From: D. J. Bernstein <djb@cr.yp.to>
Sent: Monday, June 24, 2019 1:39 PM
To: pqc-comments
Cc: pqc-forum
Subject: [pqc-forum] ROUND 2 OFFICIAL COMMENT: LEDAcrypt
Attachments: signature.asc

I mentioned this in separate discussions of lattice proofs, but I've realized that filing this as an official comment is better.

https://www.ledacrypt.org/documents/LEDAcrypt_spec_latest.pdf appeals to the HHK17 CCA theorems, but actually uses a different notion of failure probability, without commenting on the difference in the definitions.
It's not clear to me that this proof gap can be fixed.

---Dan

--

From: Marco Baldi <m.baldi@univpm.it>
Sent: Friday, July 5, 2019 10:19 AM
To: pqc-comments
Cc: pqc-forum
Subject: R: [pqc-forum] ROUND 2 OFFICIAL COMMENT: LEDAcrypt
Attachments: LEDACrypt_KEM-LT.pdf

Dear all,

please find our comments below.

Kind regards,
LEDAcrypt team

From: D. J. Bernstein
Sent: monday 24 june 2019 21:41
To: pqc-comments@nist.gov
Cc: pqc-forum@list.nist.gov
Subject: [pqc-forum] ROUND 2 OFFICIAL COMMENT: LEDAcrypt

> I mentioned this in separate discussions of lattice proofs, but I've realized
> that filing this as an official comment is better.
>
>> https://www.ledacrypt.org/documents/LEDAcrypt_spec_latest.pdf
> appeals to the HHK17 CCA theorems, but actually uses a different notion of
> failure probability, without commenting on the difference in the definitions.

Thanks for pointing this fact out

> It's not clear to me that this proof gap can be fixed.

The delta-correctness definition in HHK17 considers the probability of failure of the decryption primitive on a legitimate ciphertext, averaged over all possible key pairs and taken as the maximum achievable by an adversary who knows both the public and private key of the scheme.

In our decoding failure analysis, we consider an upper bound on the decoding failure rate for any given keypair, assuming that the adversary is randomly picking error vectors.

* Concerning LEDACrypt PKC: our way of considering the decoding failures matches the requirements of the definitions by HHK17. In fact, due to the KI-gamma construction the maximization of the decoding failure rate over all the plaintexts has no effects, since the message is employed as a codeword of the underlying McEliece scheme, and thus does not influence the decoding failure probability.
Indeed, the error vector in the KI-gamma construction is picked as the output of a hash (encoded with a constant weight encoding procedure) and thus the attacker is not able to choose a specific error pattern unless

he finds a preimage for the said hash.

* Concerning LEDACrypt KEM(-LT): The cryptoscheme is Niederreiter-based, and currently there is a difference between the DFR definition employed by us and the one in HHK17.
This mismatch can be compensated employing a construction similar to the ones reported by Dent in [1] to enforce the generation of the error via a cryptographically safe PRNG, fed with the hash of a randomly drawn seed. This can be done efficiently by sending, concatenated to the syndrome, the seed of the PRNG blinded with the hash of the error vector.
The receiver will start by decoding the received syndrome, then hash the obtained error message and recover the value of the seed of the PRNG. The recovered seed can then be used to regenerate the error vector via the PRNG and check it against the one obtained via decoding.

Employing this construction allows an attacker to pick maliciously only the seed, but, under the assumption of randomness of the output of the hash, this will pick an error vector uniformly at random from the available ones. Therefore, all messages (i.e., seeds) chosen by an adversary will have a probability of causing a decoding failure equal to the one of picking a random error vector of proper weight. As a consequence, our definition of DFR will match the HHK17 definition of the value delta.

Attached to this e-mail we provide a PDF with a typeset sketch of the construction, and we will integrate it, together with the outcome of this discussion, updating the LEDACrypt KEM-LT specification and implementation.

[1] Alexander W. Dent: A Designer's Guide to KEMs. IMA Int. Conf. 2003: 133-151

IND-CCA2 Conversion for LEDACrypt KEM-LT

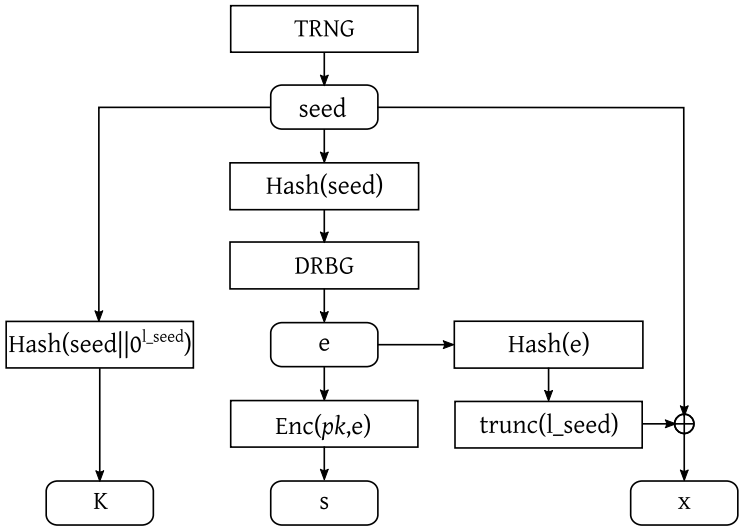

Figure 1: LEDACrypt KEM-LT encapsulation $s, x \leftarrow \text{ENCAP}(pk, seed)$. s and x are transmitted from the sender to the receiver. The derived session key is denoted as K, l_seed denotes the length in bits of the seed and $trunc(\cdot)$ denotes bitwise truncation.

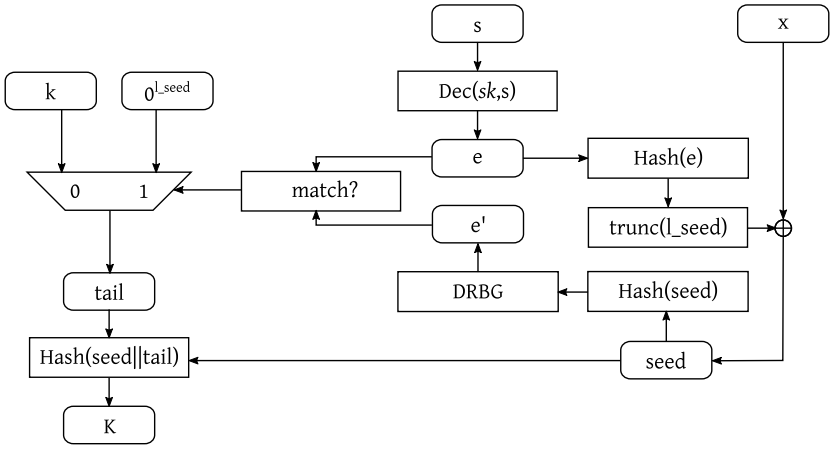

Figure 2: LEDACrypt KEM-LT decapsulation. s and x are received from the sender. The session K is derived after reconstructing the value of the seed, and checking that the DRBG based procedure effectively regenerates an error vector e' matching the one output by $Dec(sk, s)$. k is a secret value, as long as the seed, known to the receiver only, and thus part of the secret key.

From: Marco Baldi <m.baldi@univpm.it>
Sent: Tuesday, July 30, 2019 5:00 PM
To: pqc-comments
Cc: pqc-forum
Subject: LEDAcrypt implementation update

Dear all,

an updated implementation of the LEDAcrypt primitives has been released and is available here:
https://www.ledacrypt.org/archives/LEDAcrypt_implementation_latest.zip

Best regards,
LEDAcrypt team

From: Marco Baldi <m.baldi@univpm.it>
Sent: Thursday, August 1, 2019 9:25 AM
To: pqc-comments
Cc: pqc-forum
Subject: Re: LEDAcrypt implementation update

Dear all,

in addition to the announcement of the updated implementation of the LEDAcrypt primitives (available at https://www.ledacrypt.org/archives/LEDAcrypt_implementation_latest.zip), let us provide some details.

The latest implementation was finalized to be complete and proper as per the requirements of round 2 (e.g., KATs are included in the referenced .zip file). We also sent it to the SUPERCOP team.

Both the reference codebase and the optimized codebase were re-engineered to improve maintainability, e.g., de-duplicating code portions shared by the three proposed primitives:
LEDAcrypt KEM (i.e., a IND-CPA KEM with ephemeral keys),
LEDAcrypt KEM-LT (i.e., a IND-CCA2 KEM with long-term keys), and
LEDAcrypt PKC (i.e., a IND-CCA2 PKC with long-term keys).

Concerning performance:
* we improved the optimized implementation of arithmetic operations and of the decoding algorithm;
* we implemented the recent polynomial inversion algorithm appeared in [BY2019], which is faster than the one we adopted previously, to speed up the execution of key-generation algorithms;
* we further optimized the implementation of key-generation algorithms (presented in the round 2 specification document) for KEM-LT and PKC obtaining a decrease of the execution times by a factor around 100 w.r.t. the ones of the implementation we provided on March 2019 at the beginning of round 2.

Finally, we updated the implementation of the LEDAcrypt KEM-LT scheme to integrate the IND-CCA2 construction described in the Official Comment posted on July 5th 2019.

Overall, the total execution time of LEDAcrypt KEM is improved by a factor in the (1.1; 2.9) range w.r.t. the previous version, mainly due to the improved implementation of the arithmetic as well as of the decoding algorithm.
Concerning LEDAcrypt KEM-LT and LEDAcrypt PKC, the execution times of their key-generation algorithms are substantially improved by a 100+ factor, while the execution times of their enc+dec operations are improved by a factor in the (1.1; 1.3) range.

[BY2019] Bernstein, D., & Yang, B.-Y. (2019). ''Fast constant-time gcd computation and modular inversion.'' IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(3), 340-398. https://doi.org/10.13154/tches.v2019.i3.340-398

Best regards,
LEDAcrypt team

From: 'daniel.apon' via pqc-forum <pqc-forum@list.nist.gov>
Sent: Friday, August 2, 2019 7:50 PM
To: pqc-forum
Cc: pqc-comments
Subject: [pqc-forum] Re: LEDAcrypt implementation update

Hi Marco / LEDAcrypt team,

Just to confirm: Only your implementation has changed (thanks for giving a preview of the projected performance improvements!), but nothing like the algorithmic design nor the underlying parameters have changed, right?

Thanks!
--Daniel Apon, NIST PQC

On Thursday, August 1, 2019 at 9:25:22 AM UTC-4, Marco Baldi wrote:

> Dear all,
>
> in addition to the announcement of the updated implementation of the LEDAcrypt primitives (available at https://www.ledacrypt.org/archives/LEDAcrypt_implementation_latest.zip), let us provide some details.
>
> The latest implementation was finalized to be complete and proper as per the requirements of round 2 (e.g., KATs are included in the referenced .zip file). We also sent it to the SUPERCOP team.
>
> Both the reference codebase and the optimized codebase were re-engineered to improve maintainability, e.g., de-duplicating code portions shared by the three proposed primitives:
> LEDAcrypt KEM (i.e., a IND-CPA KEM with ephemeral keys),
> LEDAcrypt KEM-LT (i.e., a IND-CCA2 KEM with long-term keys), and
> LEDAcrypt PKC (i.e., a IND-CCA2 PKC with long-term keys).
>
> Concerning performance:
> * we improved the optimized implementation of arithmetic operations and of the decoding algorithm;
> * we implemented the recent polynomial inversion algorithm appeared in [BY2019], which is faster than the one we adopted previously, to speed up the execution of key-generation algorithms;
> * we further optimized the implementation of key-generation algorithms (presented in the round 2 specification document) for KEM-LT and PKC obtaining a decrease of the execution times by a factor around 100 w.r.t. the ones of the implementation we provided on March 2019 at the beginning of round 2.
>
> Finally, we updated the implementation of the LEDAcrypt KEM-LT scheme to integrate the IND-CCA2 construction described in the Official Comment posted on July 5th 2019.
>
> Overall, the total execution time of LEDAcrypt KEM is improved by a factor in the (1.1; 2.9) range w.r.t. the previous version, mainly due to the improved implementation of the arithmetic as well as of the decoding algorithm.
> Concerning LEDAcrypt KEM-LT and LEDAcrypt PKC, the execution times of their key-generation algorithms are substantially improved by a 100+ factor, while the execution times of their enc+dec operations are improved by a factor in the (1.1; 1.3) range.
>
> [BY2019] Bernstein, D., & Yang, B.-Y. (2019). ''Fast constant-time gcd computation and modular inversion.'' IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(3), 340-398. https://doi.org/10.13154/tches.v2019.i3.340-398

From: Marco Baldi <m.baldi@univpm.it>
Sent: Sunday, August 4, 2019 10:32 AM
To: Apon, Daniel C. (Fed); pqc-forum
Cc: pqc-comments
Subject: Re: [pqc-forum] Re: LEDAcrypt implementation update

Dear Daniel and dear all,

we confirm that no changes were made either to the algorithms or to the parameter choices.

Only the IND-CCA2 construction for the KEM-LT primitive was added following the official comment posted on July 5th.

Thank you and best regards,
LEDAcrypt team

From: 'daniel.apon' via pqc-forum <pqc-forum@list.nist.gov>
Sent: sabato 3 agosto 2019 01:49
To: pqc-forum <pqc-forum@list.nist.gov>
Cc: pqc-comments@nist.gov <pqc-comments@nist.gov>
Subject: [pqc-forum] Re: LEDAcrypt implementation update

Hi Marco / LEDAcrypt team,

Just to confirm: Only your implementation has changed (thanks for giving a preview of the projected performance improvements!), but nothing like the algorithmic design nor the underlying parameters have changed, right?

Thanks!
--Daniel Apon, NIST PQC

On Thursday, August 1, 2019 at 9:25:22 AM UTC-4, Marco Baldi wrote:

> Dear all,
>
> in addition to the announcement of the updated implementation of the LEDAcrypt primitives (available at https://www.ledacrypt.org/archives/LEDAcrypt_implementation_latest.zip), let us provide some details.
>
> The latest implementation was finalized to be complete and proper as per the requirements of round 2 (e.g., KATs are included in the referenced .zip file). We also sent it to the SUPERCOP team.
>
> Both the reference codebase and the optimized codebase were re-engineered to improve maintainability, e.g., de-duplicating code portions shared by the three proposed primitives:
> LEDAcrypt KEM (i.e., a IND-CPA KEM with ephemeral keys),
> LEDAcrypt KEM-LT (i.e., a IND-CCA2 KEM with long-term keys), and
> LEDAcrypt PKC (i.e., a IND-CCA2 PKC with long-term keys).