
`ledakemlt50`	6414	11136	17550216
`ledakemlt31`	5793	12032	17825724
`hqc2561`	4309	15961	20270712
`hqc2562`	4576	16985	21561072
`hqc2563`	4695	17777	22472212
`ledakemlt51`	11393	19040	30433952

Non-Algebraic, Hamming metric code-based KEMs, Category 1, Eph. use

What are the best computation vs bandwidth tradeoffs? / Which n_0 should be picked?

Figures from `supercop-20190816`, `Intel Xeon E3-1220 v3 (haswell)`, `hiphop`

Supercop tag	Time (kc) (**kcycles**)	transmitted (**B**)	cycles+1000×B
`ledakem13`	2635	3120	5755764
`bike1l1nc`	1596	5084	6680112
`ledakem14`	2964	3776	6740276
`bike3l1nc`	1595	5516	7111960
`bike1l1`	3407	5084	8491364
`ledakem12`	5470	3744	9214880
`bike3l1`	4302	5516	9818592
`bike1l1sc`	4797	5084	9881160
`hqc1281`	1840	9359	11199668
`bike2l1`	7326	5084	12410180
`bike3l1sc`	6949	5516	12465900

Future directions

Decoder and code parameters

- Analysis of performance with $n_0 \in \{3, 4\}$ for KEM-LT/PKE
- Decoder with higher computational efficiency/correction capability
- Joint DFR/security parameter design
  - Possible IND-CCA2 parameter shrinking as a result

Implementations

- Finalizing constant time amd64 implementation
- Side-channel resistant Cortex-M4 implementation (PQClean project)
- ARMv7/ARMv8a optimized implementations
- Ongoing Xilinx Artix-7 implementation

Questions?

Thanks for the attention!

CPA/CCA2 comparison, CAT 3, CAT 5

Non-Algebraic, Hamming metric code-based KEMs, Category 3, Eph. use

All figures obtained from `supercop-20190811, Intel Xeon E3-1220 v5 (Skylake)`

Supercop tag	Time (kc) (**kcycles**)	transmitted (**B**)	c+1000×b
`ledakem33`	1,353	6,048	13539812
`ledakem34`	1,426	7,200	14269705
`hqc1921`	1,913	16,480	19139559
`hqc1922`	2,039	17,633	20391339
`ledakem32`	2,302	6,432	23024615

Non-Algebraic, Hamming metric code-based KEMs, Category 5, Eph. use

All figures obtained from `supercop-20190811, Intel Xeon E3-1220 v5 (Skylake)`

Supercop tag	Time (kc) (**kcycles**)	transmitted (**B**)	c+1000×b
`ledakem54`	16,681	11,360	28041294
`hqc2562`	4,214	25,488	29702525
`hqc2563`	4,369	26,674	31043365
`ledakem53`	21,836	10,296	32132565
`ledakem52`	35,343	9,232	44575781