From: Tobias Schneider <to.schneider72@gmail.com> via pqc-forum@list.nist.gov
To: pqc-forum <pqc-forum@list.nist.gov>
Subject: [pqc-forum] Implicit Rejection in Kyber
Date: Friday, December 16, 2022 04:09:04 AM ET

Hi PQC mailing list,
We have been studying the implicit rejection and have some observations regarding the role of the long-term secret value ‘z’ in Kyber.

In Kyber.CCAKEM.Dec, ‘z’ is used to implement implicit rejection as

if c = c’ then

return K := KFD(\bar{K}’ || H(c))

else

return K := KDF(z || H(c))

We look at this from a high-assurance implementation (i.e., side-channel protected) point of view.
If ‘z’ requires the same level of protection against physical attacks as the secret key ‘s’ then this results in significant costs in storage (storing shares of the masked ‘z’). Moreover, the branch K := KDF(z || H(c)) needs an DPA-protected KDF (assuming ‘z’ is secret) which is not necessarily the case for the K := KFD(\bar{K}’ || H(c)) branch.

Academic works have commonly left this part unprotected in masked implementations of Kyber / Saber (cf. [1-4]), which removes the unpredictability property of the rejection mechanism (‘z’ is assumed to be leaked in full).
Therefore, for high-assurance implementations a switch to an explicit rejection mechanism would be preferable and improve the efficiency.

If implicit rejection is a design goal which is desirable (but is it really?) we suggest one of the following.
What about optionally randomizing ‘z’?
This could be achieved by using K := KDF((z xor R) || H(c)) where ‘R’ denotes an optionally true random element.
For ‘R==0’ it implements the current solution as specified in the specification while for ‘R==random’ it randomizes the input to the KDF, which removes the strict DPA-protection requirement and makes high assurance implementations much simpler.

What do you (= the Kyber team, NIST and the entire PQC community) think about such a change to the rejection?
Any obvious downsides we overlooked?

Best regards
NXP PQC Team

[1] https://eprint.iacr.org/2020/733.pdf
[2] https://eprint.iacr.org/2021/483.pdf
[3] https://eprint.iacr.org/2022/158.pdf
[4] https://eprint.iacr.org/2022/389.pdf

From: D. J. Bernstein <djb@cr.yp.to> via pqc-forum@list.nist.gov
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Friday, December 16, 2022 04:54:04 AM ET
Attachments: smime.p7m

> K := KDF((z xor R) || H(c)) where ‘R’
> denotes an optionally true random element.

Switching the implicit-rejection z to a new value can lead to very easy breaks of IND-CCA2. This is how https://cr.yp.to/papers.html#ntrw breaks IND-CCA2 for NTRU-HRSS in the presence of a natural DRAM fault.

There are some systems with another layer of defense beyond implicit rejection (e.g., the plaintext confirmation in sntrup), but in general implementors should treat implicit rejection as fragile, should presume that any tweak to implicit rejection is a disaster, and should make sure to protect z and computations flowing from z as critical secrets.

The same paper also has a new survey of chosen-ciphertext attacks and defenses, looking at a wide range of defenses from the attacker's perspective. There are also various papers proving things about some defenses, and it's important to look carefully at exactly what's covered by these proofs and what isn't.

——D. J. Bernstein

From: Mike Hamburg <mike@shiftleft.org> via pqc-forum@list.nist.gov
To: Tobias Schneider <to.schneider72@gmail.com>
CC: pqc-forum <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Friday, December 16, 2022 08:14:18 AM ET

Hi Tobias,

Part of the point of implicit rejection is that it enables security proof techniques which are otherwise unavailable in the quantum random oracle model. I'm not up-to-date on the latest security proofs of Kyber's Fujisaki-Okamoto mode, but it improves tightness in several of the ones I've seen in the past.

If z is randomized, then an adversary who can observe a function of the output can determine whether the ciphertext was rejected. So you lose the advantage of implicit rejection in the formal security analysis of Kyber, though some practical advantages may remain. That is, some attackers might not be able to observe a function of the output, and perhaps implicit rejection leaves fewer avenues clients of this function to make serious mistakes.

In my opinion, it's pretty unlikely that explicit rejection makes a real difference in the formal security of Kyber: I think the supposed advantage there is an artifact of the proof. But it does make the API cleaner, and once you're doing that you might as well enjoy the proof and testability advantages.

I agree that the side-channel issues are annoying, but z is hardly the largest object you'll have to store in shares.

Regards,

— Mike

On Dec 16, 2022, at 10:08 AM, Tobias Schneider <to.schneider72@gmail.com> wrote:

Hi PQC mailing list,
We have been studying the implicit rejection and have some observations regarding the role of the long-term secret value ‘z’ in Kyber.

In Kyber.CCAKEM.Dec, ‘z’ is used to implement implicit rejection as

if c = c’ then

return K := KFD(\bar{K}’ || H(c))

else

return K := KDF(z || H(c))

We look at this from a high-assurance implementation (i.e., side-channel protected) point of view.
If ‘z’ requires the same level of protection against physical attacks as the secret key ‘s’ then this results in significant costs in storage (storing shares of the masked ‘z’).
Moreover, the branch K := KDF(z || H(c)) needs an DPA-protected KDF (assuming ‘z’ is secret) which is not necessarily the case for the K := KFD(\bar{K}’ || H(c)) branch.

Academic works have commonly left this part unprotected in masked implementations of Kyber / Saber (cf. [1-4]), which removes the unpredictability property of the rejection mechanism (‘z’ is assumed to be leaked in full).
Therefore, for high-assurance implementations a switch to an explicit rejection mechanism would be preferable and improve the efficiency.

If implicit rejection is a design goal which is desirable (but is it really?) we suggest one of the following.
What about optionally randomizing ‘z’?
This could be achieved by using K := KDF((z xor R) || H(c)) where ‘R’ denotes an optionally true random element.
For ‘R==0’ it implements the current solution as specified in the specification while for ‘R==random’ it randomizes the input to the KDF, which removes the strict DPA-

protection requirement and makes high assurance implementations much simpler.

What do you (= the Kyber team, NIST and the entire PQC community) think about such a change to the rejection?
Any obvious downsides we overlooked?

Best regards
NXP PQC Team

[1] https://eprint.iacr.org/2020/733.pdf
[2] https://eprint.iacr.org/2021/483.pdf
[3] https://eprint.iacr.org/2022/158.pdf
[4] https://eprint.iacr.org/2022/389.pdf

From: Simon Hoerder <simon@hoerder.net> via pqc-forum@list.nist.gov
To: pqc-forum <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Friday, December 16, 2022 10:56:28 AM ET

Hi,
Apologies for a somewhat stupid question: Does the implicit rejection serve any purpose beyond enabling a better security proof?

I am thinking about the following scenario:

```
def high_level_protocol(...):

...

K = kyber_decaps(...)

# the decapsulation failed, K == KDF(z || H(c))

reply = symmetric_send_recv(K, msg)

If (reply == error):

... # Handle error
```

I would assume that we have three problems at this point:

1. The CCA2 proof is broken because the adversary will be able to distinguish the error handling.
2. We just amplified the denial of service attack surface as we've delayed the abort.
3. We just lost reaction time if the error handling requires safety-relevant actions.

Having `K = [KDF(z || H(c)), ERROR]` would be preferable in my opinion as long as that doesn't cause security issues within Kyber. `K = ERROR` would work as well but is more prone to pebkac coding vulnerabilities in the higher level protocol implementation as Mike Hamburg reminded me in an offline chat.

Thanks,

Simon

On 16 Dec 2022, at 14:14, Mike Hamburg <mike@shiftleft.org> wrote:

Hi Tobias,
Part of the point of implicit rejection is that it enables security proof techniques which are

otherwise unavailable in the quantum random oracle model. I'm not up-to-date on the latest

security proofs of Kyber's Fujisaki-Okamoto mode, but it improves tightness in several of

the ones I've seen in the past.

If z is randomized, then an adversary who can observe a function of the output can determine

whether the ciphertext was rejected. So you lose the advantage of implicit rejection in the

formal security analysis of Kyber, though some practical advantages may remain. That is,

some attackers might not be able to observe a function of the output, and perhaps implicit

rejection leaves fewer avenues clients of this function to make serious mistakes.

In my opinion, it's pretty unlikely that explicit rejection makes a real difference in the formal

security of Kyber: I think the supposed advantage there is an artifact of the proof. But it

does make the API cleaner, and once you're doing that you might as well enjoy the proof

and testability advantages.

I agree that the side-channel issues are annoying, but z is hardly the largest object you'll

have to store in shares.

Regards,

— Mike

> On Dec 16, 2022, at 10:08 AM, Tobias Schneider <to.schneider72@gmail.com> wrote:
>
> Hi PQC mailing list,
> We have been studying the implicit rejection and have some observations regarding the role of the long-term secret value ‘z’ in Kyber.
>
> In Kyber.CCAKEM.Dec, ‘z’ is used to implement implicit rejection as
>
> if c = c’ then
>
> return K := KFD(\bar{K}’ || H(c))
>
> else
>
> return K := KDF(z || H(c))
>
> We look at this from a high-assurance implementation (i.e., side-channel protected) point of view.
> If ‘z’ requires the same level of protection against physical attacks as the secret key ‘s’ then this results in significant costs in storage (storing shares of the masked ‘z’). Moreover, the branch K := KDF(z || H(c)) needs an DPA-protected KDF (assuming ‘z’ is secret) which is not necessarily the case for the K := KFD(\bar{K}’ || H(c)) branch.
>
> Academic works have commonly left this part unprotected in masked implementations of Kyber / Saber (cf. [1-4]), which removes the unpredictability property of the rejection mechanism (‘z’ is assumed to be leaked in full).
> Therefore, for high-assurance implementations a switch to an explicit rejection mechanism would be preferable and improve the efficiency.

If implicit rejection is a design goal which is desirable (but is it really?) we suggest one of the following.
What about optionally randomizing ‘z’?
This could be achieved by using K := KDF((z xor R) || H(c)) where ‘R’ denotes an optionally true random element.
For ‘R==0’ it implements the current solution as specified in the specification while for ‘R==random’ it randomizes the input to the KDF, which removes the strict DPA-protection requirement and makes high assurance implementations much simpler.

What do you (= the Kyber team, NIST and the entire PQC community) think about such a change to the rejection?
Any obvious downsides we overlooked?

Best regards
NXP PQC Team

[1] https://eprint.iacr.org/2020/733.pdf
[2] https://eprint.iacr.org/2021/483.pdf
[3] https://eprint.iacr.org/2022/158.pdf
[4] https://eprint.iacr.org/2022/389.pdf

From: D. J. Bernstein <djb@cr.yp.to> via pqc-forum@list.nist.gov
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Friday, December 16, 2022 01:09:06 PM ET
Attachments: smime.p7m

Simon Hoerder writes:

```
> K = kyber_decaps(…)
> # the decapsulation failed, K == KDF(z || H(c))
> reply = symmetric_send_recv(K, msg)
> If (reply == error):
>     … # Handle error
```

Yes, _if_ the symmetric protocol includes good enough authentication then this can have the same effect as wrapping plaintext confirmation around the KEM, rescuing security even if implicit rejection collapses. In the type of attack addressed by implicit rejection, the attacker is trying to convert an encryption of m into an encryption of a modified message m', but has no way to convert H(m) into H(m') for any reasonable hash function H, so an early enough check of H will stop the attack.

The proof view of this would say that if the objective is to convert a KEM into, e.g., a ROM IND-CCA2 PKE (let's not worry about the difference between ROM IND-CCA2 and IND-CCA2), then one can start with a weaker property than IND-CCA2 for the KEM, either imitating the monolithic REACT idea or factoring through a simpler plaintext-confirmation step (which can send a hash of the session key rather than an arbitrary hash of m, so it can treat the KEM as a black box).

However, given that NIST specified the KEM objective as IND-CCA2, I'd expect people to be plugging NIST-approved KEMs into a wider range of protocols, including protocols that really do need the full power of IND-CCA2 rather than something weaker——and then what happens if the KEM implementation effectively removes implicit rejection?

One way out would be for NIST to separately validate two explicitly labeled universes, one for implementations of the full KEM aiming for

IND-CCA2 and one for implementations that skip (or don't guarantee security of) implicit rejection, with protocol designers choosing which one they need. But I would recommend against this split unless there's solid evidence that the cost savings will make a big difference in deployment. I'm reminded of what

   https://www.imperialviolet.org/2018/12/12/cecpq2.html

said about having an IND-CPA universe and an IND-CCA2 universe: "CPA vs CCA security is a subtle and dangerous distinction, and if we're going to invest in a post-quantum primitive, better it not be fragile."

——D. J. Bernstein

From: Mike Hamburg <mike@shiftleft.org> via pqc-forum@list.nist.gov
To: D. J. Bernstein <djb@cr.yp.to>
CC: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Sunday, December 18, 2022 06:30:32 AM ET

> On Dec 16, 2022, at 10:53 AM, D. J. Bernstein <djb@cr.yp.to> wrote:
>
> > K := KDF((z xor R) || H(c)) where ‘R’
> > denotes an optionally true random element.
>
> Switching the implicit-rejection z to a new value can lead to very easy breaks of IND-CCA2. This is how https://cr.yp.to/papers.html#ntrw breaks IND-CCA2 for NTRU-HRSS in the presence of a natural DRAM fault.
>
> There are some systems with another layer of defense beyond implicit rejection (e.g., the plaintext confirmation in sntrup), but in general implementors should treat implicit rejection as fragile, should presume that any tweak to implicit rejection is a disaster, and should make sure to protect z and computations flowing from z as critical secrets.
>
> The same paper also has a new survey of chosen-ciphertext attacks and defenses, looking at a wide range of defenses from the attacker's perspective. There are also various papers proving things about some defenses, and it's important to look carefully at exactly what's covered by these proofs and what isn't.
>
> ---D. J. Bernstein

Perhaps this is a sidetrack, but...

This exchange clarified my intuition on something that hadn’t been

entirely clear before, which is why Kyber and SABER and ThreeBears etc

had different security effects (according to both proofs and attacks) from NTRU and McEliece etc in regard to implicit vs explicit rejection, and in regard to plaintext confirmation vs none.

For [R/M] LWE NewHope v2-like systems, the plaintext is used as the seed to a pseudorandom number generator (and in the analysis it's modeled as a random oracle) that's used to generate the noise for the encryption. This introduces extra looseness into the QROM proof, and extra complexity for side-channel protections. But it ought to remove the need for the plaintext confirmation hash: in some sense the ciphertext is already a hash of the plaintext. Likewise, in the same way that schemes with a plaintext confirmation hash may not need implicit rejection (in the QROM, under certain conditions: https://eprint.iacr.org/2019/590.pdf, Theorem 4), intuitively I expect these LWE-based systems not to need implicit rejection either.

I've waved my hands a lot in the above ("intuitively" etc) to indicate that one must actually go through the analysis to be confident. I did a QROM analysis for ThreeBears when I submitted it, and I would expect it to work also for Frodo and likely Kyber, SABER etc, but at least for Kyber and SABER there's some fiddling to do because of the rounding steps. But this is what was behind my confidence when I wrote that implicit vs explicit rejection probably doesn't make a real difference in Kyber's security.

On the other hand, Classic McEliece and (at least some flavors of)

NTRU don’t use this structure. Instead, the plaintext is a low-Hamming-weight vector that’s recovered by decryption, and a hash function is not necessarily required before multiplication by the public key (whether that’s f/g in NTRU or a parity matrix in Classic McEliece). This structure gives a tighter QROM reduction and a simpler variant of the FO transform. But the ciphertext really isn’t in any sense a cryptographic hash of the plaintext, and instead it is somewhat malleable. This means that you really do need either implicit rejection or plaintext confirmation to retain CCA security.

Anyway, I hope that explanation made sense and is useful to other people.

Regards,

— Mike

From: Thom Wiggers <thom@thomwiggers.nl> via pqc-forum@list.nist.gov
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Monday, December 19, 2022 05:00:52 AM ET

Hi Dan, all,

Op vr 16 dec. 2022 om 19:09 schreef D. J. Bernstein <djb@cr.yp.to>:

> One way out would be for NIST to separately validate two explicitly labeled universes, one for implementations of the full KEM aiming for IND-CCA2 and one for implementations that skip (or don't guarantee security of) implicit rejection, with protocol designers choosing which one they need. But I would recommend against this split unless there's solid evidence that the cost savings will make a big difference in deployment. I'm reminded of what
>
> https://www.imperialviolet.org/2018/12/12/cecpq2.html
>
> said about having an IND-CPA universe and an IND-CCA2 universe: "CPA vs CCA security is a subtle and dangerous distinction, and if we're going to invest in a post-quantum primitive, better it not be fragile."

I would like to illustrate this point by the example of the (PQ) TLS 1.3 and KEMTLS ephemeral key exchange. The common understanding is that for ephemeral key exchange, IND-CPA ought to be enough, right? However, it turns out that in these protocols one needs to be able to answer a (single) decapsulate query to make the proofs work: something that can be referred to as IND-1CCA (or IND-nCCA). [1]

In an earlier draft of the KEMTLS paper, we fell into the IND-CPA trap but a very careful expert analysis taught us that we needed at least IND-1CCA security. In summary, while IND-CPA can be a bit cheaper, allowing it seems like opening a very big trap door. As someone who is just writing papers and "having a bit of fun" with arrows on a whiteboard, it is a lot of fun to have more toys in my toolbox; but I don't mind if someone shows me I shot myself in the foot by

publishing a nice attack paper on one of my proposals. However, taking those footguns away from e.g. my bank would probably be a wise move.

Cheers,

Thom

[1] Further reading: https://eprint.iacr.org/2021/844 by Loïs Huguenin-Dumittan and Serge Vaudenay is a nice paper that further investigates this issue and shows how to construct IND-nCCA KEMs from OW-CPA secure KEMs cheaper than the full FO transform.

> ---D. J. Bernstein
>
> --
> You received this message because you are subscribed to the Google Groups "pqc-forum" group.
> To unsubscribe from this group and stop receiving emails from it, send an email to pqc-forum+unsubscribe@list.nist.gov.
> To view this discussion on the web visit https://groups.google.com/a/list.nist.gov/d/msgid/pqc-forum/20221216180834.222129.qmail%40cr.yp.to.

--
You received this message because you are subscribed to the Google Groups "pqc-forum" group.
To unsubscribe from this group and stop receiving emails from it, send an email to pqc-forum+unsubscribe@list.nist.gov.
To view this discussion on the web visit https://groups.google.com/a/list.nist.gov/d/msgid/pqc-forum/CABzBS7mEAWmG6HSSXa_hBjHUBLOv4%3DAfzMiVX%2BS2LNtBVro5aw%40mail.gmail.com.

From: Christian Majenz <christian.majenz@gmail.com> via pqc-forum@list.nist.gov
To: Mike Hamburg <mike@shiftleft.org>
CC: D. J. Bernstein <djb@cr.yp.to>, pqc-forum@list.nist.gov, Kathrin Hövelmanns <kathrin@hoevelmanns.net>, Andreas Hülsing <andreas@huelsing.net>
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Tuesday, December 20, 2022 03:31:51 AM ET

Dear all,

On 18-12-2022 12:28, Mike Hamburg wrote:
“For [R/M] LWE NewHope v2-like systems, the plaintext is used as the seed to a pseudorandom number generator (and in the analysis it’s modeled as a random oracle) that’s used to generate the noise for the encryption. This introduces extra looseness into the QROM proof, and extra complexity for side-channel protections. But it ought to remove the need for the plaintext confirmation hash: in some sense the ciphertext is already a hash of the plaintext. Likewise, in the same way that schemes with a plaintext confirmation hash may not need implicit rejection (in the QROM, under certain conditions: https://eprint.iacr.org/2019/590.pdf, Theorem 4), intuitively I expect these LWE-based systems not to need implicit rejection either.”

We can actually back up Mike’s intuition. Our recent work https://eprint.iacr.org/2022/365 (Asiacrypt 2022) implies that for such (NewHope v2-like) KEMs explicit rejection without plaintext confirmation is as secure as implicit rejection from a provable security perspective in the QROM*.

Indeed, our proof makes crucial use of the derandomization step described by Mike, where the coins used to encrypt a message m are obtained as H(m) for a random oracle H. This derandomization step is often referred to as T-transform.

Our result does not cover schemes like Classic McEliece or NTRU-HRSS which start from deterministic, perfectly-correct PKE and hence do not make use of the T-transform.

*(Our new reduction and previous ones differ in how decryption failures are handled, leading to different requirements on scheme-specific characterizations.)
Best wishes,Kathrin, Chris, Andreas

~~~~

Christian Majenz
~~~~

Assistant Professor

DTU Compute

chmaj@dtu.dk

Website:https://www.christianmajenz.info

On 18 Dec 2022, at 12:28, Mike Hamburg <mike@shiftleft.org> wrote:

> On Dec 16, 2022, at 10:53 AM, D. J. Bernstein <djb@cr.yp.to> wrote:
>
> > K := KDF((z xor R) || H(c)) where ‘R’
> > denotes an optionally true random element.
>
> Switching the implicit-rejection z to a new value can lead to very easy breaks of IND-CCA2. This is how https://cr.yp.to/papers.html#ntrw breaks IND-CCA2 for NTRU-HRSS in the presence of a natural DRAM fault.
>
> There are some systems with another layer of defense beyond implicit rejection (e.g., the plaintext confirmation in sntrup), but in general implementors should treat implicit rejection as fragile, should presume that any tweak to implicit rejection is a disaster, and should make sure to protect z and computations flowing from z as critical secrets.
>
> The same paper also has a new survey of chosen-ciphertext attacks and defenses, looking at a wide range of defenses from the attacker's perspective. There are also various papers proving things about some defenses, and it's important to look carefully at exactly what's covered by these proofs and what isn't.
>
> ---D. J. Bernstein

Perhaps this is a sidetrack, but...

This exchange clarified my intuition on something that hadn’t been

entirely clear before, which is why Kyber and SABER and ThreeBears etc

had different security effects (according to both proofs and attacks)

from NTRU and McEliece etc in regard to implicit vs explicit

rejection, and in regard to plaintext confirmation vs none.

For [R/M] LWE NewHope v2-like systems, the plaintext is used as the seed to a pseudorandom number generator (and in the analysis it's modeled as a random oracle) that's used to generate the noise for the encryption. This introduces extra looseness into the QROM proof, and extra complexity for side-channel protections. But it ought to remove the need for the plaintext confirmation hash: in some sense the ciphertext is already a hash of the plaintext. Likewise, in the same way that schemes with a plaintext confirmation hash may not need implicit rejection (in the QROM, under certain conditions: https://eprint.iacr.org/2019/590.pdf, Theorem 4), intuitively I expect these LWE-based systems not to need implicit rejection either.

I've waved my hands a lot in the above ("intuitively" etc) to indicate that one must actually go through the analysis to be confident. I did a QROM analysis for ThreeBears when I submitted it, and I would expect it to work also for Frodo and likely Kyber, SABER etc, but at least for Kyber and SABER there's some fiddling to do because of the rounding steps. But this is what was behind my confidence when I wrote that implicit vs explicit rejection probably doesn't make a real difference in Kyber's security.

On the other hand, Classic McEliece and (at least some flavors of) NTRU don't use this structure. Instead, the plaintext is a low-Hamming-weight vector that's recovered by decryption, and a hash function is not necessarily required before multiplication by the public key (whether that's f/g in NTRU or a parity matrix in Classic McEliece). This structure gives a tighter QROM reduction and a simpler variant of the FO transform. But the ciphertext really isn't in any sense a cryptographic hash of the plaintext, and instead it is somewhat malleable. This means that you really do need either implicit rejection or plaintext confirmation to retain CCA security.

Anyway, I hope that explanation made sense and is useful to other people.

Regards,

— Mike

--

From: D. J. Bernstein <djb@cr.yp.to> via pqc-forum@list.nist.gov
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Tuesday, December 20, 2022 08:26:18 AM ET
Attachments: smime.p7m

From a designer's perspective, it's useful to realize that the potential benefit being ascribed to derandomization here——protecting against chosen-ciphertext attacks without plaintext confirmation and without implicit rejection——comes from a PKE feature that is also easy to achieve for deterministic PKEs.

The attack/defense survey inside https://cr.yp.to/papers.html#ntrw splits this into two features:

* "Limited ciphertext space": e.g., rounding ciphertexts.
* "Limited plaintext space": e.g., taking (m,r) = H(256-bit seed).

Section 3.14 explains that derandomization limits the plaintext space but also incurs security risks, and explains how to limit the plaintext space without incurring those risks; this is a standard generic approach that straightforwardly applies to deterministic NTRU, McEliece, etc.

In other words, randomized PKEs force a tradeoff between the last two rows in Table 3.1, while deterministic PKEs can achieve both.

An important caveat is that, with or without derandomization, almost all options need cryptanalysis. The only cases where further cryptanalysis isn't required are where there are tight proofs from assumptions that have already been adequately cryptanalyzed; very few options provide such proofs.

This is why I recommend starting with a deterministic PKE and applying implicit rejection. This gives, assuming merely OW-CPA, a very tight ROM IND-CCA2 proof and a reasonably tight QROM IND-CCA2 proof (just sqrt(eps), no q or d loss factors), although cryptanalysis of IND-CCA2 beyond QROM IND-CCA2 is still required.

To be clear, this isn't a recommendation against _also_ limiting the ciphertext space and/or limiting the plaintext space. I simply wouldn't rely on those extra defenses as substitutes for implicit rejection. A KEM with those defenses but without implicit rejection could be much weaker than the underlying PKE.

Christian Majenz writes:
> We can actually back up Mike’s intuition. Our recent work
> https://eprint.iacr.org/2022/365 <https://eprint.iacr.org/2022/365>
> (Asiacrypt 2022) implies that for such (NewHope v2-like) KEMs explicit
> rejection without plaintext confirmation is as secure as implicit
> rejection from a provable security perspective in the QROM*.

To clarify, by "as secure" you mean that a polynomial-time QROM IND-CCA2 attack against the KEM, with implicit or explicit rejection, implies a polynomial-time attack against the PKE?

Many readers will expect "as secure" to be saying that the KEMs match the PKE in security level. The available evidence says that security levels can vary, sometimes do vary, and perhaps much more often vary, between OW-CPA of a randomized PKE, IND-CPA of the same PKE, IND-CCA2 or QROM IND-CCA2 or ROM IND-CCA2 of a KEM built from the derandomized PKE plus implicit rejection, and the same with explicit rejection:

* Almost all of the available theorems, including the main 2022/365 theorems, allow QROM IND-CCA2 security of the KEM to be weaker than the PKE by a factor q or more. This is true even for ROM IND-CCA2, if one starts with OW-CPA rather than IND-CPA. Public cryptanalysis looks much more at OW-CPA than at IND-CPA.

* https://cr.yp.to/papers.html#footloose constructs cryptosystem examples, including a reasonable ElGamal-type example, for which derandomization damages security against non-quantum attacks by a factor q (i.e., ~100 bits of security for large-scale attacks today) compared to OW-CPA of the original cryptosystem.

* There could be quantum attacks doing even more damage. This needs cryptanalysis.

* There could be similar attacks within the GAM/LPR family of lattice-based cryptosystems. This needs cryptanalysis.

* The claim that "the FrodoKEM parameter sets comfortably match their target security levels with a large margin" recently collapsed; see https://cr.yp.to/papers.html#lprrr, specifically Section 6. This is a case of a high-profile KEM having much lower security than the PKE against known attacks, exactly because of security damage done by derandomization.

I agree that intuitively the hashing inside derandomization creates an annoyance for the chosen-ciphertext attacker: it's very easy to modify (s,e,As+e) for small secret (s,e) and public (A,As+e) into a related tuple; requiring, e.g., s = H(e) stops the easy modifications. But does it stop, say, time-2^64 modifications? As far as I know, there are no attack papers studying the security level of this problem, never mind the more complicated versions that show up in attacking cryptosystems.

——D. J. Bernstein

From: Andreas Hülsing <ietf@huelsing.net> via pqc-forum@list.nist.gov
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Tuesday, December 20, 2022 10:29:40 AM ET

Dear Dan,

I am sorry, but I fail to see how this relates to the question if a dedicated side-channel protection for the secret PRF key for Kyber is necessary.

Our mail said, from a provable security perspective, the bounds are the same for explicit and implicit rejection for Kyber. This suggests that there is no problem with falling into explicit rejection via a side-channel attack.
Sure, proofs are not to be considered as a 100% guarantee. The given proofs are not tight, there is clearly a chance that there may be a flaw in any of these proofs as these are hand written and human verified, etc. pp..
This is in my eyes similar to arguing that cryptanalysis did not find an attack. It is a good indication but everything but a guarantee.

Cheers,

Andreas

On 20-12-2022 14:25, D. J. Bernstein wrote:
> From a designer's perspective, it's useful to realize that the potential
> benefit being ascribed to derandomization here——protecting against
> chosen-ciphertext attacks without plaintext confirmation and without
> implicit rejection——comes from a PKE feature that is also easy to
> achieve for deterministic PKEs.
>

> The attack/defense survey inside https://gcc02.safelinks.protection.outlook.com/?url=https%3A%2F%2Fcr.yp.to%2Fpapers.html%23ntrw&data=05%7C01%7Cyi-kai.liu%40nist.gov%7Cc2041f54f267468d336a08dae29f01be%7C2ab5d82fd8fa4797a93e054655c61dec%7C1%7C0%7C638071469806284783%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=rqvPUzGitygHfoeBkkBg6E1%2FS7MvVkYrSZidsVGW8uI%3D&reserved=0
> splits this into two features:
>
>   * "Limited ciphertext space": e.g., rounding ciphertexts.
>   * "Limited plaintext space": e.g., taking (m,r) = H(256-bit seed).
>
> Section 3.14 explains that derandomization limits the plaintext space
> but also incurs security risks, and explains how to limit the plaintext
> space without incurring those risks; this is a standard generic approach
> that straightforwardly applies to deterministic NTRU, McEliece, etc.
>
> In other words, randomized PKEs force a tradeoff between the last two
> rows in Table 3.1, while deterministic PKEs can achieve both.
>
> An important caveat is that, with or without derandomization, almost all
> options need cryptanalysis. The only cases where further cryptanalysis
> isn't required are where there are tight proofs from assumptions that
> have already been adequately cryptanalyzed; very few options provide
> such proofs.
>
> This is why I recommend starting with a deterministic PKE and applying
> implicit rejection. This gives, assuming merely OW-CPA, a very tight
> ROM IND-CCA2 proof and a reasonably tight QROM IND-CCA2 proof (just
> sqrt(eps), no q or d loss factors), although cryptanalysis of IND-CCA2
> beyond QROM IND-CCA2 is still required.
>
> To be clear, this isn't a recommendation against _also_ limiting the
> ciphertext space and/or limiting the plaintext space. I simply wouldn't
> rely on those extra defenses as substitutes for implicit rejection. A
> KEM with those defenses but without implicit rejection could be much
> weaker than the underlying PKE.
>

> Christian Majenz writes:
>> We can actually back up Mike's intuition. Our recent work
>> https://gcc02.safelinks.protection.outlook.com/?
url=https%3A%2F%2Feprint.iacr.org%2F2022%2F365&data=05%7C01%7Cyi-
kai.liu%40nist.gov%7Cc2041f54f267468d336a08dae29f01be%7C2ab5d82fd8fa4797a93e054655c61
dec%7C1%7C0%7C638071469806284783%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoi
V2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=PvrMht4sKS58IEGd%2Bl02jd
62ZFlstz4vMWz8YMOJngA%3D&reserved=0 <https://gcc02.safelinks.protection.outlook.com/?
url=https%3A%2F%2Feprint.iacr.org%2F2022%2F365&data=05%7C01%7Cyi-
kai.liu%40nist.gov%7Cc2041f54f267468d336a08dae29f01be%7C2ab5d82fd8fa4797a93e054655c61
dec%7C1%7C0%7C638071469806284783%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoi
V2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=PvrMht4sKS58IEGd%2Bl02jd
62ZFlstz4vMWz8YMOJngA%3D&reserved=0>
>> (Asiacrypt 2022) implies that for such (NewHope v2-like) KEMs explicit
>> rejection without plaintext confirmation is as secure as implicit
>> rejection from a provable security perspective in the QROM*.
> To clarify, by "as secure" you mean that a polynomial-time QROM IND-CCA2
> attack against the KEM, with implicit or explicit rejection, implies a
> polynomial-time attack against the PKE?
>
> Many readers will expect "as secure" to be saying that the KEMs match
> the PKE in security level. The available evidence says that security
> levels can vary, sometimes do vary, and perhaps much more often vary,
> between OW-CPA of a randomized PKE, IND-CPA of the same PKE, IND-CCA2 or
> QROM IND-CCA2 or ROM IND-CCA2 of a KEM built from the derandomized PKE
> plus implicit rejection, and the same with explicit rejection:
>
>   * Almost all of the available theorems, including the main 2022/365
>   theorems, allow QROM IND-CCA2 security of the KEM to be weaker than
>   the PKE by a factor q or more. This is true even for ROM IND-CCA2,
>   if one starts with OW-CPA rather than IND-CPA. Public cryptanalysis
>   looks much more at OW-CPA than at IND-CPA.
>

> * https://gcc02.safelinks.protection.outlook.com/?
url=https%3A%2F%2Fcr.yp.to%2Fpapers.html%23footloose&data=05%7C01%7Cyi-
kai.liu%40nist.gov%7Cc2041f54f267468d336a08dae29f01be%7C2ab5d82fd8fa4797a93e054655c61
dec%7C1%7C0%7C638071469806284783%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoi
V2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=89DoRfw5pL2ARGspGwf5YylQ
b1%2BNm2prorxaOrdrvCY%3D&reserved=0 constructs cryptosystem
> examples, including a reasonable ElGamal-type example, for which
> derandomization damages security against non-quantum attacks by a
> factor q (i.e., ~100 bits of security for large-scale attacks
> today) compared to OW-CPA of the original cryptosystem.
>
> * There could be quantum attacks doing even more damage. This needs
> cryptanalysis.
>
> * There could be similar attacks within the GAM/LPR family of
> lattice-based cryptosystems. This needs cryptanalysis.
>
> * The claim that "the FrodoKEM parameter sets comfortably match their
> target security levels with a large margin" recently collapsed; see
> https://gcc02.safelinks.protection.outlook.com/?
url=https%3A%2F%2Fcr.yp.to%2Fpapers.html%23lprrr&data=05%7C01%7Cyi-
kai.liu%40nist.gov%7Cc2041f54f267468d336a08dae29f01be%7C2ab5d82fd8fa4797a93e054655c61
dec%7C1%7C0%7C638071469806284783%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoi
V2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=heNM8ENQ3UQNKPLpzpD%2F6U
4×4mOIXN0m%2B8NeFw%2Fbkoc%3D&reserved=0, specifically Section 6. This is
> a case of a high-profile KEM having much lower security than the
> PKE against known attacks, exactly because of security damage done
> by derandomization.
>
> I agree that intuitively the hashing inside derandomization creates an
> annoyance for the chosen-ciphertext attacker: it's very easy to modify
> (s,e,As+e) for small secret (s,e) and public (A,As+e) into a related
> tuple; requiring, e.g., s = H(e) stops the easy modifications. But does
> it stop, say, time-2^64 modifications? As far as I know, there are no
> attack papers studying the security level of this problem, never mind
> the more complicated versions that show up in attacking cryptosystems.
>
> ——D. J. Bernstein

>

--

From: Christian Majenz <christian.majenz@gmail.com> via pqc-forum@list.nist.gov
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Tuesday, December 20, 2022 11:12:41 AM ET

> On 20 Dec 2022, at 14:25, D. J. Bernstein <djb@cr.yp.to> wrote:
>
> To clarify, by "as secure" you mean that a polynomial-time QROM IND-CCA2
> attack against the KEM, with implicit or explicit rejection, implies a
> polynomial-time attack against the PKE?
>
> Many readers will expect "as secure" to be saying that the KEMs match
> the PKE in security level.

We mean that for KEMs that start from a randomized PKE, the best known security reductions for the implicit reject variant and the explicit reject variant, have the same loss (up to a difference in additivie loss terms, as mentioned). This holds when assuming OW-CPA for both variants, and when assuming IND-CPA for both variants.

Best,

Chris

~~~~

Christian Majenz

Assistant Professor

DTU Compute

chmaj@dtu.dk

Website: https://www.christianmajenz.info
~~~~

From: D. J. Bernstein <djb@cr.yp.to> via pqc-forum@list.nist.gov
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Implicit Rejection in Kyber
Date: Tuesday, December 20, 2022 11:58:30 AM ET
Attachments: smime.p7m

Andreas Hülsing writes:
> Our mail said, from a provable security perspective, the bounds are the same
> for explicit and implicit rejection for Kyber. This suggests that there is
> no problem with falling into explicit rejection via a side-channel attack.

I'll substitute numbers for the actual formulas to clarify my concern
here. When the proofs available merely say

   SecLevel(ImplicitRejectKEM) ⩾ SecLevel(PKE) - 200

and

   SecLevel(ExplicitRejectKEM) ⩾ SecLevel(PKE) - 200

they do not provide a logical basis for the following equations:

   SecLevel(ExplicitRejectKEM) = SecLevel(ImplicitRejectKEM) = SecLevel(PKE).

I understood "as secure as implicit rejection" to be indicating at least
the first equation, and in context I'd expect many people to understand
it as also indicating the second.

——D. J. Bernstein