From: pqc-forum@list.nist.gov on behalf of Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>
Sent: Friday, February 7, 2020 6:24 AM
To: pqc-forum
Subject: [pqc-forum] ROUND 2 OFFICIAL COMMENT: ROLLO

Hello,

I don't know if this trivial observation actually violates the ROLLO-I security claims -- but I was surprised by it and I can see that it can be a problem in case of ephemeral key exchange:

If you set the ROLLO-I (ex "LAKE") public key (polynomial h) to have coefficients in {0,1}, *any* secret key will recover the shared secret from encapsulation that uses that "h".

The easiest test case is simply to zero the public key to all zero bytes; decapsulation still works. I've verified it against ROLLO-I variants in "rollo-optimized-implementation_2019-08-24.zip", which I believe to be the current official implementation.

Imagine a key exchange protocol:

1. Alice creates a keypair (pk,sk) and tries sending pk to Bob.
2. Eve intercepts the message, sets pk=000..00 (all zero bytes, h=0).
3. Bob encapsulates with this public key, sends ct to Alice.
4. Alice decrypts ct using sk (which does not actually matter).
5. The shared secrets of Alice and Bob *match*, but Eve can also read it. No need for further man-in-the-middle, a passive intercept is sufficient.

So this is a little bit like injecting 0 or 1 as a public Diffie-Hellman value; the two parties will have a trivially recoverable shared secret. Needless to say, DH implementations reject such public values, and I'd think that ROLLO-I should too.

Also note that if you XOR any polynomial with {0,1} coefficients with a public key, that is an equivalent ROLLO-I key. So any public key always has at least 2^n trivial equivalent public keys; 2^67 in case of ROLLO-I-256 -- in addition to those "universal public keys" with {0,1} coefficients. All of this is especially problematic since in ROLLO-I Bob completely controls the shared secret -- many of these trivial problems could be avoided if the public key would be included in the hash that is used to create shared secrets (this is essentially free).

Rollo-II or Rollo-III do not behave in this way.

Cheers,
- markku

Dr. Markku-Juhani O. Saarinen <mjos@pqshield.com> PQShield, Oxford UK.
--

From: pqc-forum@list.nist.gov on behalf of Gaborit <gaborit@unilim.fr>
Sent: Friday, February 7, 2020 7:31 PM
To: pqc-forum
Cc: Philippe Gaborit
Subject: Re: [pqc-forum] ROUND 2 OFFICIAL COMMENT: ROLLO

dear markku,

thank you for your remarks.

For the first case you mention, if one considers a code of length n over GF(2^m), the probability that it happens is roughly 2^(-mn), for m and n around 60, it means that it does not really happen, I guess this a similar situation to the case when for discrete log you consider a secret key which is 1, of course it can happen but the probability is so small that it is not mentioned and not taken in to account. The same analysis applies for your third remark.

For the attack you describe, you consider a kind of "weak" man in the middle attack but this kind of attack is not taken into account by the kem security model, (it is more Oscar/Eve than Eve in what you propose I think) of course if you want to be secure against the man in the middle attack you need an additional counter measure like a signature for instance but this attack is generic for all proposed schemes to the NIST standardization. If you would consider a weaker attacker, as you propose, a signature would avoid this or the counter measure you propose, but I guess typically a signature is used (as it is the case for TLS) which avoid the man in the middle attack and the "weak" man in the middle attack you propose.

Now in practice using a hash is free, as you say, and even if I do not see any threat here, it could easily be modified

best,

the ROLLO team

if you consider a key exchange you need an additional counter measure

Le 07/02/2020 à 12:24, Markku-Juhani O. Saarinen a écrit :

> Hello,
>
> I don't know if this trivial observation actually violates the ROLLO-I security claims -- but I was surprised by it and I can see that it can be a problem in case of ephemeral key exchange:

From:	pqc-forum@list.nist.gov on behalf of D. J. Bernstein <djb@cr.yp.to>
Sent:	Saturday, February 8, 2020 7:51 AM
To:	pqc-forum
Subject:	Re: [pqc-forum] ROUND 2 OFFICIAL COMMENT: ROLLO
Attachments:	signature.asc

If a user encrypts data to a public key generated by an attacker then the attacker can read the data, and can delegate, to anyone or everyone, the ability to read the data. This is true for all of the cryptosystems, so blaming it on the cryptosystem doesn't make sense.

Tweaking the key-substitution details to be specific to a particular cryptosystem---ROLLO, BIKE, etc.---does not turn this into a problem with that system. We instead focus on clear security goals such as IND-CCA2, and compose these goals with authentication of public keys so that the attacker can't substitute public keys in the first place.

> Imagine a key exchange protocol:
> 1. Alice creates a keypair (pk,sk) and tries sending pk to Bob.
> 2. Eve intercepts the message, sets pk=000..00 (all zero bytes, h=0).
> 3. Bob encapsulates with this public key, sends ct to Alice.
> 4. Alice decrypts ct using sk (which does not actually matter).
> 5. The shared secrets of Alice and Bob *match*, but Eve can also read
> it. No need for further man-in-the-middle, a passive intercept is sufficient.
> So this is a little bit like injecting 0 or 1 as a public
> Diffie-Hellman value; the two parties will have a trivially
> recoverable shared secret. Needless to say, DH implementations reject
> such public values

It is correct that _some_ DH implementations reject such public values.
However, rejecting such public values does _not_ stop man-in-the-middle
attacks: the attacker simply carries out the low-cost extra work labeled above as the "further man-in-the-middle". Real-world example:

https://www.eff.org/deeplinks/2011/09/post-mortem-iranian-diginotar-attack

We need public keys to be authenticated---but this authentication also eliminates the above argument for rejecting specific DH values, and similarly eliminates the above complaint regarding ROLLO-I.

> I don't know if this trivial observation actually violates the ROLLO-I
> security claims

The ROLLO specification says on the first page that ROLLO-I (LAKE) targets IND-CPA, that ROLLO-II (LOCKER) targets IND-CCA2, etc. Key substitution, whether phrased generically or tied artificially to a particular system, is not relevant to these security notions.

The security section in NIST's call for proposals begins by citing various applications, and says "schemes will be evaluated by the security they provide in these applications". Regarding other claimed applications, it says "claimed applications will be evaluated for their practical importance if this evaluation is necessary for deciding which algorithms to standardize." It continues by identifying IND-CCA2, IND-CPA, and EUF-CMA as evaluation targets; identifying target security levels; and, finally, saying that other properties would be desirable, including "perfect forward secrecy" and "resistance to side-channel attacks" and "resistance to multi-key attacks" and an "ill-defined" property of "resistance to misuse".

Obviously this allows complaints saying that a cryptosystem could be misused by protocol X. But such complaints almost never violate the security claims in the submissions---and this is directly relevant to further evaluation criteria, namely quality and maturity of analysis.
I think the complaints have to quote the security claims and compare themselves to those claims, in particular admitting non-violations of the security claims, rather than declaring ignorance. (For example, there's a huge difference between reusing keypairs to break an IND-CCA security claim and reusing keypairs to "break" a system that merely claimed IND-CPA security.)

Furthermore, if X isn't one of the listed applications then, by the rules, it has to be evaluated for its "practical importance". From this perspective, it's a problem if a misuse complaint is exaggerating the importance of the scenario it describes---e.g., trying to suggest that "key exchange" is broken for a particular submission, when the reality is that we're talking about a misdesigned key-exchange protocol that's breakable at low cost no matter which cryptosystem it uses.

> Also note that if you XOR any polynomial with {0,1} coefficients with
> a public key, that is an equivalent ROLLO-I key.

Authenticating public keys stops this too. Malleability of public keys is not relevant to the ROLLO security claims.

> All of this is especially problematic since in ROLLO-I Bob completely
> controls the shared secret -- many of these trivial problems could be
> avoided if the public key would be included in the hash that is used
> to create shared secrets (this is essentially free).

Every extra complication needs a rationale. This extra hashing doesn't stop man-in-the-middle attacks, and doesn't turn the protocol shown above into a secure protocol. More broadly, no justification is given here for the words "problematic" and "problems" in the above sentence.

I'm not saying that it's impossible to construct arguments for going beyond the core goal of IND-CCA2 security (or IND-CPA security for one-time systems such as ROLLO-I). On the contrary, it's not a hard exercise to construct more obscure protocols that are generically broken on top of IND-CCA2 but that can be saved by some extra hashing; does this mean that the primitive should be changed, or does it mean that the protocol should be changed?

Section 7.1 of

https://ntruprime.cr.yp.to/nist/ntruprime-20190330.pdf

summarizes various arguments for and against extra hashing. I don't find it at all obvious which way is better (short term or long term), and I don't see how anyone who has studied the arguments in the literature can be confident about this. There are tricky interactions here between different ecosystems of implementations, protocol designs, and protocol analyses, producing complicated tradeoffs between different types of robustness, verifiability, etc.

More to the point, I'm bothered by a complaint alleging that ROLLO-I doesn't meet certain security goals, when the complaint doesn't clearly define the goals, doesn't compare these to the ROLLO-I security claims, doesn't clearly state why these goals are supposed to be important, and doesn't analyze whether _any_ submission is acceptable from the same perspective. Having security goals whose definitions are unclear or unstable, just like having cryptosystems whose definitions are unclear or unstable, causes problems for evaluators and for the standardization project as a whole.

---Dan

From: pqc-forum@list.nist.gov on behalf of Markku-Juhani O. Saarinen <mjos.crypto@gmail.com>
Sent: Saturday, February 8, 2020 8:25 AM
To: pqc-forum
Subject: Re: [pqc-forum] ROUND 2 OFFICIAL COMMENT: ROLLO

Hi,

I was just pointing out that the actual shared secret can be trivially forced to a weak one in ROLLO-1 and BIKE-1 which is not the case for most CPA KEMs. An anonymous key exchange can always be MITM'd but not necessarily quite like that.

In many ways this is analogous to Diffie-Hellman with a composite, unchecked group order. Most protocol designers prefer to avoid that unnecessary risk.

Cheers
- markku

On Sat, Feb 8, 2020, 12:51 D. J. Bernstein <djb@cr.yp.to> wrote:

> If a user encrypts data to a public key generated by an attacker then
> the attacker can read the data, and can delegate, to anyone or everyone,
> the ability to read the data. This is true for all of the cryptosystems,
> so blaming it on the cryptosystem doesn't make sense.
>
> Tweaking the key-substitution details to be specific to a particular
> cryptosystem---ROLLO, BIKE, etc.---does not turn this into a problem
> with that system. We instead focus on clear security goals such as
> IND-CCA2, and compose these goals with authentication of public keys so
> that the attacker can't substitute public keys in the first place.
>
> \> Imagine a key exchange protocol:
> \> 1. Alice creates a keypair (pk,sk) and tries sending pk to Bob.
> \> 2. Eve intercepts the message, sets pk=000..00 (all zero bytes, h=0).
> \> 3. Bob encapsulates with this public key, sends ct to Alice.
> \> 4. Alice decrypts ct using sk (which does not actually matter).
> \> 5. The shared secrets of Alice and Bob *match*, but Eve can also read it. No
> \> need for further man-in-the-middle, a passive intercept is sufficient.
> \> So this is a little bit like injecting 0 or 1 as a public Diffie-Hellman value;
> \> the two parties will have a trivially recoverable shared secret. Needless to
> \> say, DH implementations reject such public values
>
> It is correct that _some_ DH implementations reject such public values.
> However, rejecting such public values does _not_ stop man-in-the-middle
> attacks: the attacker simply carries out the low-cost extra work labeled
> above as the "further man-in-the-middle". Real-world example:
>
> https://www.eff.org/deeplinks/2011/09/post-mortem-iranian-diginotar-attack
>
> We need public keys to be authenticated---but this authentication also