Hi all,

This isn’t an attack on NewHope, just another note on how bad it might be if you reuse private keys in the CPA version. It came out of discussion with Mark Marson and Mélissa Rossi. Sorry if this has been discussed before, but I thought it was interesting.

Let Alice’s public key be \((A, A_s+e)\), and Bob’s ciphertext be \((B, C)\), so that Alice is extracting a key based on C-Bs.

Suppose that Bob chooses \(B\) to be a zero divisor. For example, in the NTT domain, \(b\) might be zero in all coefficients except one. Then there are only \(q=12289\) different possible keys that Alice can extract.

If for some reason Alice confirms the key first, by sending \(eg\ hash(key)\), then Bob can just check all \(12289\) keys to recover that component of NTT(s). He could even do this for \(eg\ 3\) nonzero coefficients at a time, at a cost of \(12289^3\) work per coefficient, which would mean 340-ish chosen messages to recover the key. If the key confirmation is just \(hash(key)\), he might be able to accelerate this with a rainbow table. This is stronger than other known attacks, and it’s presumably possible to cut that down a little further by finishing with a lattice reduction or a combinatorial attack.

If Bob confirms the key first with SHA (which is the sane way to do it), then the attack is weaker than other known attacks, since it requires about \(1024*12289/2\) chosen messages. But if Bob confirms first with a polynomial MAC (eg \(poly1305\) or GCM), and if there is some large-ish data earlier in the key exchange that Bob has control over, then Bob can compute a \(12289\)-block message that verifies will all \(12289\) possible keys. Then after Alice confirms (or shares some function of the decrypted message with Bob) he can still mount the attack. So this attack wouldn’t work efficiently on TLS 1.3, but it might work with some custom protocol.

This is a good reason in general that you should confirm keys with SHA and not with AEAD (or with the FO transform), and preferably as soon as possible.

It’s easy to come up with countermeasures to this, such as rejecting capsules with too many coefficients equal to 0 (or \(q\)), but of course the only real countermeasure is don’t reuse keys in CPA mode.

Cheers,
— Mike
It’s easy to come up with countermeasures to this, such as rejecting capsules with too many coefficients equal to 0 (or q)

The PQCrypto 2017 Gong–Zhao paper that introduced zero-divisor attacks also put some effort into trying to obfuscate the attacks in a way that avoids some simple countermeasures, although it’s not clear how strong the obfuscation is.

> the only real countermeasure is don’t reuse keys in CPA mode.

That has a more obvious effect, yes. Even more effective is to avoid standardizing any sort of "CPA mode" in the first place.

Question for the people who are proposing both IND-CPA and IND-CCA2 modes for lattice systems: Are there any publicly verifiable examples of applications where the extra cost of IND-CCA2 security is a significant part of the end user’s total costs?

---Dan
> the only real countermeasure is don’t reuse keys in CPA mode.

That has a more obvious effect, yes. Even more effective is to avoid standardizing any sort of "CPA mode" in the first place.

Question for the people who are proposing both IND-CPA and IND-CCA2 modes for lattice systems: Are there any publicly verifiable examples of applications where the extra cost of IND-CCA2 security is a significant part of the end user’s total costs?

One possible example is given in Section 4.1 of [https://eprint.iacr.org/2018/1037](https://eprint.iacr.org/2018/1037).

It introduces "Continuous Key Agreement" (CKA), which abstracts the "double ratchet" mechanism of the Signal messaging protocol. In this context, CKA is run over an authenticated channel (provided by AEAD), so there's no need for additional authentication or validity checks. In other words, CKA aims for security against passive attacks.

The paper shows how to obtain CKA generically using any CPA-secure KEM. It then recalls Signal's optimization that saves about 2x in communication for the ElGamal KEM: a single group element plays two roles, first as a ciphertext (encapsulation), then as the sender's next public key. (See Figure 4.)

Finally, it shows that an analogous optimization is possible for CPA-secure LWE-based KEMs, thanks to their "noisy key agreement" properties.

Might the same kind of optimization be available for CCA-secure LWE-based KEMs? It's quite plausible---they include all the components of the simpler CPA-secure ones. But they also do extra (and unnecessary in this context) work of re-encrypting to check ciphertext validity. I will leave it to others to say how significant that extra work is.

In any case, this application seems like another nice advantage of the "noisy key agreement" feature of (Ring/Module-)LWE proposals.

Sincerely yours in cryptography,
Chris
I wrote:
> Question for the people who are proposing both IND-CPA and IND-CCA2
> modes for lattice systems: Are there any publicly verifiable examples
> of applications where the extra cost of IND-CCA2 security is a
> significant part of the end user's total costs?

In response, somebody advertises an application of Ring-LWE that might or might not be within scope for this standardization project, and then correctly observes that this doesn't answer my question about comparing the user's total costs to the cost of CCA2 security. ("I will leave it to others to say how significant that extra work is.")

Regarding the question at hand, let me add some further comments and then a data point regarding costs of a major application.

I wrote in [https://blog.cr.yp.to/20161030-pqnist.html](https://blog.cr.yp.to/20161030-pqnist.html) that "NIST should explicitly allow non-CCA2-secure single-message KEMs such as New Hope", but my support for this was explicitly conditional upon cost issues:

> I prefer the simplicity of using pure encryption ... This requires
> multiple-message support and CCA2 security, but my current impression
> is that this robustness has only minor costs, and I wouldn't be
> surprised if the New Hope team decides to move in this direction.
> However, if they instead decide that CCA2 security is too expensive,
> they shouldn't be rejected for targeting TLS!

New Hope did in fact decide to add support for CCA2 security. I don't see where the New Hope submission argues that the historical CPA options should be provided to users. On the contrary, the submission says

NewHope-CCA-KEM's extremely fast performance means that the cost of this re-encryption is still quite small.

Specifically, the submission reports 220864 total Haswell cycles for NH-1024-CCA-KEM decapsulation. It's hard to see how this can be a significant cost problem compared to receiving 2208-byte ciphertexts.

I see IND-CPA and IND-CCA2 options in two other round-2 lattice submissions (LAC and Round5), plus Three Bears if that's counted as a lattice submission. I don't see where any of these submissions argue that these options should be provided to users. (Round5 cites NIST _allowing_ IND-CPA options, but that isn't the question.)

Meanwhile five round-2 lattice submissions---Frodo, Kyber, NTRU, NTRU Prime, and Saber---provide only IND-CCA2 options. (For Saber I didn't find this clearly stated in the documentation, but I see KATs only for the IND-CCA2 options.) Two of these, Kyber and (of course) NTRU Prime, state rationales for _not_ providing IND-CPA options to users:

[Kyber:] Kyber is defined as an IND-CCA2 secure KEM only. For many applications ... active security is mandatory. However, also in use cases (like key exchange in TLS) that do not strictly speaking require active security, using an actively secure KEM has advantages.
Most notably ... Furthermore ... As a conclusion, we believe that the overhead of providing CCA security is not large enough to justify saving it and making the scheme less robust.

[NTRU Prime:] It is possible to save time, especially in decapsulation, by abandoning protection against chosen-ciphertext attacks. This submission intentionally avoids providing any such options. It is not clear that the speedup is relevant to users ... whereas there is a clear risk that providing options vulnerable to chosen-ciphertext attacks will lead to deployment of those options in scenarios that turn out to allow such attacks.

Obviously we can all provide IND-CPA options with faster decapsulation, but surely this should be backed by

* evidence that the speedup matters for some applications and
* an argument that the speedup outweighs the risks.

Otherwise we can and should focus on the IND-CCA2 options, simplifying analyses, benchmarks, comparisons, usage, etc.

Google, which had previously experimented with the old non-CCA2-secure version of New Hope, recently started a new experiment with something CCA2-secure (namely NTRU-HRSS), and stated that this was intentional:

CCA2-security is worthwhile, even though TLS can do without. ... 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.

Source: [https://www.imperialviolet.org/2018/12/12/cecpq2.html](https://www.imperialviolet.org/2018/12/12/cecpq2.html) Extra speed was mentioned as a bonus but evidently didn’t override the CCA2 security goal.

Finally, some data regarding the costs of cryptography in context:

[https://blog.cloudflare.com/how-expensive-is-crypto-anyway/](https://blog.cloudflare.com/how-expensive-is-crypto-anyway/)

says "Cloudflare is the largest provider of TLS on the planet" and reports measurements showing that a Cloudflare server spent "just 1.8% of the CPU time" on TLS. What’s even more striking is that X25519 consumed just 0.06% of the server’s CPU time, even though 30% of the TLS connections exchanged keys with X25519. Some extrapolations:

* X25519 would have consumed just 0.2% (1/500) of the CPU time if it had been used for 100% of the connections.

* Compared to X25519, lattice systems are much bigger but typically use CPU time within a factor 2 (including reencryption), so the CPU time consumed by these systems would again be negligible.

Of course it’s possible that other applications are different. This brings me to the original question regarding lattice submissions: Are there any publicly verifiable examples of applications where the extra cost of IND-CCA2 security is a significant part of the end user’s total costs?

---Dan

--

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. Visit this group at [https://groups.google.com/a/list.nist.gov/group/pqc-forum/](https://groups.google.com/a/list.nist.gov/group/pqc-forum/).
Performance between CCA2 and CPA is probably not an issue on high-end desktop CPUs but for sure on small (e.g. Cortex-M) chips which are used in smart cards and small IOT devices. In smart card application you have the additional burden to make the crypto side channel and fault resistant. This is highly required for CCA2 long-term keys but not so much for CPA ephemeral keys.

BR,
Rainer

> On Apr 18, 2019, at 7:17 PM, D. J. Bernstein <djb@cr.yp.to> wrote:
> 
> I wrote:
> >> Question for the people who are proposing both IND-CPA and IND-CCA2
> >> modes for lattice systems: Are there any publicly verifiable examples
> >> of applications where the extra cost of IND-CCA2 security is a
> >> significant part of the end user's total costs?
> >
> > In response, somebody advertises an application of Ring-LWE that might
> > or might not be within scope for this standardization project, and
> > then correctly observes that this doesn't answer my question about
> > comparing the user's total costs to the cost of CCA2 security. ("I
> > will leave it to others to say how significant that extra work is.")
> >
> > Regarding the question at hand, let me add some further comments and
> > then a data point regarding costs of a major application.
> >
> > I wrote in
> > [https://blog.cr.yp.to/20161030-pqnist.html](https://blog.cr.yp.to/20161030-pqnist.html) that "NIST should explicitly allow non-CCA2-secure single-message KEMs such as New Hope", but my support for this was explicitly conditional upon cost issues:
> >
> > I prefer the simplicity of using pure encryption ... This requires
> > multiple-message support and CCA2 security, but my current impression
> > is that this robustness has only minor costs, and I wouldn't be
> > surprised if the New Hope team decides to move in this direction.
> > However, if they instead decide that CCA2 security is too expensive,
> > they shouldn't be rejected for targeting TLS!
> >
> > New Hope did in fact decide to add support for CCA2 security. I don't
> > see where the New Hope submission argues that the historical CPA
> > options should be provided to users. On the contrary, the submission
Rainer Urian writes:
> Performance between CCA2 and CPA is probably not an issue on high-end
> desktop CPUs but for sure on small (e.g. Cortex-M) chips which are
> used in smart cards and small IOT devices.

The question regarding lattice submissions was "Are there any publicly verifiable examples of applications where the extra cost of IND-CCA2 security is a significant part of the end user's total costs?"

Here are three reasons that pointing generically at IoT devices doesn't answer the question:

* doesn't provide a reason to think that the total cost of lattice crypto is significant compared to the total application cost;

* doesn't provide a reason to think that the cost is mainly from decapsulation time rather than from communication; and

* doesn't give us any publicly verifiable numbers.

Of course a tiny device doesn't have a quad-core 3GHz Intel CPU, but it also doesn't have a 100Mbps Ethernet connection. The numbers from


(caveat: this is already ten years old!) say that receiving a byte on an 8-bit MICAz or a 16-bit TelosB costs as much energy as, respectively, 1500 cycles of computation or 5400 cycles of computation. I'd guess that modern 32-bit Cortex-M devices have even larger ratios (and obviously they also do more per cycle), since better manufacturing rewards computation much more than communication. See, e.g., the Intel data reviewed on the top of page 46 of the NTRU Prime submission.

> In smart card application you have the additional burden to make the
> crypto side channel and fault resistant.
> This is highly required for CCA2 long-term keys but not so much for
> CPA ephemeral keys.

An application is using an IND-CCA2 lattice system. You're asking the application to switch keys as frequently as possible, to help resist physical attacks. Sounds reasonable. However, I don't see how this is relevant to the question I asked.

---Dan
Hi Dan,

> On Apr 18, 2019, at 2:32 PM, D. J. Bernstein <djb@cr.yp.to> wrote:
> 
> > The question regarding lattice submissions was "Are there any publicly
> > verifiable examples of applications where the extra cost of IND-CCA2
> > security is a significant part of the end user's total costs?"

I don’t want to get involved in a debate about what level of difference is “significant”. But consider that Round5 CCA and CPA versions have different parameters due to failure rate, which impacts their bandwidth in addition to CPU and memory consumption. For example, R5ND(1,3,5)KEM_5d have bandwidth (994,1639,2035) bytes, but R5ND(1,3,5)PKE_5d have bandwidth (1097,1730,2279) — a difference of {10%,6%,12%}.

A similar tradeoff holds for ThreeBears, but instead of being smaller, the CPA parameters are more secure against lattice attacks, going from (154,235,314) bits core-sieve to (168,262,351). As with Round5, this comes at the cost of a higher failure rate (< 2^-50 instead of cryptographically negligible).

Again, a similar situation holds for BIKE, though of course that’s not a lattice scheme: it’s ring-LPN instead of LWE.

On the subject of performance measurements in general, I think that to make a fair performance comparison between the different KEMs, we should primarily consider the CCA versions. All the remaining submissions have one, and it’s certain to be a major use case. Perhaps the following criteria would make sense for a first cut at a fair performance evaluation, at least for the lattice schemes:

* Consider the performance of the IND-CCA version of the primary recommendation at each security level. Round5 doesn’t list a primary recommendation, but probably the most interesting is the most aggressive one, R5ND(1,3,5)_PKE_5d.

* Measure on Haswell (HT/TB off), Cortex-A53 (or maybe A57 or A76), and Cortex-M4 on the STM Discovery board.

* Use code which is constant-time with respect to secrets, to the extent that would be required for CCA deployment in a network-facing part. Or else benchmark decryption failures instead of successes, to measure the impact of error correction routines and of ThreeBears’ optional implicit rejection code (if it survives to the next round it will adopt the state of the art CCA transform, which is reasonably likely to use implicit rejection).

* For schemes that have a SHAKE version and an AES+SHA2 version, use the SHAKE version. It’s like CPA vs CCA: every scheme benefits from changing SHAKE to hardware-accelerated AES, so let’s start with the SHAKE version that everyone included.

* Most schemes generate their private key from a seed. Make sure to normalize how much the implementations cache — is it just the seed, the seed and private noise, or the whole ring element / matrix (if present)?
* Different implementations have wildly different code and memory sizes. It’s probably worth controlling for this by at least not manually unrolling all loops (or doing it everywhere).

* Graph the performance (bandwidth, cycles, code size, memory, energy etc) on each platform against estimated security according to your favorite estimator(s).

Note that the estimator you use matters, but it might only affect the relative position of Round5. I’m basing this on “Estimate all the LWE” on parameters from the first round (I haven’t reanalyzed them), where the security with enumeration-like estimators is almost a smooth monotone function of the security with sieve-like estimators. The exceptions are Round2, NTRU Prime, PapaBear and FireSaber, but only Round2 is drastically off curve. I think at least for the first three, the issue is that hybrid attacks affect the enumeration estimates but not the sieve ones.

Maybe it would be best to ask authors to make a SUPERCOP submission including at least one version with the above criteria, once the forum can agree on the details?

Of course, you could always measure other scenarios or other platforms, but the above is easiest.

Thoughts?
— Mike

--

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.
Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.
a concrete example could be a post quantum ICAO passport. Current ICAO passports use a PAKE protocol (i.e. PACE) which uses an ephemeral ECDH to establish a secure channel.

Contactless communication speed is meanwhile above 6Mbps. So we can assume that a PQ passport will have a communication speed above that. Reading out passport data, say, 30kb, can be done in 50ms.

On the other hand, a masked Cortex M4 implementation of NewHope takes about 25.000.000 cycles for CCA2 and 500.000 cycles for CPA. With a 100 Mhz clock, the CCA2 would need 250ms, the CPA only 5ms.

Viele Grüße / Best regards,
Rainer

> On 18. Apr 2019, at 23:32, D. J. Bernstein <djb@cr.yp.to> wrote:
> >
> > Rainer Urian writes:
> >> Performance between CCA2 and CPA is probably not an issue on high-end
> >> desktop CPUs but for sure on small (e.g. Cortex-M) chips which are
> >> used in smart cards and small IOT devices.
> >
> > The question regarding lattice submissions was "Are there any publicly
> > verifiable examples of applications where the extra cost of IND-CCA2
> > security is a significant part of the end user's total costs?"
> >
> > Here are three reasons that pointing generically at IoT devices doesn't
> > answer the question: it
> >
> > * doesn't provide a reason to think that the total cost of lattice
> > crypto is significant compared to the total application cost;
> >
> > * doesn't provide a reason to think that the cost is mainly from
> > decapsulation time rather than from communication; and
> >
> > * doesn't give us any publicly verifiable numbers.
> >
> > Of course a tiny device doesn't have a quad-core 3GHz Intel CPU, but it
> > also doesn't have a 100Mbps Ethernet connection. The numbers from
> >
Rainer Urian writes:
> Current ICAO passports use a PAKE protocol (i.e. PACE) which uses an
> ephemeral ECDH to establish a secure channel.
> Contactless communication speed is meanwhile above 6Mbps.
> So we can assume that a PQ passport will have a communication speed above that.
> Reading out passport data, say, 30kb , can be done in 50ms.
> On the other hand, a masked Cortex M4 implementation of NewHope takes
> about
> 25.000.000 cycles for CCA2 and 500.000 cycles for CPA.
> With a 100 Mhz clock, the CCA2 would need 250ms, the CPA only 5ms.

https://travel.state.gov/content/travel/en/passports/requirements/fees.html
says that renewing a U.S. passport costs $110. You're talking about a CPU that costs only a small percentage of this.

(For intensely competitive markets, one can try to argue that small cost differences matter. For U.S. passports, this agency doesn't have any competitors, aside from occasional black-market competitors that presumably have little effect on the overall economics. Also, the cost of a passport is only a small fraction of the cost of international travel, so there's no reason to think that small changes in passport fees will noticeably affect the number of passports obtained.)

As for time, even if 250ms is the best that can be done, it's hard to see why this matters in context. Aren't border interactions normally two orders of magnitude slower than this? (I'm ignoring the often vastly larger time spent waiting in line before the border interactions begin.) Advertisements for passport-reading machines typically say "within seconds", which sounds like much more time than is needed here.

Another passport-reading scenario outlined in

http://multimedia.3m.com/mws/media/1102183O/cs-passengershiptecno/technology-finalstory-aug2015.pdf

is thousands of people boarding a cruise ship "in a few hours", which suggests a budget of about 3 seconds per passport--but surely this can be, should be, and is parallelized across multiple passport readers. One of these readers costs somewhat more than $1000 according to


but surely each reader lasts for many thousands of uses, which surely means many millions of dollars of cruises, so the costs of the passport reader are almost unnoticeable. (They're still large enough to fund someone to negotiate a bulk purchase with 3M, obviously, but this is very far from indicating that the cost of IND-CCA2 matters.)

This example looks like a great illustration of the difference between looking at cryptographic costs in isolation and looking at them as part of the total costs of the application.

---Dan
Hi all,

Round5 proposes as NIST submissions a KEM that is an indcpa-kem and a PKE that builds on an indcca-kem.

The reason for having as a NIST KEM submission an indcpa-kem is because we improve both CPU and bandwidth compared with the indcca-kem. This is also what Mike pointed out in his email: “For example, R5ND\{1,3,5\}KEM_5d have bandwidth \{994,1639,2035\} bytes, but R5ND\{1,3,5\}PKE_5d have bandwidth \{1097,1730,2279\} — a difference of \{10\%,6\%,12\%\}.” We can obtain better parameters because in Round5 we optimize over a large parameter space, and we can set as targets a low enough failure rate and a high enough security level. By doing this optimization we can get to smaller key sizes.

Having as small as possible key sizes is very important. All Internet protocols will benefit of keys that are as small as possible. See for instance the text in the Round5 submission:

"An example of a protocol for which direct integration of a KEM is challenging is IKEv2 (RFC 7296). The reason is that the first message exchange in IKEv2, IKE SA INIT does not support fragmentation. If we assume Ethernet (1500 B layer 2 packets), and we use the minimal header sizes, then there is room for a 1384 B public-key/ciphertext assuming IPv4, with IPv6, this is 1364 B. Still, this misses important information that is exchanged in most real-world deployments and that further reduces the available space. Examples of such information are notification/vendor ids, a cookie that the responder could use to decide whether it might be under a DoS attack (a minimum of 12 Bytes), and an initial contact notify that tells the responder that it is the first time we are talking to it and it should clear out any stale state (8 Bytes). If NAT traversal needs to be supported, then another 56 B are required for the corresponding notify. In general, most implementations might have enough space for perhaps 1250-1300 B; smaller than that makes things easy; larger than that forces implementations to make hard decisions. All ring-based Round5 parameter sets fit in IKEv2’s IKE SA INIT."

Similarly, other protocols used, e.g., in IoT have more resource constrained links with limited data rates, small packets, and limited energy budget. Examples of those protocols are IEE 802.15.4, ZigBee, 6LoWPAN,... Many of those protocols do not have public-key solutions today because of resource constraints. The reason why in Round5 we included an IoT indcpa-kem parameter set was to try to go as small as possible and as efficient as possible so that applications relying on those protocols can also use a public-key solution. This Round5 IoT parameter set requires 736 Bytes (public-key + ciphertext). This is still a lot for many applications, but this is the smallest that we managed to get so far.

Kind regards, Oscar on behalf of the Round5 team.

On Friday, April 19, 2019 at 1:27:40 PM UTC+2, Rainer Urian wrote:

a concrete example could be a post quantum ICAO passport.
Current ICAO passports use a PAKE protocol (i.e. PACE) which uses an ephemeral ECDH to establish a secure channel.

Contactless communication speed is meanwhile above 6Mbps.
So we can assume that a PQ passport will have a communication speed above that.
Reading out passport data, say, 30kb, can be done in 50ms.

On the other hand, a masked Cortex M4 implementation of NewHope takes about 25.000.000 cycles for CCA2 and 500.000 cycles for CPA.
With a 100 Mhz clock, the CCA2 would need 250ms, the CPA only 5ms.

Viele Grüße / Best regards,
'Rainer Urian' via pqc-forum <pqc-forum@list.nist.gov> wrote:

Dear Rainer, dear all,

> a concrete example could be a post quantum ICAO passport.
> Current ICAO passports use a PAKE protocol (i.e. PACE) which uses an
> ephemeral ECDH to establish a secure channel.
> > Contactless communication speed is meanwhile above 6Mbps.
> > So we can assume that a PQ passport will have a communication speed above that.
> > Reading out passport data, say, 30kb , can be done in 50ms.
> >
> > On the other hand, a masked Cortex M4 implementation of NewHope takes
> > about
> > 25.000.000 cycles for CCA2 and 500.000 cycles for CPA.
> > With a 100 Mhz clock, the CCA2 would need 250ms, the CPA only 5ms.

What would be the reason to apply masking to a key-exchange protocol without CCA security that can only be used with ephemeral keys? My understanding is that masking helps against DPA, but DPA is only possible if a secret is used multiple times.

All the best,

Peter
On 19/04/2019 18:20, Peter Schwabe wrote:
> 'Rainer Urian' via pqc-forum <pqc-forum@list.nist.gov> wrote:
> 
> Dear Rainer, dear all,
>
>> a concrete example could be a post quantum ICAO passport.
>> Current ICAO passports use a PAKE protocol (i.e. PACE) which uses an
>> ephemeral ECDH to establish a secure channel.
>>
>> Contactless communication speed is meanwhile above 6Mbps.
>> So we can assume that a PQ passport will have a communication speed above that.
>> Reading out passport data, say, 30kb , can be done in 50ms.
>>
>> On the other hand, a masked Cortex M4 implementation of NewHope takes
>> about
>> 25.000.000 cycles for CCA2 and 500.000 cycles for CPA.
>> With a 100 Mhz clock, the CCA2 would need 250ms, the CPA only 5ms.
>
> What would be the reason to apply masking to a key-exchange protocol
> without CCA security that can only be used with ephemeral keys? My
> understanding is that masking helps against DPA, but DPA is only
> possible if a secret is used multiple times.

Horizontal side channel attacks and template attacks both work with a single trace (i.e. execution of the algorithm).

Kind regards,

-- Alessandro

--

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. 
Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.
[ quote from the Round5 submission: ]

> All ring-based Round5 parameter sets fit in IKEv2’s IKE SA INIT.

This includes the IND-CCA2 parameter sets, right? None of the Round5 parameter sets need the minor IND-CPA squeezing (whether it's 10% or 6% or 12%) to fit into this application?

I'm puzzled to see a claim that "Having as small as possible key sizes is very important", and then various text about the size limits in this application, where the bottom line is that the size limits are large enough for every proposed parameter set to fit. In the words of the Round5 submission, "smaller than that makes things easy". This is an example where being as small as possible _isn't_ important.

In general, it looks difficult to use protocol size cutoffs as an argument for the IND-CPA options in the round-2 lattice submissions. The IND-CPA-vs.-IND-CCA2 size gaps that we're talking about are very small:

* LAC: 0%, if I'm reading correctly.
* New Hope: 3% or 1.4% for ciphertexts; 0% for keys.
* Round5: reportedly 10%, 6%, 12%.
* Three Bears, if that's counted as a lattice submission: 0%.

Do we have any publicly verifiable example of a protocol that simultaneously (1) has a size cutoff, (2) can't easily change the cutoff as part of a post-quantum upgrade, and (3) has this cutoff within the tiny gaps listed above?

It occurs to me that LAC and New Hope and Three Bears actually have an incentive to _avoid_ collecting data about hard-to-change size cutoffs in real protocols. Here's why: If these cutoffs are in the same ballpark as ciphertext sizes then they'll probably fall in the much larger gaps between supported ciphertext sizes, probably forcing the application to sacrifice many bits of security. This would be a much bigger change in security level than the ~10% that Mike mentioned for IND-CPA vs. IND-CCA2 in Three Bears.

Round5 is different, since it can target many security levels. The IKEv2 example highlighted in the Round5 submission _does_ seem to force, e.g., New Hope to drop from dimension 1024 all the way down to dimension 512, which is an interesting argument against New Hope. But this example still doesn't answer the question I asked in the first place: Are there any publicly verifiable examples of applications where the extra cost of IND-CCA2 security is a significant part of the end user's total costs?

---Dan
Hi Peter,

yes, I think you are right.
One would probably implement a non-masked CPA version and a masked CCA2 version only.

Viele Grüße / Best regards,
Rainer

> On 19. Apr 2019, at 18:20, Peter Schwabe <peter@cryptojedi.org> wrote:
> 
> 'Rainer Urian' via pqc-forum <pqc-forum@list.nist.gov> wrote:
> 
> Dear Rainer, dear all,
> 
> a concrete example could be a post quantum ICAO passport.
> Current ICAO passports use a PAKE protocol (i.e. PACE) which uses an
> ephemeral ECDH to establish a secure channel.
> 
> Contactless communication speed is meanwhile above 6Mbps.
> So we can assume that a PQ passport will have a communication speed above that.
> Reading out passport data, say, 30kb, can be done in 50ms.
> 
> On the other hand, a masked Cortex M4 implementation of NewHope takes
> about
> 25.000.000 cycles for CCA2 and 500.000 cycles for CPA.
> With a 100 Mhz clock, the CCA2 would need 250ms, the CPA only 5ms.
> 
> What would be the reason to apply masking to a key-exchange protocol
> without CCA security that can only be used with ephemeral keys? My
> understanding is that masking helps against DPA, but DPA is only
> possible if a secret is used multiple times.
> 
> All the best,
> 
> Peter
The IKEv2 example highlighted in the Round5 submission _does_ seem to force, e.g., New Hope to drop from dimension 1024 all the way down to dimension 512, which is an interesting argument against New Hope.

The StrongSwan IPsec/IKE implementation has a NewHope 1024 CPA extension. Works like a charm ...

> On Apr 19, 2019, at 7:20 PM, D. J. Bernstein <djb@cr.yp.to> wrote:
> > [ quote from the Round5 submission: ]
> >> All ring-based Round5 parameter sets fit in IKEv2’s IKE SA INIT.
> >
> > This includes the IND-CCA2 parameter sets, right? None of the Round5 parameter sets need the minor IND-CPA squeezing (whether it's 10% or 6% or 12%) to fit into this application?
> >
> > I'm puzzled to see a claim that "Having as small as possible key sizes is very important", and then various text about the size limits in this application, where the bottom line is that the size limits are large enough for every proposed parameter set to fit. In the words of the Round5 submission, "smaller than that makes things easy". This is an example where being as small as possible _isn't_ important.
> >
> > In general, it looks difficult to use protocol size cutoffs as an argument for the IND-CPA options in the round-2 lattice submissions.
> >
> > iND-CPA-vs.-IND-CCA2 size gaps that we're talking about are very small:
> >
> > * LAC: 0%, if I'm reading correctly.
> > * New Hope: 3% or 1.4% for ciphertexts; 0% for keys.
> > * Round5: reportedly 10%, 6%, 12%.
> > * Three Bears, if that's counted as a lattice submission: 0%.
> >
> > Do we have any publicly verifiable example of a protocol that simultaneously (1) has a size cutoff, (2) can't easily change the cutoff as part of a post-quantum upgrade, and (3) has this cutoff within the tiny gaps listed above?
> >
Alessandro Barenghi <alessandro.barenghi@polimi.it> wrote:
> On 19/04/2019 18:20, Peter Schwabe wrote:

Dear Alessandro, dear all,

> > 'Rainer Urian' via pqc-forum <pqc-forum@list.nist.gov> wrote:
> >
> > > Dear Rainer, dear all,
> >>
> > > a concrete example could be a post quantum ICAO passport.
> > > Current ICAO passports use a PAKE protocol (i.e. PACE) which uses
> > > an ephemeral ECDH to establish a secure channel.
> >>
> > > Contactless communication speed is meanwhile above 6Mbps.
> > > So we can assume that a PQ passport will have a communication speed above that.
> > > Reading out passport data, say, 30kb, can be done in 50ms.
> >>
> > > On the other hand, a masked Cortex M4 implementation of NewHope
> >> takes about
> > > 25.000.000 cycles for CCA2 and 500.000 cycles for CPA.
> > > With a 100 Mhz clock, the CCA2 would need 250ms, the CPA only 5ms.
> >>
> > > What would be the reason to apply masking to a key-exchange protocol
> > > without CCA security that can only be used with ephemeral keys? My
> > > understanding is that masking helps against DPA, but DPA is only
> > > possible if a secret is used multiple times.
> >
> > Horizontal side channel attacks and template attacks both work with a
> > single trace (i.e. execution of the algorithm).

Yes, absolutely. But would masking stop those attacks?

All the best,

Peter

--
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.
Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.
Applying a masking countermeasure, raises the number of traces required for a template attack exponentially in the order of the masking, while providing a concrete hindrance to horizontal attacks (provided that they are taken into account when implementing the masking scheme).
In both cases, masking alone may not be sufficient to stop an attack, if it is possible for the attacker to obtain high SNR measurements of the power consumption and EM sampling of the traces of the device, but it represents a very effective component in lowering the SNR of the side channel at hand.

The typical case for a protection is a combination of masking, hiding, and possibly code morphing which in turn, is effective in stopping horizontal and template attacks.

Cheers,

Alessandro

--

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. Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.
I would like to raise something that has probably been discussed. If it has then I am unaware and apologise for time wasting.

Personally I feel like labelling side channels generally is problematic and suggests that risk analysis of each type is not really done. Perhaps that is just a side effect of it being easier to simply avoid branches etc. and label the reason as sidechannel protection and be done with it. Some may see this code as beautiful but IMO it is often horrific to read. (I have actually ripped some out of mbedtls before, obviously with much vector testing).

I feel like the cryptographic community is used to looking at any and all attacks in detail as it should and often does for a research paper or discussion.

When it affects the code however. I feel that like a product developer may decide to take measures that are outside their security risk model simply to avoid headlines of lab based attacks that don't actually apply (commercial risk model). A cryptographic library writer probably thinks he MUST and includes code that he hates because it is better than being publicly criticised and losing users/eyeballs.

Personally I think that most side channels should be dealt with in hw and don't apply in most use cases. Perhaps a smart card design should cover those risks in a special cryptographic library?

If they additionally mean that less people analyse the much more difficult to read code and that speed decreases mean less deployment of encryption.

Has the risk-benefit model of sw based side channel protections themselves been sufficiently considered, to date?

--

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. Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.
When using ElGamal-type KEM to replace the CPA-secure Diffie-Hellman in TLS, there are the following problems that may cause serious security concerns:

(1) Lazy user: as the session-key is predetermined and set by one user. If it is lazy, it may re-use the same session-key for many concurrent sessions. This has already been reported with TLS based on RSA-KEM in the past.

(2) Relatively poor randomness: again, as the session-key is set by one user, the session-key may have poor randomness, compared to Diffie-Hellman type key exchange where the session-key is cooperatively generated by the two communicating parties.

If communication cost is a serious issue, the bandwidth of NewHope-KEM can actually be reduced. The first CPA-secure KEM based on NewHope, named AKCN4:1, was developed in https://arxiv.org/abs/1611.06150 since Nov 2016. On the same parameters (same security, same error probability, etc), AKCN4:1 has smaller bandwidth than NewHope-KEM (actually AKCN4:1 and NewHope-KEM are extremely similar in mathematical structures). To further increase session-key size and decrease bandwidth and error probability simultaneously, in https://arxiv.org/abs/1611.06150, we developed AKCN-E8 (with encoding and decoding in E8 instead of D4). On the same parameters of NewHope-KEM, AKCN-E8 can have size-doubled session-key (e.g., 512-bit session-key), smaller bandwidth and/or error probability.

Best regards
Yunlei
Dear Alessandro, dear all,

Alessandro Barenghi <alessandr...@polimi.it> wrote:

Applying a masking countermeasure, raises the number of traces required for a template attack exponentially in the order of the masking, while providing a concrete hindrance to horizontal attacks (provided that they are taken into account when implementing the masking scheme).

In such a single-trace scenario, the argument of increasing the number of required traces shouldn't really matter. Masking might protect against horizontal (template-based or standard) DPA, like the one done in https://ia.cr/2018/687. This then depends on the scheme, what is masked, etc., but I suspect that these are the things you meant with "taken into account". But I don't think that such a horizontal DPA is even possible for, e.g., NewHope. Also, if you can recover each share individually, then masking doesn't help at all. We did that in a previous paper https://ia.cr/2017/594. So in my opinion, it might be better to spend the resources used for protection on better hiding instead of masking.

Cheers,
Peter

--
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. Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.
Dear Alessandro, dear all,

Alessandro Barenghi <alessandro.barenghi@polimi.it> wrote:

Applying a masking countermeasure, raises the number of traces required for a template attack exponentially in the order of the masking, while providing a concrete hindrance to horizontal attacks. (provided that they are taken into account when implementing the masking scheme).

In such a single-trace scenario, the argument of increasing the number of required traces shouldn’t really matter.

If it is possible for an attacker building templates to have control on the RNG, and derive templates for the ephemeral key bits, it still has some meaning. I know this is platform dependent, but, in case of a uC/CPU running the algorithm it shouldn’t be too much of a problem to derive templates with known outputs from the RNG on the attacker-controlled clone device.

Masking
might protect against horizontal (template-based or standard) DPA,
like the one done in https://ia.cr/2018/687. This then depends on the scheme, what is masked, etc., but I suspect that these are the things you meant with "taken into account".

Precisely

But I don’t think that such a horizontal DPA is even possible for, e.g., NewHope.
Also, if you can recover each share individually, then masking doesn't help at all. We did that in a previous paper https://ia.cr/2017/594
Yes, if you are able to recover the shares individually, then performing a horizontal attack is just a matter of recombining them properly before

> So in my opinion, it might be better to spend the resources used for
> protection on better hiding instead of masking.

Good hiding helps a lot against horizontal attacks and, given the intrinsic parallelism of some primitives, it may be achieved at a reasonably low performance cost. My 2 cents are that the best trade-off point between computation overhead devoted to masking and computation time devoted to hiding will probably depend on the primitive. It would be interesting to find out that the best option is "just hiding, no masking".

Cheers,

Alessandro

--
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. Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.