From:	Apon, Daniel C. (Fed)
To:	Smith-Tone, Daniel C. (Fed); Perlner, Ray A. (Fed)
Subject:	RE: Call for Assistance
Date:	Monday, June 8, 2020 10:51:22 PM

Wait – GeMSS*

From: Apon, Daniel C. (Fed)
Sent: Monday, June 8, 2020 10:51 PM
To: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>; Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
Subject: RE: Call for Assistance

GeMMS*

From: Apon, Daniel C. (Fed)
Sent: Monday, June 8, 2020 10:50 PM
To: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>; Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
Subject: RE: Call for Assistance

In all seriousness though, I'd be happy to do a separate call about this.

I'd like GeMSS to not suck as much as it does right now

From: Apon, Daniel C. (Fed)
Sent: Monday, June 8, 2020 10:33 PM
To: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>; Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
Subject: RE: Call for Assistance

Get it $#@!in' done then :-)

https://www.youtube.com/watch?v=4ilZe4m2P-w

From: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>
Sent: Monday, June 8, 2020 5:54 PM
To: Perlner, Ray A. (Fed) <ray.perlner@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: RE: Call for Assistance

Yes, you're right. I think that the game should involve a probabilistic adversary, though, and that the metric for security should be the quotient of the operations that the adversary performs and the probability that the proposed message verifies. Still, though, what I'm saying doesn't actually violate the model.

It seems to me, then, that GeMSS should definitely use three iterations instead of 4.

From: Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
Sent: Monday, June 8, 2020 4:37 PM
To: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: RE: Call for Assistance

I'm still not sure I understand concretely what attack you are talking about. As far as I can tell, in the EUF-CMA, after asking for a whole bunch of signatures, the adversary is supposed to produce one message and one signature, such that, with high probability the signature is valid and matches the message. Providing a long list of signatures and a long list of messages (like 2^117 or more of each) such that one of the signatures is likely to be a valid signature of one of the messages (the only thing like what you desribed I know how to do in less than 2^143 operations) doesn't seem to violate the game.

From: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>
Sent: Monday, June 8, 2020 3:38 PM
To: Perlner, Ray A. (Fed) <ray.perlner@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: RE: Call for Assistance

I had the same calculation as you mention in mind, but I think that the issue is more serious than your email indicates. When we want EUF-CMA security, we are asking for the strongest security notion (existential unforgeability) in the weakest model (chosen message attack… basically the adversary has everything). So, for example, one could imagine a scenario in which the adversary has access to valid signatures for an extremely large family of messages and somehow succeeds in creating a different valid signature for one of the messages they already have a signature for. This would still constitute a break in this model.

So my question is referring to the event that the adversary randomly generates strings and messages of the correct size without ever even looking up the public key. If we have signatures that are less than 256 bits long, then it will take fewer than 2^128 iterations of this process for there to exist a collision in the data generated by the adversary. The adversary never checks to see if a collision has occurred, but apparently the work required to generate such a pair is less than 2^143. As I see it, this seems to technically violate the EUF-CMA security.

To state it one more time, I think that something like GeMSS with 3 iterations in the Fiestel-Patarin construction is subject to an attack in which and adversary never looks at the key or generates a single hash, but nevertheless can generate a valid message-signature pair with work less than 2^143 violating EUF-CMA security. The caveat is that the adversary will never know it. If you take the act of checking whether a pair forms a valid message-signature pair, then the work would be just barely over 2^143.

This is the question I want opinions on. It seems to me that taking the number of iterations 3 technically violates EUF-CMA security but in a way that is not at all meaningful. I want other opinions.

Cheers!
Daniel

From: Perlner, Ray A. (Fed) <ray.perlner@nist.gov>
Sent: Monday, June 8, 2020 2:37 PM
To: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: RE: Call for Assistance

"Of course another possibility is that you consider that generating a valid pair (m,s) without any check that it is valid constitutes an attack on EUF-CMA. "

How could this happen with a signature scheme, the function for checking whether (m,s) is valid is public. No?

"I am wondering if I can get a check on my arithmetic"

Your arithmetic is correct as far as I can tell. There is a minor complication, however:

If the P function costs 2^3.5 more to compute than the hash function (i.e. 2^2 more than the hash function 3 times), then the attack can be rebalanced, by precomputing only 2^(3m/4 -1/2)=2^121 P function values and trying 2^123 messages. By my calculations, this will require 2^121 P function calls and 2^124.5 hash function calls, making the total work 2^(124.5+18)+2^(121+21.5) =2^143.5 bit operations. I was going to also make a remark about memory costs and question whether there was a Van Oorschot- Wiener type trick to reduce them, but I think we're up to enough bit operations just ignoring the memory costs, so I don't think we have to worry about that stuff, unless we think there's enough there to justify going to 2 iterations (my guess is there isn't).

Cheers,
Ray

From: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>
Sent: Friday, June 5, 2020 6:21 PM
To: internal-pqc <internal-pqc@nist.gov>
Subject: RE: Call for Assistance

Of course another possibility is that you consider that generating a valid pair (m,s) without any check that it is valid constitutes an attack on EUF-CMA. If you do, please let me know this as well.

From: Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>
Sent: Friday, June 5, 2020 5:17 PM
To: internal-pqc <internal-pqc@nist.gov>
Subject: Call for Assistance

Hello, everyone,

I am wondering if I can get a check on my arithmetic. By my calculation, I think that all of the security level I parameter sets of GeMSS could use 3 rounds of the Fiestel-Patarin construction instead of 4. I think that there is an appropriate way to point this out nicely in the second round report, but I want to make sure that I'm being reasonable. It is very easy science, but I have trouble mashing calculator buttons with my under-sized front feet.

The way the attack works is trying to form a collision between the hash and the trapdoor function. We have hash H and public key P. With no construction you would simply try to generate a pair (m,s) such that H(m)=P(s).

The Fiestel-Patarin transformation does this [instead of just computing s=P^-1(H(m))]. You set s_0=0, d_1=Tr(H(m)), where Tr(.) is truncating the number of bits you want. Then you compute (s_1,x_1)=P^-1(d_1,s_0). You store x_1 and then set d_2=Tr(H(H(m))), compute (s_2,x_2)=P^-1(d_2,s_1), and repeat this some number of times. For GeMSS this number is 4. At the end, the signature is (s_4,x_4,x_3,x_2,x_1).

So to try a collision attack (simply to violate EUF-CMA security), You generate pairs (m,s) and try to make the (in GeMSS case) 4 hash values collide with the 4 outputs of the public map. So first, you take P(s_4,x_4), parse it into (d_4,s_3), check that d_4=Tr(H(H(H(H(m))))); compute P(s_3,x_3), parse it into (d_3,s_2), check that d_3=Tr(H(H(H(m)))); and so on.

So the bit-complexity of this attack should be the number of bit-operations required to perform all of these checks times the number of checks needed to achieve a collision.

By my calculation, the GeMSS team seems to be selecting the number of rounds (4) so that the number of checks needed is 2^128, but I think that it is clear that this number of pairs is not required to achieve a complexity of 2^143 for this attack. I think that with three rounds the number of pairs required to achieve a collision is sufficiently high that multiplying it by the number of bit operations required for each of the above checks still gets them over 2^143.

This is what I'd like a check on. To make things easier, evaluation of P costs between 8 and 16 times as much as one hash call.

Thanks to anybody willing to back me up on this.

Cheers,
Daniel